# RV32M1 Series Reference Manual

Supports: RV32M1

Document Number: RV32M1RM
Rev. 1.1, 11/27/2018

# Contents

**Section number** **Title** **Page**

## Chapter 1
## About This Manual


1.1 Audience ..... 47

1.2 Organization ..... 47

1.3 Module descriptions ..... 47

1.3.1 Example: chip-specific information that supersedes content in the same chapter ..... 48

1.3.2 Example: chip-specific information that refers to a different chapter ..... 49

1.4 Register descriptions ..... 50

1.5 Conventions ..... 51

1.5.1 Numbering systems ..... 51

1.5.2 Typographic notation ..... 51

1.5.3 Special terms ..... 52


## Chapter 2
## Introduction


2.1 Overview ..... 53

2.2 Block diagram ..... 53

2.3 Module functional categories ..... 54

2.4 Ordering information ..... 57


## Chapter 3
## Core Overview


3.1 Core modules ..... 59

3.1.1 ARM Cortex-M0+ core ..... 59

3.1.2 ARM Cortex-M4F Core ..... 60

3.1.3 RISC-V RI5CY core ..... 62

3.1.4 RISC-V ZERO-RISCY core ..... 96

3.2 Nested Vectored Interrupt Controller (NVIC) modules ..... 113

3.2.1 Cortex-M0+ Nested Vectored Interrupt Controller (NVIC) ..... 113

3.2.2 Cortex-M4 Nested Vectored Interrupt Controller (NVIC) configuration ..... 119

3.3 Asynchronous Wake-up Interrupt Controller (AWIC)..... 126
3.3.1 AWIC overview..... 126
3.3.2 Wake-up sources..... 126
3.4 Event unit..... 128
3.4.1 Introduction..... 128
3.4.2 Memory map and register descriptions..... 129
3.4.3 Interrupt assignment to RI5CY..... 148
3.4.4 Interrupt assignment to ZERO-RISCY..... 150

## Chapter 4
## Peripheral Bridge (AIPS-Lite)

4.1 Chip-specific AIPS-Lite information..... 151
4.1.1 Memory maps..... 151
4.2 Memory map/register definition..... 151
4.3 Functional description..... 151
4.3.1 Access support..... 151

## Chapter 5
## Crossbar Switch (AXBS)

5.1 Chip-specific AXBS information..... 153
5.1.1 Crossbar switch master assignments - with system MPU..... 153
5.1.2 Crossbar switch slave assignments - with system MPU..... 153
5.2 Introduction..... 154
5.2.1 Features..... 154
5.3 Memory Map / Register Definition..... 155
5.3.1 AXBS0 register descriptions..... 155
5.4 Functional Description..... 162
5.4.1 General operation..... 162
5.4.2 Register coherency..... 163
5.4.3 Arbitration..... 164
5.5 Initialization/application information..... 167

## Chapter 6
## Miscellaneous Control Module for CM4F (MCM0)

6.1 Introduction....169
6.1.1 Features....169
6.2 Memory map/register descriptions....169
6.2.1 MCM register descriptions....169
6.3 Functional description....178
6.3.1 Interrupts....178
6.3.2 Determining source of the interrupt....179

## Chapter 7
## Miscellaneous Control Module for CM0+ (MCM1)

7.1 Introduction....181
7.1.1 Features....181
7.2 Memory map/register descriptions....181
7.2.1 MCM register descriptions....181

## Chapter 8
## Miscellaneous System Control Module (MSCM)

8.1 Chip-specific MSCM information....187
8.2 Overview....187
8.3 Chip Configuration and Boot....187
8.4 MSCM Memory Map/Register Definition....188
8.4.1 CPU Configuration Memory Map and Registers....188
8.4.2 MSCM register descriptions....188

## Chapter 9
## Extended Resource Domain Controller (XRDC)

9.1 Chip-specific XRDC information....233
9.1.1 XRDC configuration....233
9.1.2 XRDC register configurations....233
9.2 Introduction....239
9.2.1 Features....240

9.2.2 Block diagram.....241
9.2.3 Modes of operation.....242
9.3 External signal description.....243
9.4 Memory map/register definition.....243
9.4.1 XRDC register descriptions.....244
9.5 Functional description.....299
9.5.1 Manager (XRDC_MGR).....299
9.5.2 Master Domain Assignment Controller (XRDC_MDAC).....299
9.5.3 Special 4-State Model.....301
9.5.4 DxACP evaluation.....301
9.5.5 Exclusive Access Lock.....303
9.5.6 Memory Region Controller (XRDC_MRC).....305
9.5.7 Peripheral Access Controller (XRDC_PAC).....309
9.6 Initialization information.....311
9.7 Application information.....311
9.7.1 Master domain assignments.....311
9.7.2 Memory region descriptor management.....312
9.7.3 Domain error capture management.....312

## Chapter 10
## Semaphores2 (SEMA42)

10.1 Introduction.....315
10.1.1 Multi-core programming 101: software gates.....315
10.1.2 Features.....317
10.2 Memory map/register definition.....318
10.2.1 SEMA420 register descriptions.....319
10.2.2 SEMA421 register descriptions.....324
10.3 Functional description.....330

## Chapter 11
## Memory-Mapped Divide and Square Root (MMDVSQ)

11.1 Introduction....333
11.1.1 Features....333
11.1.2 Block diagram....334
11.1.3 Modes of operation....335
11.2 External signal description....336
11.3 Memory map and register definition....336
11.3.1 MMDVSQ register descriptions....336
11.4 Functional description....343
11.4.1 Algorithms....343
11.4.2 Execution times....347
11.4.3 Software interface....348

## Chapter 12
## ARM Core Debug

12.1 Introduction....351
12.2 Debug system block diagram....351
12.2.1 Debug components....352
12.2.2 JTAG-to-SWD change sequence....354
12.3 Debug port pin descriptions....354
12.4 System TAP connection....355
12.5 JTAG/SWD status and control registers....355
12.5.1 MDM-AP status registers....357
12.5.2 MDM-AP Control Register....359
12.5.3 MDM-AP Core Status Register....360
12.5.4 MDM-AP Core Clock Status Register....360
12.5.5 MDM-AP System Clock Status Register....361
12.5.6 MDM-AP System Power Status Register....362
12.5.7 MDM-AP PCC0_CGC0 Register....363
12.5.8 MDM-AP PCC0_CGC1 Register....363
12.5.9 MDM-AP PCC0_CGC2 Register....363

12.5.10 MDM-AP PCC0_CGC3 Register ... 363
12.5.11 MDM-AP PCC0_CGC4 Register ... 364
12.5.12 MDM-AP PCC1_CGC0 Register ... 364
12.5.13 MDM-AP PCC1_CGC1 Register ... 364
12.5.14 MDM-AP PCC1_CGC2 Register ... 364
12.5.15 MDM-AP PCC1_CGC3 Register ... 365
12.5.16 MDM-AP PCC1_CGC4 Register ... 365
12.6 Debug resets ... 365
12.7 SWD Multi-Drop ... 366
12.8 ITM ... 366
12.9 Core trace connectivity with ETM ... 367
12.10 Embedded Trace Macrocell v3.5 (ETM) ... 367
12.11 TPIU ... 368
12.12 DWT ... 368
12.13 Micro Trace Buffer (MTB) ... 369
12.14 Debug in low-power modes ... 369
12.15 Debug module state in low power modes ... 370
12.16 Debug and security ... 371

**Chapter 13
RISC-V Core Debug**

13.1 Introduction ... 373
13.2 Memory map for RISC-V debug port ... 373
13.3 Advanced debug interface ... 374
13.3.1 Error register ... 375
13.3.2 DEBUG_PATH_SELECTION register ... 375
13.3.3 POWER_UP register ... 376

**Chapter 14
Micro Trace Buffer (MTB)**

14.1 Introduction ... 377

14.1.1 Overview....377
14.1.2 Features....380
14.1.3 Modes of operation....381
14.2 External signal description....381
14.3 Memory map and register definition....382
14.3.1 MTB register descriptions....382
14.3.2 DWT register descriptions....400
14.3.3 ROM register descriptions....413

## Chapter 15
## JTAG Controller (JTAGC)

15.1 Introduction....421
15.1.1 Block diagram....421
15.1.2 Features....422
15.1.3 Modes of operation....422
15.2 External signal description....423
15.2.1 TCK—Test clock input....424
15.2.2 TDI—Test data input....424
15.2.3 TDO—Test data output....424
15.2.4 TMS—Test mode select....424
15.3 Register description....424
15.3.1 Instruction register....424
15.3.2 Bypass register....425
15.3.3 Device identification register....425
15.3.4 Boundary scan register....426
15.4 Functional description....426
15.4.1 JTAGC reset configuration....427
15.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port....427
15.4.3 TAP controller state machine....427
15.4.4 JTAGC block instructions....429

15.4.5 Boundary scan....432
15.5 Initialization/Application information....432

## Chapter 16
## Memory Map

16.1 Introduction....433
16.2 System memory map....433
16.3 Flash memory map....435
16.4 Flash access control introduction....436
16.5 SRAM memory map....436
16.6 Peripheral bridge memory maps....437
16.6.1 ARM Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge memory maps....437
16.6.2 ARM Cortex-M4RISC-V RI5CY peripheral bridge memory maps....441
16.7 Private Peripheral Bus (PPB) memory map....445
16.7.1 ARM Cortex-M0+ PPB memory map....445
16.7.2 ARM Cortex-M4 PPB memory map....446

## Chapter 17
## Flash Memory Module (FTFE)

17.1 Chip-specific FTFE information....449
17.1.1 Flash Option (FOPT) customization....449
17.2 External signal description....451
17.3 Memory map and registers....451
17.3.1 IFR descriptions....452
17.3.2 Register descriptions....456
17.4 Functional description....481
17.4.1 Primary Program flash memory swap....481
17.4.2 Flash Program/Erase Protection....482
17.4.3 Flash Access Protection....483
17.4.4 Interrupts....484
17.4.5 Flash Operation in Low-Power Modes....485

17.4.6 Flash memory reads and ignored writes..... 485
17.4.7 Read while write (RWW)..... 486
17.4.8 Flash Program and Erase.....486
17.4.9 Flash Command Operations.....486
17.4.10 Margin Read Commands..... 493
17.4.11 Flash command descriptions.....494
17.4.12 Security..... 527
17.4.13 Reset Sequence..... 529

## Chapter 18
## Flash Memory Controller (FMC)

18.1 Chip-specific FMC information.....531
18.1.1 FMC buffers control.....531
18.1.2 Flash Access Control (FAC) Function.....531
18.2 Introduction.....538
18.2.1 Overview.....538
18.2.2 Features..... 538
18.3 Modes of operation..... 539
18.4 External signal description.....539
18.5 Functional description.....539
18.5.1 Default configuration..... 539
18.5.2 Configuration options..... 540
18.5.3 Wait states..... 540
18.5.4 Speculative reads.....540
18.6 Initialization and application information.....541

## Chapter 19
## Flash security

19.1 Introduction.....543
19.2 Flash Security..... 543
19.3 Security Interactions with other Modules..... 544

19.3.1 Security Interactions with Debug....544

## Chapter 20
## External Bus Interface (FlexBus)

20.1 Chip-specific FlexBus information....545
20.1.1 FlexBus clocking....545
20.1.2 FlexBus signal multiplexing....545
20.1.3 FlexBus CSCR0 reset value....547
20.1.4 FlexBus Security....547
20.1.5 FlexBus line transfers....547
20.2 Introduction....547
20.3 Signal descriptions....549
20.4 Memory Map/Register Definition....551
20.4.1 FB register descriptions....552
20.5 Functional description....561
20.5.1 Address and data bus configurations....561
20.5.2 Address comparison....561
20.5.3 Address driven on address bus....562
20.5.4 Connecting address/data lines....562
20.5.5 Bit ordering....562
20.5.6 Global chip-select....562
20.5.7 Data transfer signals....563
20.5.8 Signal transitions....563
20.5.9 Data-byte alignment and physical connections....563
20.5.10 Address/data bus multiplexing....565
20.5.11 Data transfer states....566
20.5.12 FlexBus Timing Examples....567
20.5.13 Burst cycles....586
20.5.14 Misaligned operands....595
20.5.15 Extended Transfer Start/Address Latch Enable....596

20.5.16 Bus errors ... 597
20.6 Initialization/Application Information ... 598
20.6.1 Initializing a chip-select ... 598
20.6.2 Reconfiguring a chip-select ... 598

## Chapter 21
## AHB Low Power Cache Controller (AHB_LPCAC)

21.1 Chip-specific LPCAC information ... 601
21.1.1 Cache controls ... 601
21.2 Introduction ... 601
21.2.1 Overview ... 601
21.3 Signal descriptions ... 602
21.4 Memory map and Input Control description ... 602
21.4.1 Memory regions ... 602
21.4.2 Input controls ... 602
21.5 Functional description ... 603
21.5.1 Cache Functional description ... 603

## Chapter 22
## AHB Low Power Local Memory Controller (AHB_LPLMEM)

22.1 Chip-specific LPLMEM information ... 605
22.1.1 Cache controls ... 605
22.2 Introduction ... 605
22.2.1 Overview ... 605
22.3 Signal descriptions ... 606
22.4 Memory map and Input Control description ... 606
22.4.1 Memory regions ... 606
22.4.2 Input controls ... 606
22.5 Functional description ... 606
22.5.1 Cache Functional description ... 607

## Chapter 23
## Signal Multiplexing and Signal Descriptions

23.1 Signal Multiplexing Introduction....609
23.2 Clock gating....609
23.3 Pinouts....610
23.3.1 RV32M1 Subfamily Pinout....610
23.3.2 Pin properties....619
23.3.3 Pinouts diagram....629
23.4 Module Signal Description Tables....630
23.4.1 Core Modules....630
23.4.2 System Modules....631
23.4.3 Memories and Memory Interfaces....632
23.4.4 Clock Modules....633
23.4.5 Communication Interfaces....634
23.4.6 Analog....640
23.4.7 Human-Machine Interfaces (HMI)....641
23.4.8 Timer Modules....641
23.4.9 Security Modules....643

## Chapter 24
## Port Control and Interrupts (PORT)

24.1 Introduction....645
24.1.1 Overview....645
24.1.2 Features....645
24.1.3 Modes of operation....646
24.2 External signal description....646
24.2.1 Detailed signal description....647
24.3 Memory map and register definition....647
24.3.1 PORTA register descriptions....647
24.3.2 PORTB register descriptions....664
24.3.3 PORTC register descriptions....673
24.3.4 PORTD register descriptions....687

24.3.5 PORTE register descriptions....701
24.4 Functional description....716
24.4.1 Pin control....716
24.4.2 Global pin control....717
24.4.3 Global interrupt control....717
24.4.4 External interrupts....718

## Chapter 25
## General-Purpose Input/Output (GPIO)

25.1 Chip-specific GPIO information....721
25.1.1 GPIO signals....721
25.2 Introduction....721
25.2.1 Features....721
25.2.2 Modes of operation....722
25.2.3 GPIO signal descriptions....722
25.3 Memory map and register definition....723
25.3.1 GPIO register descriptions....723
25.4 FGPIO memory map and register definition....730
25.4.1 FGPIOE register descriptions....731
25.5 Functional description....737
25.5.1 General-purpose input....737
25.5.2 General-purpose output....737
25.5.3 IOPORT....738

## Chapter 26
## Reset and Boot

26.1 Introduction....739
26.2 Reset....740
26.2.1 Voltage monitor reset sources....740
26.2.2 External system reset sources....742
26.2.3 Internal system reset sources....743

26.2.4 Debug resets....746
26.3 Boot....747
26.3.1 Boot sources....747
26.3.2 Flash Option (FOPT) customization....747
26.3.3 Boot sequence....750

## Chapter 27
## Kinetis ROM Bootloader

27.1 Chip-specific information....753
27.1.1 Also see the Kinetis Bootloader Reference Manual....753
27.1.2 Kinetis bootloader peripheral pinmux....753
27.1.3 Bootloader Memory Access....754
27.2 Introduction....754
27.3 Functional Description....756
27.3.1 The Kinetis Bootloader Configuration Area (BCA)....756
27.3.2 Start-up Process....758
27.3.3 Bootloader Entry Point / API Tree....760
27.3.4 Bootloader Protocol....762
27.3.5 Bootloader Packet Types....767
27.3.6 Bootloader Command API....775
27.3.7 Bootloader Exit state....799
27.4 Kinetis Flash Driver API....799
27.4.1 Flash Driver Entry Point....800
27.4.2 Flash driver API Tree....800
27.4.3 Quick demo using Kinetis Flash Driver API....801
27.4.4 Flash driver data structures....802
27.4.5 Flash driver API....803
27.5 Peripherals Supported....816
27.5.1 LPI2C Peripheral....816
27.5.2 LPSPI Peripheral....818

27.5.3 LPUART Peripheral....821
27.5.4 USB peripheral....823
27.6 Get/SetProperty Command Properties....827
27.6.1 Property Definitions....829
27.7 ROM API....831
27.7.1 ROM API....831
27.7.2 ROM API RAM usage....832
27.8 Security....833
27.8.1 Introduction....833
27.8.2 Notation....834
27.8.3 Use cases....835
27.8.4 Security fields....836
27.8.5 Signed image structure....839
27.8.6 SB file formats....846
27.8.7 Using SB files....858
27.8.8 Security considerations....862
27.8.9 Bootable section contents....864
27.9 Verifying the application in flash using CRC-32....870
27.10 Kinetis Bootloader Status Error Codes....871

## Chapter 28
## Clock Distribution

28.1 Introduction....875
28.2 Clock control modules....875
28.2.1 System Clock Generation (SCG) module....876
28.2.2 Peripheral Clock Control (PCC) module....877
28.3 SCG input clock sources....878
28.4 SCG system output clocks....878
28.5 SCG peripheral output clocks....879
28.6 Clock definitions....880

28.7 Clock gating ..... 881
28.8 Module clock assignments ..... 882

## Chapter 29
## System Clock Generator (SCG)

29.1 Chip-specific SCG information ..... 887
29.1.1 LPFLL configuration ..... 887
29.2 Introduction ..... 887
29.2.1 Features ..... 887
29.3 Memory Map/Register Definition ..... 889
29.3.1 SCG register descriptions ..... 890
29.4 Functional description ..... 927
29.4.1 SCG Clock Mode Transitions ..... 927

## Chapter 30
## Peripheral Clock Controller (PCC)

30.1 Chip-specific PCC information ..... 931
30.1.1 Peripheral source to the PCC ..... 931
30.2 Introduction ..... 931
30.3 Features ..... 931
30.4 Functional description ..... 932
30.5 Memory map and register definition ..... 933
30.5.1 PCC0 register descriptions ..... 933
30.5.2 PCC1 register descriptions ..... 995

## Chapter 31
## Power Management

31.1 Introduction ..... 1025
31.2 Power supply and power domain summary ..... 1025
31.2.1 Power supply ..... 1025
31.2.2 Power domains ..... 1028
31.3 On-chip regulator summary ..... 1030
31.3.1 DCDC regulator ..... 1031

31.3.2 LPPMC regulator ... 1031
31.3.3 IOREG regulator ... 1031
31.3.4 USB regulator ... 1031
31.4 Typical power supply schematic ... 1032
31.4.1 Typical power supply schematic 1 ... 1032
31.4.2 Typical power supply schematic 2 ... 1032
31.4.3 Typical power supply schematic 3 ... 1033
31.5 Power up state ... 1034
31.6 Multi-Core power mode description ... 1034
31.6.1 Core power structure overview ... 1034
31.6.2 Power mode combinations ... 1035
31.6.3 Recommended power mode combination ... 1039
31.7 Power mode description ... 1040
31.7.1 RUN mode ... 1040
31.7.2 Wait mode ... 1040
31.7.3 Stop mode ... 1041
31.7.4 Standby (LLS) mode ... 1042
31.7.5 Shutdown (VLLSx) mode ... 1042
31.8 Entering and exiting power modes ... 1043
31.9 Power mode transitions ... 1044
31.10 Power modes shutdown sequencing ... 1045
31.11 Flash program restrictions ... 1046
31.12 Module operation in low power modes ... 1046

## Chapter 32
## Multi System Mode Controller (MSMC)

32.1 Chip-specific MSMC information ... 1051
32.1.1 SRAM blocks with corresponding control bit ... 1051
32.1.2 Power mode transition in dual core application ... 1051
32.2 Introduction ... 1052

32.3 Modes of operation.... 1053
32.4 Memory map and register descriptions.... 1054
32.4.1 SMC0 register descriptions.... 1055
32.4.2 SMC1 register descriptions.... 1076
32.5 Functional description.... 1097
32.5.1 Power mode transitions.... 1097
32.5.2 Run modes.... 1102
32.5.3 Wait modes.... 1104
32.5.4 Stop modes.... 1105
32.5.5 Debug in low power modes.... 1108

## Chapter 33
## System Power Manager (SPM)

33.1 Chip-specific SPM information.... 1111
33.1.1 VDDIO voltage monitor.... 1111
33.2 Introduction.... 1111
33.3 Features.... 1111
33.4 Memory map and register descriptions.... 1113
33.4.1 SPM register descriptions.... 1113
33.5 Functional Description.... 1150
33.5.1 Low-voltage detect (LVD) system.... 1150
33.5.2 High-voltage detect (HVD) system.... 1152
33.5.3 Stop and VLPx Modes.... 1152
33.5.4 I/O retention.... 1153
33.5.5 Regulators.... 1153

## Chapter 34
## Low-Leakage Wakeup Unit (LLWU)

34.1 Chip-specific LLWU information.... 1159
34.1.1 Wake-up sources for Cortex-M4 core (CPU0).... 1159
34.1.2 Wake-up sources for Cortex-M0+ core (CPU1).... 1161

34.2 Introduction....1163
34.2.1 Features....1163
34.2.2 Modes of operation....1163
34.2.3 Block diagram....1164
34.3 LLWU signal descriptions....1165
34.4 Memory map/register definition....1166
34.4.1 LLWU0 register descriptions....1166
34.4.2 LLWU1 register descriptions....1185
34.5 Functional description....1204
34.5.1 LLS mode....1204
34.5.2 VLLS modes....1205
34.5.3 Initialization....1205

## Chapter 35
## System Integration Module (SIM)

35.1 Introduction....1207
35.1.1 Overview....1207
35.1.2 Features....1207
35.2 Memory Map and Registers....1207
35.2.1 SIM register descriptions....1207
35.3 Application information....1219

## Chapter 36
## Interrupt Multiplexer (INTMUX)

36.1 About this module....1221
36.1.1 Introduction....1221
36.1.2 Features....1221
36.1.3 Block diagram....1221
36.2 Memory Map and register definition....1222
36.2.1 INTMUX0 register descriptions....1222
36.2.2 INTMUX1 register descriptions....1228

36.3 Functional Description....1233
36.3.1 Configuring Output Channels....1233
36.3.2 INTMUX Vectors....1234

## Chapter 37
## Trigger MUX (TRGMUX)

37.1 Chip-specific TRGMUX information....1235
37.1.1 CPU0 Trigger Multiplexer input sources....1235
37.1.2 CPU0 Trigger Multiplexer outputs....1237
37.1.3 CPU1 Trigger Multiplexer input sources....1238
37.1.4 CPU1 Trigger Multiplexer outputs....1240
37.2 Introduction....1241
37.3 Features....1241
37.4 Memory map and register definition....1241
37.4.1 TRGMUX0 register descriptions....1242
37.4.2 TRGMUX1 register descriptions....1279

## Chapter 38
## Enhanced Direct Memory Access (eDMA)

38.1 Chip-specific eDMA information....1319
38.2 Introduction....1319
38.2.1 eDMA system block diagram....1319
38.2.2 Block parts....1320
38.2.3 Features....1321
38.3 Modes of operation....1322
38.4 Memory map/register definition....1323
38.4.1 TCD memory....1323
38.4.2 TCD initialization....1323
38.4.3 TCD structure....1323
38.4.4 Reserved memory and bit fields....1324
38.4.5 DMA0 register descriptions....1324

38.4.6 DMA1 register descriptions....1382
38.5 Functional description....1431
38.5.1 eDMA basic data flow....1431
38.5.2 Fault reporting and handling....1434
38.5.3 Channel preemption....1437
38.5.4 Performance....1437
38.6 Initialization/application information....1441
38.6.1 eDMA initialization....1441
38.6.2 Programming errors....1443
38.6.3 Arbitration mode considerations....1444
38.6.4 Performing DMA transfers....1444
38.6.5 Monitoring transfer descriptor status....1448
38.6.6 Channel Linking....1450
38.6.7 Dynamic programming....1451
38.6.8 Suspend/resume a DMA channel with active hardware service requests....1455

## Chapter 39
## Direct Memory Access Multiplexer (DMAMUX)

39.1 Chip-specific DMAMUX information....1457
39.1.1 ARM Cortex-M0+ DMAMUX configuration....1457
39.1.2 ARM Cortex-M4 DMAMUX configuration....1459
39.2 Introduction....1461
39.2.1 Overview....1461
39.2.2 Features....1462
39.2.3 Modes of operation....1462
39.3 External signal description....1463
39.4 Memory map/register definition....1463
39.4.1 DMAMUX0 register descriptions....1463
39.4.2 DMAMUX1 register descriptions....1465
39.5 Functional description....1467

39.5.1 DMA channels with periodic triggering capability....1467
39.5.2 Always-enabled DMA sources....1470
39.6 Initialization/application information....1471
39.6.1 Reset....1471
39.6.2 Enabling and configuring sources....1471

## Chapter 40
## Messaging Unit (MU)

40.1 Overview....1475
40.1.1 Features....1476
40.1.2 Modes of Operation....1476
40.2 Register Definition....1476
40.2.1 MUA register descriptions....1477
40.2.2 MUB register descriptions....1491
40.3 Functional Description....1505
40.3.1 Processor A Side Memory-Mapping....1505
40.3.2 Processor B Side Memory-Mapping....1506
40.3.3 MU Messaging....1506
40.3.4 Low Power Modes....1508
40.3.5 Event Update Timing....1509
40.3.6 Interrupts....1509
40.3.7 Interrupt Messaging Protocols....1510
40.3.8 Exclusive Access to Shared Memory....1513
40.3.9 Packet Data Transfers....1515
40.3.10 Resets....1515
40.4 Software Restrictions....1516
40.4.1 General Restrictions....1516
40.4.2 Processor Restrictions....1517

## Chapter 41
## Watchdog timer (WDOG)

41.1 Chip-specific WDOG information....1519
41.1.1 WDOG clocks....1519
41.1.2 WDOG low-power modes....1519
41.2 Introduction....1520
41.2.1 Features....1520
41.2.2 Block diagram....1521
41.3 Memory map and register definition....1521
41.3.1 WDOG0 register descriptions....1521
41.3.2 WDOG1 register descriptions....1528
41.4 Functional description....1534
41.4.1 Clock source....1535
41.4.2 Watchdog refresh mechanism....1536
41.4.3 Configuring the Watchdog....1538
41.4.4 Using interrupts to delay resets....1539
41.4.5 Backup reset....1539
41.4.6 Functionality in debug and low-power modes....1540
41.4.7 Fast testing of the watchdog....1540
41.5 Application Information....1542
41.5.1 Disable Watchdog....1542
41.5.2 Disable Watchdog after Reset....1542
41.5.3 Configure Watchdog....1543
41.5.4 Refreshing the Watchdog....1543

## Chapter 42
## External Watchdog Monitor (EWM)

42.1 Chip-specific EWM information....1545
42.1.1 EWM clocks....1545
42.1.2 EWM low-power modes....1545
42.1.3 EWM_OUT pin state in low power modes....1545
42.2 Introduction....1546

42.2.1 Features ... 1546
42.2.2 Modes of Operation ... 1547
42.2.3 Block Diagram ... 1547
42.3 EWM Signal Descriptions ... 1548
42.4 Memory Map/Register Definition ... 1549
42.4.1 EWM register descriptions ... 1549
42.5 Functional Description ... 1554
42.5.1 The EWM_OUT_b Signal ... 1554
42.5.2 EWM_OUT_b pin state in low power modes ... 1555
42.5.3 The EWM_in Signal ... 1555
42.5.4 EWM Counter ... 1556
42.5.5 EWM Compare Registers ... 1556
42.5.6 EWM Refresh Mechanism ... 1556
42.5.7 Counter clock prescaler ... 1557

## Chapter 43
## Cyclic Redundancy Check (CRC)

43.1 Introduction ... 1559
43.1.1 Features ... 1559
43.1.2 Block diagram ... 1559
43.1.3 Modes of operation ... 1560
43.2 Memory map and register descriptions ... 1560
43.2.1 CRC register descriptions ... 1560
43.3 Functional description ... 1565
43.3.1 CRC initialization/reinitialization ... 1565
43.3.2 CRC calculations ... 1565
43.3.3 Transpose feature ... 1566
43.3.4 CRC result complement ... 1568

## Chapter 44
## Standalone TRNG

44.1 Standalone True Random Number Generator (SA-TRNG)....1569
44.1.1 Standalone True Random Number Generator Block Diagram....1569
44.1.2 TRNG Functional Description....1570
44.1.3 SA-TRNG hardware functional description....1571
44.1.4 Another TRNG usage example....1573
44.1.5 TRNG register descriptions....1573

## Chapter 45
## Cryptographic Acceleration Unit (CAU3)

45.1 Introduction....1625
45.2 Block Diagram....1627
45.3 Overview....1628
45.4 Features....1629
45.5 CAU3 register descriptions....1630
45.5.1 CAU3 Memory map....1630
45.5.2 Processor Core Type (PCT)....1631
45.5.3 Memory Configuration (MCFG)....1632
45.5.4 Control Register (CR)....1634
45.5.5 Status Register (SR)....1638
45.5.6 Debug Control/Status Register (DBGCSR)....1641
45.5.7 Debug PC Breakpoint Register (DBGPBR)....1644
45.5.8 Debug Memory Command Register (DBGMCMD)....1645
45.5.9 Debug Memory Address Register (DBGMADR)....1647
45.5.10 Debug Memory Data Register (DBGMDR)....1648
45.5.11 Semaphore Register (SEMA4)....1649
45.5.12 Semaphore Ownership Register (SMOWNR)....1651
45.5.13 Address Remap Register (ARR)....1653
45.5.14 CryptoCore General Purpose Registers (CC_R0 - CC_R29)....1654
45.5.15 General Purpose R30 (CC_R30)....1655
45.5.16 General Purpose R31 (CC_R31)....1656

45.5.17 Program Counter (CC_PC)..... 1657
45.5.18 Start Command Register (CC_CMD)..... 1658
45.5.19 Condition Flag (CC_CF)..... 1659
45.5.20 Mode Register (PublicKey) (MDPK)..... 1660
45.5.21 Command Register (COM)..... 1662
45.5.22 Control Register (CTL)..... 1664
45.5.23 Clear Written Register (CW)..... 1666
45.5.24 Status Register (STA)..... 1668
45.5.25 Error Status Register (ESTA)..... 1670
45.5.26 PKHA A Size Register (PKASZ)..... 1672
45.5.27 PKHA B Size Register (PKBSZ)..... 1673
45.5.28 PKHA N Size Register (PKNSZ)..... 1674
45.5.29 PKHA E Size Register (PKESZ)..... 1675
45.5.30 PKHA Revision ID 1 (PKHA_VID1)..... 1676
45.5.31 PKHA Revision ID 2 (PKHA_VID2)..... 1677
45.5.32 CHA Revision ID (CHA_VID)..... 1678
45.5.33 PKHA Clock Control Register (PKHA_CCR)..... 1679
45.5.34 Global Status Register (GSR)..... 1681
45.5.35 Clock Linear Feedback Shift Register (CKLFSR)..... 1683
45.5.36 PKHA A0 Register (PKA0_0 - PKA0_31)..... 1684
45.5.37 PKHA A1 Register (PKA1_0 - PKA1_31)..... 1685
45.5.38 PKHA A2 Register (PKA2_0 - PKA2_31)..... 1686
45.5.39 PKHA A3 Register (PKA3_0 - PKA3_31)..... 1687
45.5.40 PKHA B0 Register (PKB0_0 - PKB0_31)..... 1689
45.5.41 PKHA B1 Register (PKB1_0 - PKB1_31)..... 1690
45.5.42 PKHA B2 Register (PKB2_0 - PKB2_31)..... 1691
45.5.43 PKHA B3 Register (PKB3_0 - PKB3_31)..... 1692
45.5.44 PKHA N0 Register (PKN0_0 - PKN0_31)..... 1694
45.5.45 PKHA N1 Register (PKN1_0 - PKN1_31)..... 1695

45.5.46 PKHA N2 Register (PKN2_0 - PKN2_31)....1696
45.5.47 PKHA N3 Register (PKN3_0 - PKN3_31)....1697
45.5.48 PKHA E Register (PKE_0 - PKE_127)....1698
45.6 Functional description....1700
45.6.1 CryptoCore States....1700
45.6.2 Memory Map....1701
45.6.3 Access to Cryptographic Routines....1702
45.6.4 Integrity checks....1703
45.6.5 DMA and Address Remapping....1703
45.6.6 Semaphore Operation....1705
45.6.7 Debug Resources....1707
45.6.8 Virtual External Registers....1709
45.6.9 Security Violation....1711
45.6.10 Module Reset....1712
45.6.11 Handling Exceptions....1713
45.6.12 Access to Memory Mapped Registers....1715
45.6.13 Low Power Operation....1717
45.7 Public-key hardware accelerator (PKHA) functionality....1719
45.7.1 Modular math....1720
45.7.2 About Montgomery values....1720
45.7.3 Non-modular Math....1722
45.7.4 Elliptic-Curve Math....1722
45.7.5 PKHA Mode Register....1725
45.7.6 PKHA functions....1737
45.7.7 Special values for common ECC domains....1771
45.8 Application/initialization information....1790
45.8.1 Code example....1790
45.8.2 Assembler equate values....1792

## Chapter 46

## Analog-to-Digital Converter (ADC)


46.1 Chip-specific LPADC information....1797

46.1.1 LPADC0 channel assignments....1797

46.1.2 Triggers....1799

46.1.3 Voltage reference options....1799

46.1.4 Functional clock options....1800

46.1.5 Direct Memory Access (DMA) support....1800

46.2 Introduction....1800

46.2.1 Features....1801

46.2.2 Block diagram....1801

46.2.3 Modes of operation....1802

46.3 Signal descriptions....1803

46.3.1 Analog Power (VDDA)....1803

46.3.2 Analog Ground (VSSA)....1804

46.3.3 Analog Channel Inputs (CHnA and CHnB)....1804

46.4 LPADC register descriptions....1804

46.4.1 LPADC Memory map....1804

46.4.2 Version ID Register (VERID)....1806

46.4.3 Parameter Register (PARAM)....1808

46.4.4 LPADC Control Register (CTRL)....1809

46.4.5 LPADC Status Register (STAT)....1810

46.4.6 Interrupt Enable Register (IE)....1812

46.4.7 DMA Enable Register (DE)....1813

46.4.8 LPADC Configuration Register (CFG)....1814

46.4.9 LPADC Pause Register (PAUSE)....1817

46.4.10 LPADC FIFO Control Register (FCTRL)....1818

46.4.11 Software Trigger Register (SWTRIG)....1819

46.4.12 LPADC Offset Trim Register (OFSTRIM)....1820

46.4.13 Trigger Control Register (TCTRL0 - TCTRL3)....1822

46.4.14 LPADC Command Low Buffer Register (CMDL1 - CMDL15)....1823
46.4.15 LPADC Command High Buffer Register (CMDH1 - CMDH4)....1825
46.4.16 LPADC Command High Buffer Register (CMDH5 - CMDH15)....1828
46.4.17 Compare Value Register (CV1 - CV4)....1831
46.4.18 LPADC Data Result FIFO Register (RESFIFO)....1832
46.5 Functional description....1834
46.5.1 Voltage reference....1837
46.5.2 Power control....1837
46.5.3 Clock operation....1838
46.5.4 Trigger detect and command execution....1838
46.5.5 Pause option....1839
46.5.6 Result FIFO operation....1840
46.5.7 Compare function....1840

## Chapter 47
## Low Power Comparator (LPCMP)

47.1 Chip-specific LPCMP information....1843
47.1.1 LPCMP input connections....1843
47.1.2 LPCMP external references....1843
47.2 Introduction....1844
47.3 Features....1844
47.4 LPCMP Block Diagram....1845
47.5 LPCMP0 register descriptions....1846
47.5.1 LPCMP0 Memory map....1847
47.5.2 Version ID Register (VERID)....1847
47.5.3 Parameter Register (PARAM)....1848
47.5.4 Comparator Control Register 0 (CCR0)....1849
47.5.5 Comparator Control Register 1 (CCR1)....1850
47.5.6 Comparator Control Register 2 (CCR2)....1852
47.5.7 DAC Control Register (DCR)....1854

47.5.8 Interrupt Enable Register (IER)..... 1855
47.5.9 Comparator Status Register (CSR)..... 1856
47.6 LPCMP1 register descriptions..... 1857
47.6.1 LPCMP1 Memory map..... 1857
47.6.2 Version ID Register (VERID)..... 1858
47.6.3 Parameter Register (PARAM)..... 1859
47.6.4 Comparator Control Register 0 (CCR0)..... 1860
47.6.5 Comparator Control Register 1 (CCR1)..... 1861
47.6.6 Comparator Control Register 2 (CCR2)..... 1863
47.6.7 DAC Control Register (DCR)..... 1865
47.6.8 Interrupt Enable Register (IER)..... 1866
47.6.9 Comparator Status Register (CSR)..... 1867
47.7 LPCMP Function Description..... 1868
47.7.1 Functional Block Diagram..... 1868
47.7.2 Functional Modes..... 1870
47.7.3 Hysteresis..... 1879
47.7.4 Initialization..... 1880
47.7.5 Low-Pass Filter..... 1881
47.7.6 Interrupts..... 1882
47.7.7 DMA Support..... 1882
47.7.8 Low Power Mode Operation..... 1883

## Chapter 48
## 12-bit Low Power Digital-to-Analog Converter (LPDAC)

48.1 Chip-specific LPDAC information..... 1885
48.1.1 12-bit LPDAC Overview..... 1885
48.1.2 12-bit LPDAC Output..... 1885
48.1.3 12-bit LPDAC Reference..... 1885
48.2 Introduction..... 1886
48.3 Features..... 1886

48.4 Block diagram....1886
48.5 LPDAC register descriptions....1887
48.5.1 LPDAC Memory map....1888
48.5.2 Version Identifier Register (VERID)....1888
48.5.3 Parameter Register (PARAM)....1889
48.5.4 DAC Data Register (DATA)....1890
48.5.5 DAC Global Control Register (GCR)....1891
48.5.6 DAC FIFO Control Register (FCR)....1893
48.5.7 DAC FIFO Pointer Register (FPR)....1894
48.5.8 FIFO Status Register (FSR)....1895
48.5.9 DAC Interrupt Enable Register (IER)....1897
48.5.10 DAC DMA Enable Register (DER)....1898
48.5.11 DAC Reset Control Register (RCR)....1899
48.5.12 DAC Trigger Control Register (TCR)....1900
48.6 Functional description....1901
48.6.1 Buffer mode....1902
48.6.2 FIFO mode....1902
48.6.3 Swing back mode....1902
48.6.4 Interrupts....1903
48.6.5 DMA operation....1903
48.6.6 Low-power mode operation....1903

## Chapter 49
## Voltage Reference (VREF)

49.1 Chip-specific VREF information....1905
49.1.1 VREF Overview....1905
49.2 Introduction....1905
49.2.1 Overview....1906
49.2.2 Features....1907
49.2.3 Modes of Operation....1907

49.2.4 VREF Signal Descriptions ... 1907
49.3 Memory Map and Register Definition ... 1908
49.3.1 VREF register descriptions ... 1908
49.4 Functional Description ... 1912
49.4.1 Voltage Reference Disabled, SC[VREFEN] = 0 ... 1912
49.4.2 Voltage Reference Enabled, SC[VREFEN] = 1 ... 1913
49.4.3 Internal voltage regulator ... 1914
49.5 Initialization/Application Information ... 1915

## Chapter 50
## Low Power Interrupt Timer (LPIT)

50.1 Introduction ... 1917
50.1.1 Overview ... 1917
50.1.2 Block Diagram ... 1918
50.2 Modes of operation ... 1919
50.3 Memory Map and Registers ... 1920
50.3.1 LPIT0 register descriptions ... 1920
50.3.2 LPIT1 register descriptions ... 1935
50.4 Functional description ... 1950
50.4.1 LPIT programming model ... 1951
50.4.2 Initialization ... 1952
50.4.3 Timer Modes ... 1953
50.4.4 Trigger Control for Timers ... 1954
50.4.5 Channel Chaining ... 1954
50.4.6 Detailed timing ... 1955

## Chapter 51
## Timer/PWM Module (TPM)

51.1 Chip-specific TPM information ... 1969
51.1.1 TPM Instantiation Information ... 1969
51.1.2 Global Timebase ... 1969

51.1.3 TPM Interrupts....1969
51.2 Introduction....1970
51.2.1 TPM Philosophy....1970
51.2.2 Features....1970
51.2.3 Modes of operation....1971
51.2.4 Block diagram....1971
51.3 TPM Signal Descriptions....1972
51.3.1 TPM_EXTCLK — TPM External Clock....1973
51.3.2 TPM_CHn — TPM Channel (n) I/O Pin....1973
51.4 Memory Map and Register Definition....1973
51.4.1 TPM0 register descriptions....1973
51.4.2 TPM1 register descriptions....1997
51.4.3 TPM2 register descriptions....2018
51.4.4 TPM3 register descriptions....2042
51.5 Functional description....2064
51.5.1 Clock domains....2064
51.5.2 Prescaler....2064
51.5.3 Counter....2065
51.5.4 Input Capture Mode....2068
51.5.5 Output Compare Mode....2069
51.5.6 Edge-Aligned PWM (EPWM) Mode....2070
51.5.7 Center-Aligned PWM (CPWM) Mode....2072
51.5.8 Combine PWM mode....2074
51.5.9 Combine Input Capture mode....2077
51.5.10 Input Capture Filter....2078
51.5.11 Deadtime insertion....2080
51.5.12 Quadrature Decoder mode....2081
51.5.13 Registers Updated from Write Buffers....2085
51.5.14 DMA....2086

51.5.15 Output triggers ... 2086
51.5.16 Reset Overview ... 2087
51.5.17 TPM Interrupts ... 2087

## Chapter 52
## Time Stamp Timer Module (TSTMR)

52.1 Chip-specific TSTMR information ... 2089
52.2 Introduction ... 2089
52.2.1 Features ... 2089
52.3 TSTMR Memory map and register definition ... 2089
52.3.1 TSTMRA register descriptions ... 2089
52.3.2 TSTMRB register descriptions ... 2092
52.4 Functional description ... 2095

## Chapter 53
## Low-Power Timer (LPTMR)

53.1 Chip-specific LPTMR information ... 2097
53.1.1 LPTMRx prescaler/glitch filter clocking options ... 2097
53.1.2 LPTMRx pulse counter input options ... 2097
53.2 Introduction ... 2098
53.2.1 Features ... 2098
53.2.2 Modes of operation ... 2098
53.3 LPTMR signal descriptions ... 2099
53.3.1 Detailed signal descriptions ... 2099
53.4 Memory map and register definition ... 2100
53.4.1 LPTMR0 register descriptions ... 2100
53.4.2 LPTMR1 register descriptions ... 2105
53.4.3 LPTMR2 register descriptions ... 2110
53.5 Functional description ... 2116
53.5.1 LPTMR power and reset ... 2116
53.5.2 LPTMR clocking ... 2116

53.5.3 LPTMR prescaler/glitch filter....2117
53.5.4 LPTMR compare....2118
53.5.5 LPTMR counter....2118
53.5.6 LPTMR hardware trigger....2119
53.5.7 LPTMR interrupt....2119

## Chapter 54
## Real Time Clock (RTC)

54.1 Chip-specific RTC information....2121
54.1.1 RTC_CLKOUT signal....2121
54.1.2 Number of tamper inputs....2121
54.2 Introduction....2122
54.2.1 Features....2122
54.2.2 Modes of operation....2122
54.2.3 RTC signal descriptions....2123
54.3 Register definition....2124
54.3.1 RTC register descriptions....2124
54.4 Functional description....2148
54.4.1 Power, clocking, and reset....2148
54.4.2 Time counter....2149
54.4.3 Compensation....2150
54.4.4 Time alarm....2151
54.4.5 Update mode....2151
54.4.6 Monotonic counter....2151
54.4.7 Tamper detect....2152
54.4.8 Register lock....2152
54.4.9 Access control....2152
54.4.10 Interrupt....2153

## Chapter 55
## Low Power Inter-Integrated Circuit (LPI2C)

55.1 Chip-specific LPI2C information ..... 2155
55.1.1 LPI2C instantiations ..... 2155
55.2 Introduction ..... 2156
55.2.1 Features ..... 2157
55.2.2 Block Diagram ..... 2159
55.2.3 Modes of operation ..... 2159
55.2.4 Signal Descriptions ..... 2160
55.2.5 Wiring options ..... 2160
55.3 Memory Map and Registers ..... 2162
55.3.1 LPI2C0 register descriptions ..... 2162
55.3.2 LPI2C1 register descriptions ..... 2201
55.3.3 LPI2C2 register descriptions ..... 2240
55.3.4 LPI2C3 register descriptions ..... 2279
55.4 Functional description ..... 2318
55.4.1 Clocking and Resets ..... 2318
55.4.2 Master Mode ..... 2319
55.4.3 Slave Mode ..... 2325
55.4.4 Interrupts and DMA Requests ..... 2327
55.4.5 Peripheral Triggers ..... 2329

## Chapter 56
## Low Power Universal Asynchronous Receiver/Transmitter (LPUART)

56.1 Chip-specific LPUART information ..... 2331
56.1.1 LPUART overview ..... 2331
56.1.2 LPUART Instantiation ..... 2331
56.2 Introduction ..... 2332
56.2.1 Features ..... 2332
56.2.2 Modes of operation ..... 2333
56.2.3 Signal Descriptions ..... 2333
56.2.4 Block diagram ..... 2334

56.3 Register definition....2335
56.3.1 LPUART0 register descriptions....2335
56.3.2 LPUART1 register descriptions....2360
56.3.3 LPUART2 register descriptions....2385
56.3.4 LPUART3 register descriptions....2410
56.4 Functional description....2435
56.4.1 Baud rate generation....2435
56.4.2 Transmitter functional description....2436
56.4.3 Receiver functional description....2439
56.4.4 Additional LPUART functions....2446
56.4.5 Infrared interface....2447
56.4.6 Interrupts and status flags....2448
56.4.7 Peripheral Triggers....2449

## Chapter 57
## Low Power Serial Peripheral Interface (LPSPI)

57.1 Chip-specific LPSPI information....2451
57.1.1 LPSPI instantiations....2451
57.2 Introduction....2451
57.2.1 Features....2453
57.2.2 Block Diagram....2454
57.2.3 Modes of operation....2454
57.2.4 Signal Descriptions....2455
57.2.5 Wiring options....2455
57.3 Memory Map and Registers....2457
57.3.1 LPSPI0 register descriptions....2457
57.3.2 LPSPI1 register descriptions....2479
57.3.3 LPSPI2 register descriptions....2501
57.3.4 LPSPI3 register descriptions....2523
57.4 Functional description....2545

57.4.1 Clocking and resets....2545
57.4.2 Master Mode....2546
57.4.3 Slave Mode....2552
57.4.4 Interrupts and DMA Requests....2554
57.4.5 Peripheral Triggers....2555

## Chapter 58
## Flexible I/O (FlexIO)

58.1 Chip-specific FlexIO information....2557
58.1.1 FlexIO overview....2557
58.1.2 FlexIO Instantiation....2558
58.2 Introduction....2558
58.2.1 Overview....2558
58.2.2 Features....2558
58.2.3 Block Diagram....2559
58.2.4 Modes of operation....2560
58.2.5 FlexIO Signal Descriptions....2560
58.3 Memory Map and Registers....2560
58.3.1 FLEXIO register descriptions....2560
58.4 Functional description....2589
58.4.1 Clocking and Resets....2589
58.4.2 Shifter operation....2590
58.4.3 Timer Operation....2595
58.4.4 Pin operation....2599
58.4.5 Interrupts and DMA Requests....2600
58.4.6 Peripheral Triggers....2601
58.5 Application Information....2601
58.5.1 UART Transmit....2601
58.5.2 UART Receive....2602
58.5.3 SPI Master....2604

58.5.4 SPI Slave....2606
58.5.5 I2C Master....2608
58.5.6 I2S Master....2610
58.5.7 I2S Slave....2611
58.5.8 Camera Interface....2613
58.5.9 Motorola 68K/Intel 8080 Bus Interface....2614
58.5.10 Low Power State Machine....2615

## Chapter 59
## Inter-IC Sound (I2S)

59.1 Chip-specific I2S information....2619
59.1.1 Instantiation information....2619
59.1.2 I2S/SAI operation in low power modes....2619
59.2 Introduction....2620
59.2.1 Features....2620
59.2.2 Block diagram....2621
59.2.3 Modes of operation....2621
59.3 External signals....2622
59.4 Memory map and register definition....2623
59.4.1 I2S register descriptions....2623
59.5 Functional description....2652
59.5.1 SAI clocking....2653
59.5.2 SAI resets....2654
59.5.3 Synchronous modes....2655
59.5.4 Frame sync configuration....2656
59.5.5 Data FIFO....2657
59.5.6 Word mask register....2661
59.5.7 Interrupts and DMA requests....2661

## Chapter 60
## Smart Card Interface Module (EMV SIM)

60.1 Chip-specific EMV SIM information ... 2665
60.1.1 EMV_SIM overview ... 2665
60.1.2 EMV_SIM Instantiation ... 2665
60.2 Introduction ... 2665
60.2.1 Features ... 2665
60.3 Block Diagram ... 2666
60.4 Design Overview ... 2667
60.5 Signal Description ... 2669
60.6 Memory Map and Registers ... 2669
60.6.1 EMVSIM register descriptions ... 2669
60.7 Functional Description ... 2697
60.7.1 Initialization ... 2697
60.7.2 Smart Card Interface and Control ... 2699
60.7.3 EMV SIM Receiver ... 2702
60.7.4 EMV SIM Transmitter ... 2706
60.7.5 LRC and CRC ... 2708
60.7.6 Message Handling ... 2710
60.7.7 Protocol Timers ... 2712
60.7.8 Answer To Reset (ATR) Detection ... 2715

## Chapter 61
## Universal Serial Bus Full Speed Controller (USBFS)

61.1 Chip-specific USB information ... 2721
61.1.1 USB wakeup ... 2721
61.1.2 LPUART over USB capability ... 2721
61.1.3 USB Power Distribution ... 2722
61.1.4 USB controller configuration ... 2724
61.1.5 Keep Alive Mode and IRC48M operation ... 2724
61.2 Introduction ... 2726
61.2.1 References ... 2726

61.2.2 USB—Overview ... 2727
61.2.3 USBFS Features ... 2728
61.3 Functional description ... 2728
61.3.1 Data Structures ... 2728
61.3.2 On-chip transceiver required external components ... 2728
61.4 Programmers interface ... 2729
61.4.1 Buffer Descriptor Table ... 2729
61.4.2 USB data transfers—Receive (Rx) and Transmit (Tx) ... 2731
61.4.3 Addressing BDT entries ... 2731
61.4.4 Buffer Descriptors (BDs) ... 2732
61.4.5 USB transaction ... 2734
61.5 Memory map/Register definitions ... 2736
61.5.1 USBFS memory mapping ... 2736
61.5.2 USB register descriptions ... 2736
61.6 Keep Alive mode ... 2776

## Chapter 62
## USB Voltage Regulator (USBVREG)

62.1 Chip-specific USBVREG information ... 2779
62.1.1 USB Voltage Regulator Configuration ... 2779
62.2 Introduction ... 2779
62.2.1 Features ... 2779
62.3 Memory Map and Registers ... 2780
62.3.1 USBVREG register descriptions ... 2780

## Chapter 63
## Ultra Secured Digital Host Controller (uSDHC)

63.1 Overview ... 2783
63.1.1 Features ... 2786
63.1.2 Modes and Operations ... 2787
63.2 External Signals ... 2787

63.2.1 Signals Overview ... 2788
63.3 Clocks ... 2789
63.4 Functional Description ... 2789
63.4.1 Data Buffer ... 2789
63.4.2 DMA AHB Interface ... 2794
63.4.3 Register Bank with IP Bus Interface ... 2802
63.4.4 Clock & Reset Manager ... 2805
63.4.5 Clock Generator ... 2806
63.4.6 SDIO Card Interrupt ... 2806
63.4.7 Card Insertion and Removal Detection ... 2808
63.4.8 Power Management and Wake Up Events ... 2809
63.4.9 MMC fast boot ... 2810
63.5 Initialization/Application of uSDHC ... 2812
63.5.1 Command Send & Response Receive Basic Operation ... 2812
63.5.2 Card Identification Mode ... 2813
63.5.3 Card Access ... 2818
63.5.4 Switch Function ... 2830
63.5.5 ADMA Operation ... 2832
63.5.6 Fast Boot Operation ... 2834
63.6 Commands for MMC/SD/SDIO ... 2837
63.7 Software Restrictions ... 2842
63.7.1 Initialization Active ... 2842
63.7.2 Software Polling Procedure ... 2843
63.7.3 Suspend Operation ... 2843
63.7.4 Data Length Setting ... 2843
63.7.5 (A)DMA Address Setting ... 2843
63.7.6 Data Port Access ... 2844
63.7.7 Change Clock Frequency ... 2844
63.7.8 Multi-block Read ... 2844

63.8 uSDHC Memory Map/Register Definition....2844
63.8.1 uSDHC register descriptions....2845

## Chapter 64
## 2.4 GHz Multi-Protocol Radio

64.1 Introduction....2903
64.2 Radio System Integration Module....2904
64.2.1 Introduction....2904
64.2.2 Radio Power....2907
64.2.3 Radio Clocks....2914
64.2.4 Radio Stop Modes....2920
64.2.5 Radio Deep Sleep....2925
64.2.6 Wi-Fi Coexistence....2931
64.2.7 RF Early Warning....2934
64.2.8 Radio Interrupts....2935
64.2.9 Radio Configuration....2938
64.2.10 Radio Resets....2939
64.2.11 Radio Gasket Bypass....2940
64.2.12 Radio Version ID....2941
64.2.13 Memory Map and Register Definition....2941
64.3 Transceiver Digital....2968
64.3.1 Transmitter Digital Module....2968
64.3.2 PLL Frequency Synthesizer....2990
64.3.3 Receiver Digital Module....3039
64.3.4 Bit Streaming Mode....3190
64.3.5 Transceiver Sequence Manager....3196
64.3.6 Transceiver DMA and Packet RAM Debug Modes....3340
64.3.7 2.4 GHz PHY....3364
64.3.8 Wake-On-Radio....3373
64.3.9 Cyclic Redundancy Check and Data Whitening Module (CRCW)....3448

64.3.10 Other Transceiver Memory Map and Register Definition..... 3457
64.4 Transceiver Analog..... 3510
64.4.1 Introduction..... 3510
64.4.2 Functional description..... 3512
64.5 Link Layer..... 3513
64.5.1 BLE Link Layer..... 3513
64.5.2 802.15.4 Link Layer Controller..... 3642
64.5.3 Generic_FSK Link Layer Controller..... 3840

# Chapter 1
# About This Manual

## 1.1 Audience

This reference manual is intended for system software and hardware developers and applications programmers who want to develop products with this device. It assumes that the reader understands operating systems, microprocessor system design, and basic principles of software and hardware.

## 1.2 Organization

This manual has two main sets of chapters.

1. Chapters in the first set contain information that applies to all components on the chip.
2. Chapters in the second set are organized into functional groupings that detail particular areas of functionality.
   - Examples of these groupings are clocking, timers, and communication interfaces.
   - Each grouping includes chapters that provide a technical description of individual modules.

## 1.3 Module descriptions

Each module chapter has two main parts:

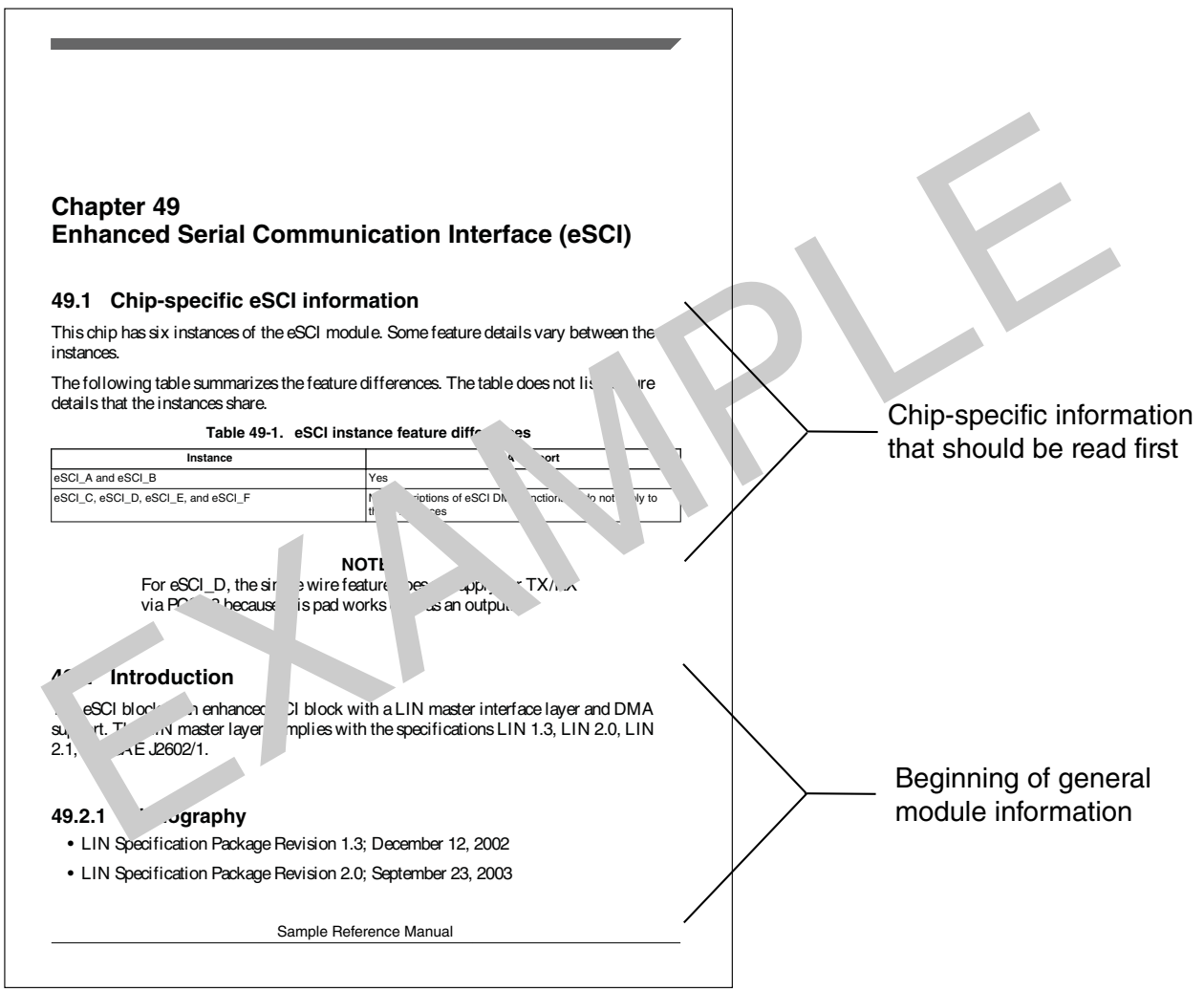
- **Chip-specific:** The first section, *Chip-specific [module name] information*, includes the number of module instances on the chip and possible implementation differences between the module instances, such as differences in FIFO depths or the number of

channels supported. It may also include functional connections between the module instances and other modules. Read this section *first* because its content is crucial to understanding the information in other sections of the chapter.

- **General:** The subsequent sections provide general information about the module, including its signals, registers, and functional description.

**NOTE**

If there is a conflict between the chip-specific module information (first section) and the general module information (subsequent sections), the chip-specific information supersedes the general information.



Chapter 49
Enhanced Serial Communication Interface (eSCI)

49.1 Chip-specific eSCI information

This chip has six instances of the eSCI module. Some feature details vary between the instances.

The following table summarizes the feature differences. The table does not list feature details that the instances share.

Table 49-1. eSCI instance feature differences

| Instance | [illegible] Support |
|---|---|
| eSCI_A and eSCI_B | Yes |
| eSCI_C, eSCI_D, eSCI_E, and eSCI_F | [illegible] descriptions of eSCI DMA functions do not apply to these instances |

NOTE

For eSCI_D, the single wire feature does support TX/RX via PC[illegible] because this pad works as an output

49.[illegible] Introduction

The eSCI block is an enhanced SCI block with a LIN master interface layer and DMA support. The LIN master layer complies with the specifications LIN 1.3, LIN 2.0, LIN 2.1, and SAE J2602/1.

49.2.1 [illegible]ography

- LIN Specification Package Revision 1.3; December 12, 2002
- LIN Specification Package Revision 2.0; September 23, 2003

Sample Reference Manual

Chip-specific information that should be read first

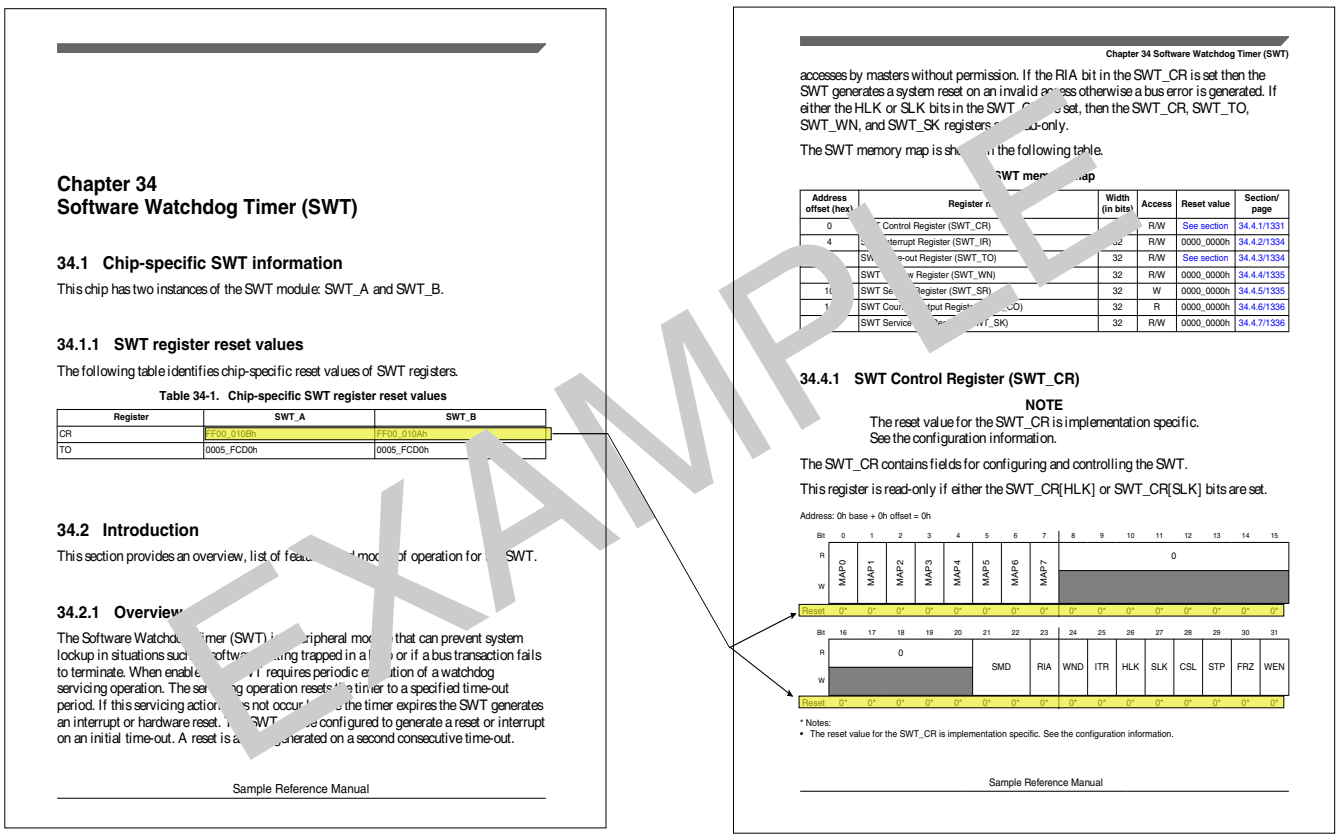
Beginning of general module information

EXAMPLE

**Figure 1-1. Example: chapter chip-specific information and general module information**

## 1.3.1 Example: chip-specific information that supersedes content in the same chapter

The example below shows chip-specific information that supersedes general module information presented later in the chapter. In this case, the chip-specific register reset values supersede the reset values that appear in the register diagram.



Chapter 34
Software Watchdog Timer (SWT)

34.1 Chip-specific SWT information

This chip has two instances of the SWT module: SWT_A and SWT_B.

34.1.1 SWT register reset values

The following table identifies chip-specific reset values of SWT registers.

Table 34-1. Chip-specific SWT register reset values

| Register | SWT_A | SWT_B |
|---|---|---|
| CR | FF00_010Bh | FF00_010Ah |
| TO | 0005_FCD0h | 0005_FCD0h |

34.2 Introduction

This section provides an overview, list of feat... mod... of operation for ... SWT.

34.2.1 Overview

The Software Watchd... imer (SWT) i... ripheral mo... that can prevent system lockup in situations suc... softw... ing trapped in a l... or if a bus transaction fails to terminate. When enable... T requires periodic e... tion of a watchdog servicing operation. The se... g operation resets ... timer to a specified time-out period. If this servicing actio... s not occur ... the timer expires the SWT generates an interrupt or hardware reset. ... SWT ... e configured to generate a reset or interrupt on an initial time-out. A reset is a... enerated on a second consecutive time-out.

Sample Reference Manual

Chapter 34 Software Watchdog Timer (SWT)

accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid a... ss otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_C... set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers ... d-only.

The SWT memory map is sh... n the following table.

SWT mem... ap

| Address offset (hex) | Register n... | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 0 | ... Control Register (SWT_CR) | | R/W | See section | 34.4.1/1331 |
| 4 | ... terrupt Register (SWT_IR) | ...2 | R/W | 0000_0000h | 34.4.2/1334 |
| | SW... e-out Register (SWT_TO) | 32 | R/W | See section | 34.4.3/1334 |
| | SWT ... w Register (SWT_WN) | 32 | R/W | 0000_0000h | 34.4.4/1335 |
| 10 | SWT Se... Register (SWT_SR) | 32 | W | 0000_0000h | 34.4.5/1335 |
| 1... | SWT Coun... tput Regis... CO) | 32 | R | 0000_0000h | 34.4.6/1336 |
| | SWT Service ... T_SK) | 32 | R/W | 0000_0000h | 34.4.7/1336 |

34.4.1 SWT Control Register (SWT_CR)

NOTE
The reset value for the SWT_CR is implementation specific.
See the configuration information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAP0 | MAP1 | MAP2 | MAP3 | MAP4 | MAP5 | MAP6 | MAP7 | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

| Bit | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | SMD | RIA | WND | ITR | HLK | SLK | CSL | STP | FRZ | WEN | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* | 0* |

* Notes:
- The reset value for the SWT_CR is implementation specific. See the configuration information.

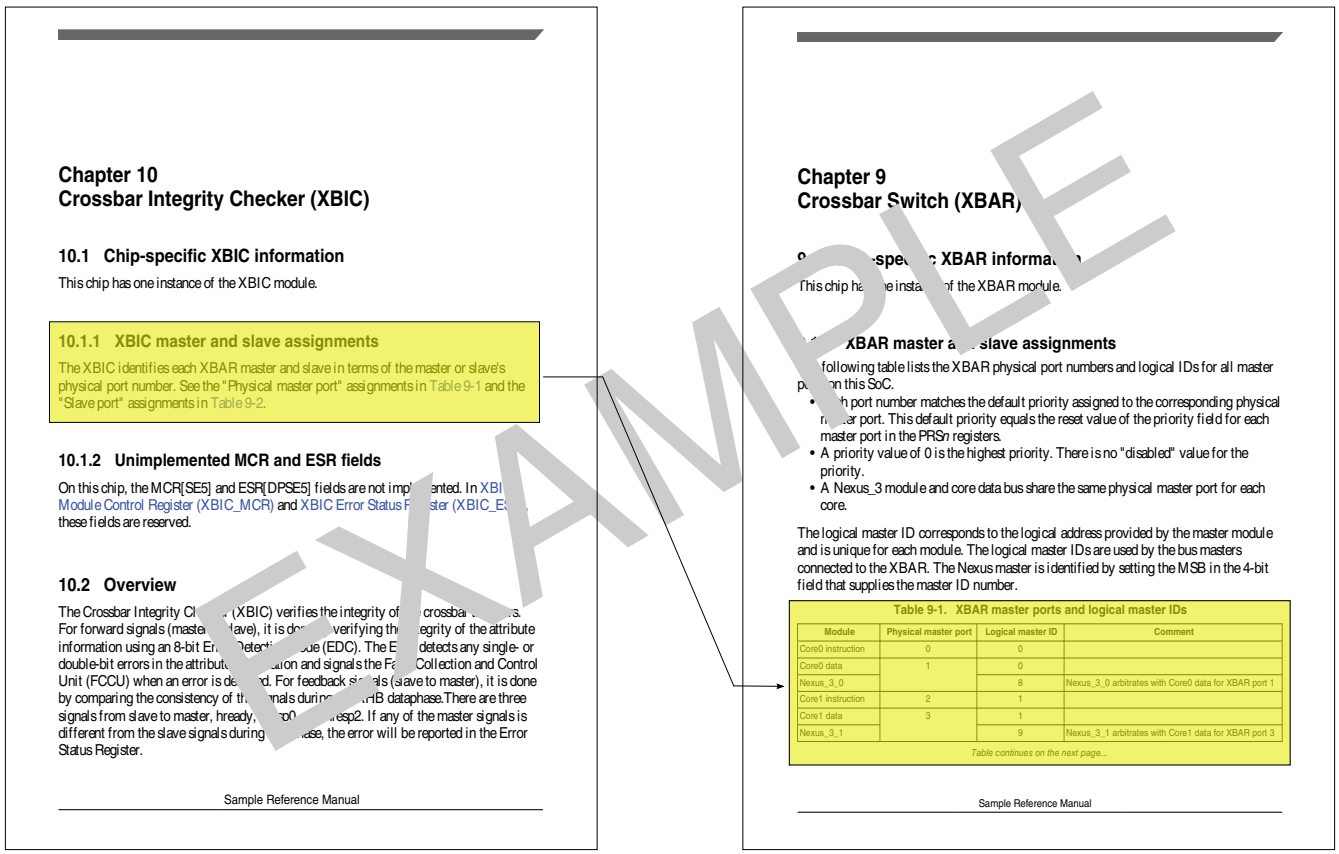
Sample Reference Manual

EXAMPLE

**Figure 1-2. Example: chip-specific information that supersedes content in the same chapter**

## 1.3.2 Example: chip-specific information that refers to a different chapter

The chip-specific information below refers to another chapter's chip-specific information. In this case, read both sets of chip-specific information before reading further in the chapter.



Chapter 10
Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

The XBIC identifies each XBAR master and slave in terms of the master or slave's physical port number. See the "Physical master port" assignments in Table 9-1 and the "Slave port" assignments in Table 9-2.

10.1.2 Unimplemented MCR and ESR fields

On this chip, the MCR[SE5] and ESR[DPSE5] fields are not implemented. In XBI Module Control Register (XBIC_MCR) and XBIC Error Status Register (XBIC_E these fields are reserved.

10.2 Overview

The Crossbar Integrity Ch (XBIC) verifies the integrity of crossba s. For forward signals (maste lave), it is do verifying th egrity of the attribute information using an 8-bit Er Detecti ue (EDC). The E detects any single- or double-bit errors in the attribut tion and signals the Fa Collection and Control Unit (FCCU) when an error is de d. For feedback si ls (slave to master), it is done by comparing the consistency of th nals duri hB dataphase.There are three signals from slave to master, hready, so0 esp2. If any of the master signals is different from the slave signals during ase, the error will be reported in the Error Status Register.

Sample Reference Manual

Chapter 9
Crossbar Switch (XBAR)

9 -spe c XBAR informa n

This chip ha e insta of the XBAR module.

XBAR master a slave assignments

following table lists the XBAR physical port numbers and logical IDs for all master p on this SoC.

- h port number matches the default priority assigned to the corresponding physical r er port. This default priority equals the reset value of the priority field for each master port in the PRSn registers.
- A priority value of 0 is the highest priority. There is no "disabled" value for the priority.
- A Nexus_3 module and core data bus share the same physical master port for each core.

The logical master ID corresponds to the logical address provided by the master module and is unique for each module. The logical master IDs are used by the bus masters connected to the XBAR. The Nexus master is identified by setting the MSB in the 4-bit field that supplies the master ID number.

Table 9-1. XBAR master ports and logical master IDs

| Module | Physical master port | Logical master ID | Comment |
|---|---|---|---|
| Core0 instruction | 0 | 0 | |
| Core0 data | 1 | 0 | |
| Nexus_3_0 | | 8 | Nexus_3_0 arbitrates with Core0 data for XBAR port 1 |
| Core1 instruction | 2 | 1 | |
| Core1 data | 3 | 1 | |
| Nexus_3_1 | | 9 | Nexus_3_1 arbitrates with Core1 data for XBAR port 3 |

Table continues on the next page...

Sample Reference Manual

EXAMPLE

**Figure 1-3. Example: chip-specific information that refers to a different chapter**

## 1.4 Register descriptions

Module chapters present register information in:

- Memory maps including:
  - Addresses
  - The name and acronym/abbreviation of each register
  - The width of each register (in bits)
  - Each register's reset value
  - The page number on which each register is described
- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in the following figure.

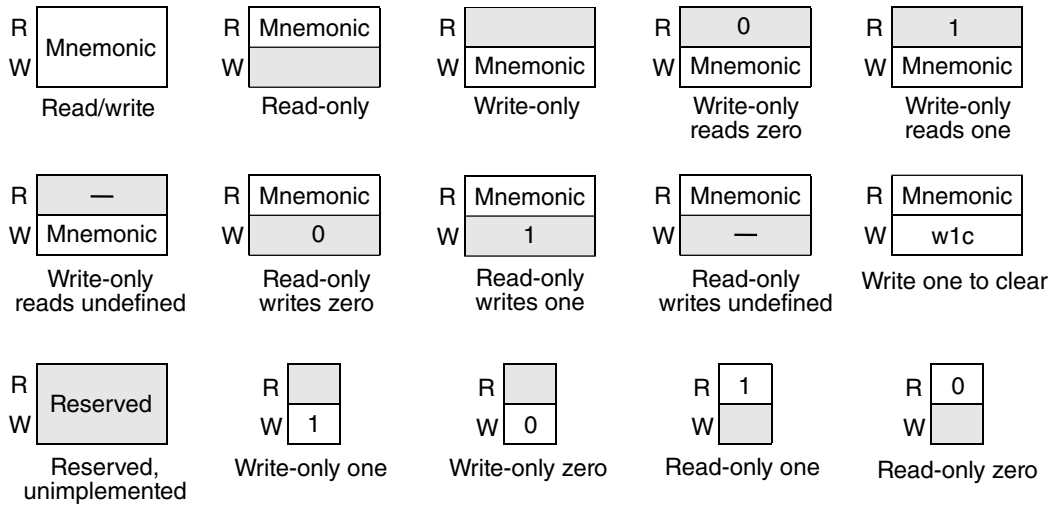



**Figure 1-4. Register figure conventions**

# 1.5 Conventions

## 1.5.1 Numbering systems

The following suffixes identify different numbering systems:

| This suffix | Identifies a |
|---|---|
| b | Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*. |
| d | Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix. |
| h | Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*. |

## 1.5.2 Typographic notation

The following typographic notation is used throughout this document:

| Example | Description |
|---|---|
| *placeholder*, x | Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers. |
| `code` | Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type |

*Table continues on the next page...*

| Example | Description |
|---|---|
| | is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR. |
| SR[SCM] | A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR). |
| REVNO[6:4], XAD[7:0] | Numbers in brackets and separated by a colon represent either:<br>• A subset of a register's named field<br>For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register.<br>• A continuous range of individual signals of a bus<br>For example, XAD[7:0] refers to signals 7–0 of the XAD bus. |

## 1.5.3 Special terms

The following terms have special meanings:

| Term | Meaning |
|---|---|
| asserted | Refers to the state of a signal as follows:<br>• An active-high signal is asserted when high (1).<br>• An active-low signal is asserted when low (0). |
| deasserted | Refers to the state of a signal as follows:<br>• An active-high signal is deasserted when low (0).<br>• An active-low signal is deasserted when high (1).<br><br>In some cases, deasserted signals are described as *negated*. |
| reserved | Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior.<br>• Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field.<br>• Consider undefined locations in memory to be reserved. |
| w1c | Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared." |

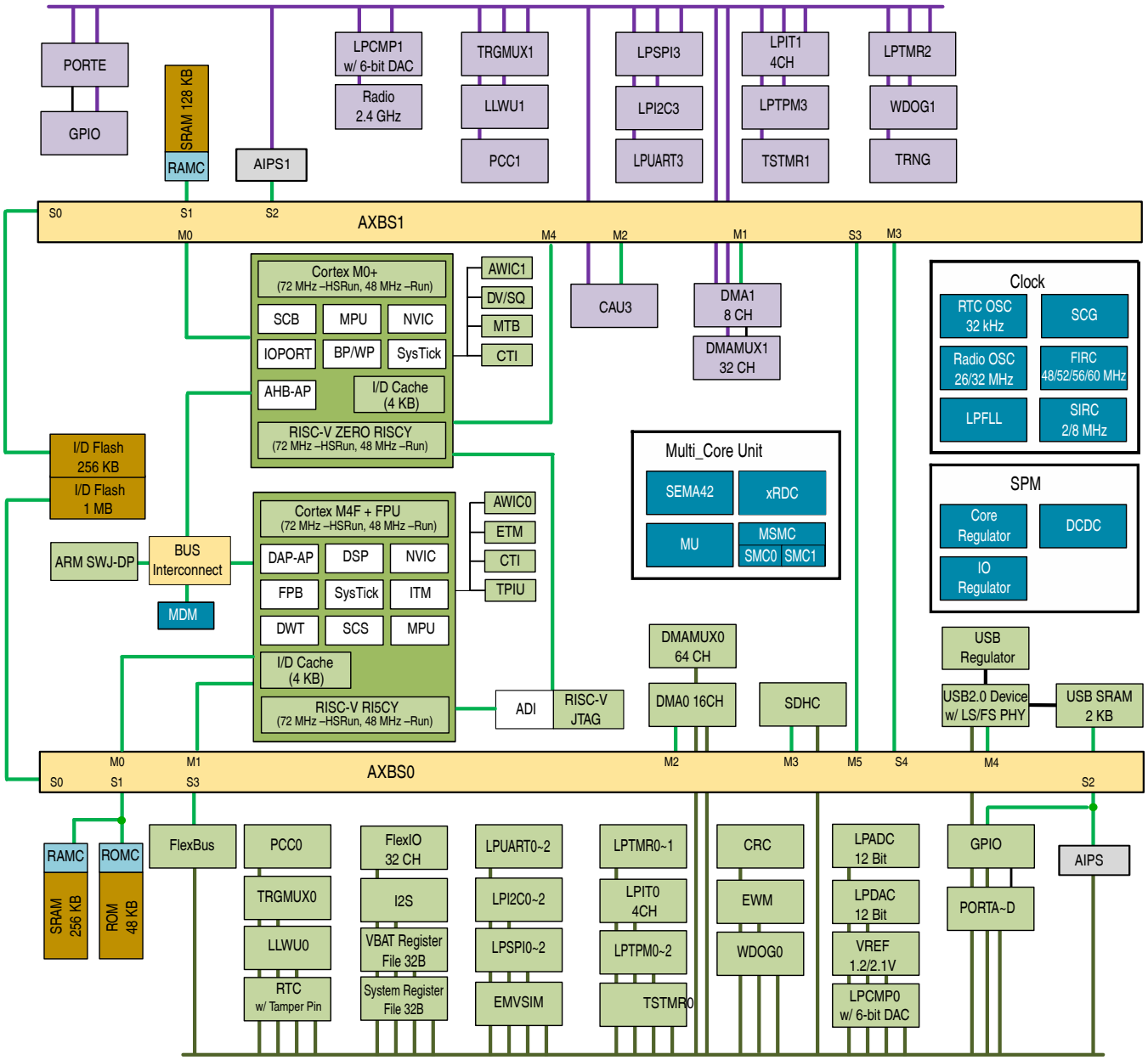
# Chapter 2
# Introduction

## 2.1 Overview

This chapter provides high-level descriptions of the modules available on the devices covered by this document.

## 2.2 Block diagram

The figure below shows a top-level block diagram.




Note: When a core needs access the other AXBS domain resources, the corresponding MUx_CCR[CLKE] bit must be set. AXBS keeps active even if another core enters low power mode (and CPO).

**Figure 2-1. Block diagram**

## 2.3 Module functional categories

The modules on this device are grouped into functional categories in the following table.

**Table 2-1. Module functional categories**

| Module category | Description |
|---|---|
| Processor cores | The available core modules are:<br>• ARM® Cortex®-M0+ core |

*Table continues on the next page...*

## Table 2-1. Module functional categories (continued)

| Module category | Description |
|---|---|
| | • Debug Watchpoint and Trace (DWT)<br>• ARM® Memory Protection Unit (MPU-ARM)<br>• Micro Trace Buffer (MTB)<br>• Nested Vectored Interrupt Controller (NVIC)<br>• ARM® Cortex®-M4F core<br>  • Digital Signal Processing (DSP) features<br>  • Debug Watchpoint and Trace (DWT)<br>  • Flash Patch Breakpoint (FPB)<br>  • Single precision Floating Point Unit (sFPU)<br>  • Instruction Trace Module (ITM)<br>  • ARM® Memory Protection Unit (MPU-ARM)<br>  • Nested Vectored Interrupt Controller (NVIC)<br>• RISC-V RI5CY core<br>• RISC-V ZERORISCY core |
| Platform | The available platform modules are:<br>• Peripheral Bridges (AIPS)<br>• Crossbar Switch (AXBS)<br>• Asynchronous Wake-up Interrupt Controller (AWIC)<br>• Boundary Scan<br>• Cache memory - Instruction/Data (CACHE-I/D)<br>• Cross Trigger Interface (CTI)<br>• Debug Access Port (DAP)<br>• Miscellaneous Control Module for CM0+ (MCM)<br>• Miscellaneous Control Module for CM4F (MCM)<br>• Miscellaneous System Control Module (MSCM)<br>• Extended Resource Domain Controller (XRDC)<br>• Semaphore Module (SEMA42)<br>• Memory Mapped Division and Square Root (MMDVSQ)<br>• Serial Wire Debug - Version 2.0 w/ Multi-Drop (SWD)<br>• Trace Port Interface Unit (TPIU) |
| Debug | The available debug modules are:<br>• Micro Trace Buffer (MTB)<br>• JTAG Controller (JTAGC)<br>• Advanced Debug Interface(ADI) |
| Memories | The available memory and memory interface modules are:<br>• Flash Memory Module (FTFE)<br>• Read Only Memory - Boot (BootROM)<br>• Register File - System (SYS-RegFile)<br>• Register File - VBAT (VBAT-RegFile)<br>• Static Random Access Memory (SRAM)<br>• Flash Memory Controller (FMC)<br>• Flash Security<br>• External Bus Interface (FlexBus)<br>• AHB Low Power Cache Controller (AHB_LPCAC)<br>• AHB Low Power Local Memory Controller (AHB_LPLMEM) |
| Pin control | The available pin control modules are:<br>• Port Control and Interrupts (PORT)<br>• General Purpose Input/Output (GPIO) |
| Clocks | The available clock modules are: |

*Table continues on the next page...*

## Table 2-1. Module functional categories (continued)

| Module category | Description |
|---|---|
| | • Low-Power Frequency Locked Loop (LPFLL)<br>• Fast Internal Reference Clock (FIRC)<br>• Slow Internal Reference Clock (SIRC)<br>• Peripheral Clock Controller (PCC)<br>• Radio Frequency Oscillator (RFOSC)<br>• Real Time Clock Oscillator (RTCOSC)<br>• System Clock Generator (SCG) |
| Power | The available power modules are:<br>• Multi System Mode Controller (MSMC)<br>• System Power Manager (SPM)<br>• Low-leakage Wakeup Unit (LLWU) |
| System | The available system modules are:<br>• System Integration Module (SIM)<br>• Interrupt Multiplexer (INTMUX)<br>• Trigger MUX (TRGMUX)<br>• Enhanced Direct Memory Access (eDMA)<br>• Direct Memory Access Multiplexer (DMAMUX)<br>• Messaging Unit (MU) |
| Security and Integrity | The available security and integrity modules are:<br>• Watchdog timer (WDOG)<br>• External Watchdog Monitor (EWM)<br>• Cyclic Redundancy Check (CRC)<br>• Standalone True Random Number Generator (SA-TRNG)<br>• Cryptographic Acceleration Unit (CAU3)<br>  • Advanced Encryption Standard (AES)<br>  • Data Encryption Standard (DES/3DES)<br>  • Elliptical Curve Cryptography (ECC)<br>  • Rivest-Shamir-Adleman cryptosystem (RSA)<br>  • Secure Hash Algorithm (SHA)<br>• Tamper detect |
| Analog | The available analog modules are:<br>• Low Power Analog-to-Digital Converter (LPADC) - 12-bit<br>• Low-Power Comparator (LPCMP)<br>• 12-bit Low Power Digital-to-Analog Converter (LPDAC)<br>• Voltage Reference (VREF) |
| Timers | The available timer modules are:<br>• Low-Power Periodic Interrupt Timer (LPIT)<br>• Timer/ PWM Module (TPM)<br>• Time Stamp Timer Module (TSTMR)<br>• Low-Power Timer (LPTMR)<br>• Real Time Clock (RTC) |
| Communication Interfaces | The available communication modules are:<br>• Low-Power Inter-Integrated Circuit (LPI2C)<br>• Low-Power Universal Asynchronous Receiver/Transmitter (LPUART)<br>• Low-Power Serial Peripheral Interface (LPSPI)<br>• Flexible Input/Output (FlexIO)<br>• Inter-IC Sound (I2S)<br>• Smart Card Interface Module (EMV SIM)<br>• Universal Serial Bus (USB) Full Speed controller :<br>  • Device<br>  • Built-in FS transceiver<br>  • Support for crystal-less operation |

*Table continues on the next page...*

**Table 2-1. Module functional categories (continued)**

| Module category | Description |
|---|---|
| | • USB Voltage Regulator (USBVREG)<br>• Ultra Secured Digital Host Controller (uSDHC) |
| Radio | The available radio modules are:<br>• Radio System Integration Module<br>• BLE Link layer<br>• Generic FSK Link Layer Controller<br>• 802.15.4 Link Layer Controller |

## 2.4 Ordering information

The following table summarizes the part numbers of the devices covered by this document.

**Table 2-2. Ordering information**

| Product | | Memory | | Radio | | | Package | |
|---|---|---|---|---|---|---|---|---|
| Part Number | Marking | Flash (MB) | SRAM (KB) | BLE | GENFSK | 802.15.4 | Pin Count | Package |
| RV32M1 | RV32M1 | 1.25 | 384 | Yes | Yes | Yes | 176 | VFBGA |

# Chapter 3
# Core Overview

## 3.1 Core modules

This device has one ARM Cortex M0+ core, one ARM Cortex M4F core, one RISC-V RI5CY core and one RISC-V ZERORISCY core.

### 3.1.1 ARM Cortex-M0+ core

#### 3.1.1.1 Buses, interconnects, and interfaces

The ARM Cortex-M0+ core has two bus interfaces:

- Single 32-bit AMBA-3 AHB-Lite system interface that provides connections to peripherals and all system memory, which includes flash memory and RAM
- Single 32-bit I/O port bus interfacing to the GPIO with 1-cycle loads and stores

#### 3.1.1.2 System tick timer

The CLKSOURCE field in SysTick Control and Status register selects either the core clock (when CLKSOURCE = 1) or a divide-by-16 of the core clock (when CLKSOURCE = 0) and the divide-by-16 clock can be gated by SIM_MISC2[systick_clk_en]. Because the timing reference is a variable frequency, the TENMS field in the SysTick Calibration Value Register is always 0.

#### 3.1.1.3 Debug facilities

This device supports standard ARM 2-pin SWD debug port and JTAG.

#### 3.1.1.4 CM0+ core caches

CM0+ domain includes one 4 KB of combined data and instructions cache to minimize the performance impact on memory access latencies for I/D bus, and minimize the power consumption because of flash access.

Features of each cache are:

- 2-way, 8 sets, and 64 lines (pages)
- Each line is 1 word of 32-bit
- Entire cache can be flushed at once

##### 3.1.1.4.1 Cache operation in low power

The caches are powered down in VLLS0/1 modes. Under LLS and VLLS2/3 modes, caches are optionally powered down and controlled by SMC1_SRAMDSR[31] bit in MSMC module. In Run mode, the SMC1_SRAMLPR[31] bit can be used to put cache into power retention state. The caches must be disabled and then flushed before being powered down. This affects the time for the system to enter new mode but not the wakeup time.

##### 3.1.1.4.2 Control

The CM0+ cache controller, LPLMEM, supports write-through mode. The memory space below 0x2000_0000 is cacheable, and caches can be enabled by MCM1_CPCR2[DCC] under Run mode.

#### 3.1.1.5 Core privilege levels

The core on this device is implemented with both privileged and unprivileged levels. The ARM documentation uses different terms from this document to distinguish between privilege levels.

| If you see this term... | it also means this term... |
|---|---|
| Privileged | Supervisor |
| Unprivileged or user | User |

### 3.1.2 ARM Cortex-M4F Core

### 3.1.2.1 Buses, interconnects, and interfaces

The ARM Cortex-M4 core has four buses as described in the following table.

| Bus name | Description |
|---|---|
| Instruction code (ICODE) bus | The ICODE and DCODE buses are muxed. This muxed bus is called the CODE bus and is connected to the crossbar switch via a single master port. In addition, the CODE bus is also tightly coupled to the lower half of the system RAM (SRAM_L). |
| Data code (DCODE) bus | |
| System bus | The system bus is connected to a separate master port on the crossbar. In addition, the system bus is tightly coupled to the upper half system RAM (SRAM_U). |
| Private peripheral (PPB) bus | The PPB provides access to these modules:<br>• ARM modules such as the NVIC, ETM, ITM, DWT, FPB, and ROM table |

### 3.1.2.2 System Tick Timer

The System Tick Timer's clock source is always the core clock, FCLK. This results in the following:

- The CLKSOURCE bit in SysTick Control and Status register is always set to select the core clock.
- Because the timing reference (FCLK) is a variable frequency, the TENMS bit in the SysTick Calibration Value Register is always zero.
- The NOREF bit in SysTick Calibration Value Register is always set, implying that FCLK is the only available source of reference timing.

### 3.1.2.3 Debug facilities

This device has extensive debug capabilities including run control and tracing capabilities. The standard ARM debug port supports JTAG and multi-drop SWD interfaces.

### 3.1.2.4 CM4F core caches

CM4F domain includes one 4 KB of combined data and instructions cache to minimize the performance impact on memory access latencies for I/D bus, and minimize the power consumption because of flash access.

Features of each cache are:

- 2-way, 8 sets, and 16 lines (pages)

- Each line is 4 words of 128-bit
- Entire cache can be flushed at once

#### 3.1.2.4.1 Low power

The caches are powered down in VLLS0/1 modes. Under LLS and VLLS2/3 modes, caches are optional powered down and controlled by SMC0_SRAMDSR[31] bit in MSMC module. In Run mode, SMC0_SRAMLPR[31] bit can be used to put cache into power retention state. The caches must be disabled and then flushed before being powered down. This affects the time for the system to enter new mode but not the wakeup time.

#### 3.1.2.4.2 Control

The CM4F cache controller, LPCAC, supports write-through mode. The memory space below 0x2000_0000 is cacheable, and caches can be enabled by MCM0_CPCR2[DCC] under Run mode.

### 3.1.2.5 Core privilege levels

The ARM documentation uses different terms than this document to distinguish between privilege levels.

| If you see this term... | it also means this term... |
|---|---|
| Privileged | Supervisor |
| Unprivileged or user | User |

## 3.1.3 RISC-V RI5CY core

### 3.1.3.1 Introduction

RI5CY is a 4-stage in-order 32b RISC-V processor core. The ISA of RI5CY was extended to support multiple additional instructions including hardware loops, post-increment load and store instructions and additional ALU instructions that are not part of the standard RISC-V ISA.

The following figure shows a block diagram of the core.

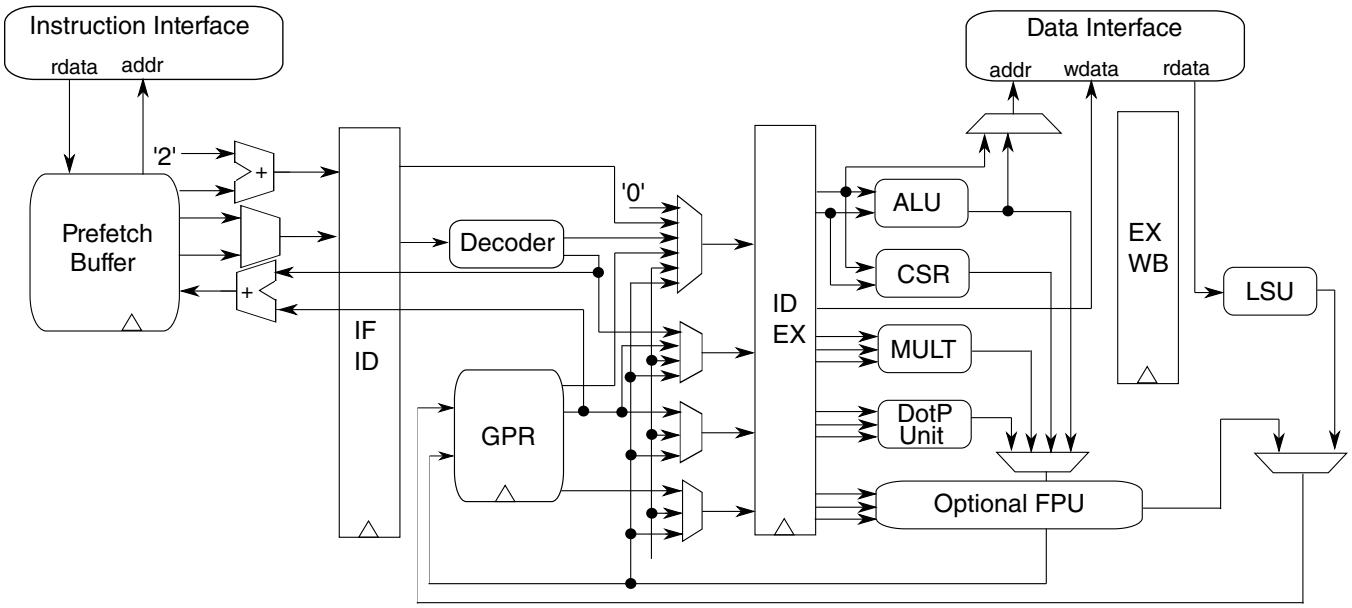



**Figure 3-1. Block diagram**

RI5CY supports the following instructions:

- Full support for RV32I Base Integer Instruction Set
- Full support for RV32C Standard Extension for Compressed Instructions
- Full support for RV32M Integer Multiplication and Division Instruction Set Extension
- Specific extensions
  - Post-Incrementing load and stores
  - Multiply-Accumulate extensions
  - ALU extensions
  - Hardware Loops

### 3.1.3.2 Post-Incrementing Load and Store Instructions

Post-incrementing load and store instructions perform a load/store operation from/to the data memory while at the same time increasing the base address by the specified offset. For the memory access, the base address without offset is used.

Post-incrementing load and stores reduce the number of required instructions to execute code with regular data access patterns, which can typically be found in loops. These post-incrementing load/store instructions allow the address increment to be embedded in the memory access instructions and get rid of separate instructions to handle pointers. Coupled with hardware loop extension, this instructions allow to reduce the loop overhead significantly.

### 3.1.3.3 Multiply-Accumulate

RI5CY uses a single-cycle 32-bit x 32-bit multiplier with a 32-bit result. All instructions of the RISC-V M instruction set extension are supported.

The multiplications with upper-word result (MSP of 32-bit x 32-bit multiplication), take 4 cycles to compute. The division and remainder instructions take between 2 and 32 cycles. The number of cycles depends on the operand values.

Additionally, RI5CY supports non-standard extensions for multiply-accumulate and half-word multiplications with an optional post-multiplication shift.

### 3.1.3.4 ALU extensions

RI5CY supports advanced ALU operations that allow to perform multiple instructions that are specified in the base instruction set in one single instruction and thus increases efficiency of the core. For example, those instructions include zero-/sign-extension instructions for 8-bit and 16-bit operands, simple bit manipulation/counting instructions and min/max/avg instructions.

The ALU does also support saturating, clipping, and normalizing instructions which make fixed-point arithmetic more efficient.

### 3.1.3.5 Hardware loop extensions

To increase the efficiency of small loops, RI5CY supports hardware loops. Hardware loops make it possible to execute a piece of code multiple times, without the overhead of branches or updating a counter. Hardware loops involve zero stall cycles for jumping to the first instruction of a loop.

A hardware loop is defined by its start address (pointing to the first instruction in the loop), its end address (pointing to the instruction that will be executed last in the loop) and a counter that is decremented every time the loop body is executed. RI5CY contains two hardware loop register sets to support nested hardware loops, each of them can store these three values in separate flip flops which are mapped in the CSR address space. If the end address of the two hardware loops is identical, loop 0 has higher priority and only the loop counter for hardware loop 0 is decremented. As soon as the counter of loop 0 reaches 1 at an end address, meaning it is decremented to 0 now, loop 1 is activated too. In this case, both counters will be decremented and the core jumps to the start of loop 1.

In order to use hardware loops, the compiler needs to setup the loop beforehand with the following instructions. Note that the minimum loop size is two instructions.

For debugging and context switches, the hardware loop registers are mapped into the CSR address space and thus it is possible to read and write them via csrr and csrw instructions. Since hardware loop registers could be overwritten when processing interrupts, the registers have to be saved in the interrupt routine together with the general purpose registers.

### 3.1.3.6 RI5CY Memory map and register descriptions

Performance Counters in RI5CY are placed inside the Control and Status Registers and can be accessed with csrr and csrw instructions.

**RI5CY memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 0 | User Status (RI5CY_USTATUS) | 32 | R/W | 0000_0000h | 3.1.3.6.1/ 66 |
| 5 | User Trap-Vector Base Address (RI5CY_UTVEC) | 32 | R/W | 0000_0000h | 3.1.3.6.2/ 67 |
| 41 | User Exception PC (RI5CY_UEPC) | 32 | R/W | 0000_0000h | 3.1.3.6.3/ 67 |
| 42 | User Cause (RI5CY_UCAUSE) | 32 | R/W | 0000_0000h | 3.1.3.6.4/ 68 |
| 300 | Machine Status (RI5CY_MSTATUS) | 32 | R/W | 0000_0000h | 3.1.3.6.5/ 69 |
| 305 | Machine Trap-Vector Base Address (RI5CY_MTVEC) | 32 | R/W | 0000_0000h | 3.1.3.6.6/ 70 |
| 341 | Machine Exception PC (RI5CY_MEPC) | 32 | R/W | 0000_0000h | 3.1.3.6.7/ 70 |
| 342 | Machine Cause (RI5CY_MCAUSE) | 32 | R/W | 0000_0000h | 3.1.3.6.8/ 71 |
| 780 | Performance Counter Counter Register n (RI5CY_PCCR0) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 781 | Performance Counter Counter Register n (RI5CY_PCCR1) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 782 | Performance Counter Counter Register n (RI5CY_PCCR2) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 783 | Performance Counter Counter Register n (RI5CY_PCCR3) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 784 | Performance Counter Counter Register n (RI5CY_PCCR4) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 785 | Performance Counter Counter Register n (RI5CY_PCCR5) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |

*Table continues on the next page...*

**RI5CY memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 786 | Performance Counter Counter Register n (RI5CY_PCCR6) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 787 | Performance Counter Counter Register n (RI5CY_PCCR7) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 788 | Performance Counter Counter Register n (RI5CY_PCCR8) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 789 | Performance Counter Counter Register n (RI5CY_PCCR9) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 78A | Performance Counter Counter Register n (RI5CY_PCCR10) | 32 | R/W | 0000_0000h | 3.1.3.6.9/ 72 |
| 7A0 | Performance Counter Event Register (RI5CY_PCER) | 32 | R/W | 0000_0000h | 3.1.3.6.10/ 74 |
| 7A1 | Performance Counter Mode Register (RI5CY_PCMR) | 32 | R/W | 0000_0003h | 3.1.3.6.11/ 75 |
| 7B0 | Hardware Loop n Start (RI5CY_LPSTART0) | 32 | R/W | 0000_0000h | 3.1.3.6.12/ 76 |
| 7B1 | Hardware Loop n End (RI5CY_LPEND0) | 32 | R/W | 0000_0000h | 3.1.3.6.13/ 76 |
| 7B2 | Hardware Loop n Count (RI5CY_LPCOUNT0) | 32 | R/W | 0000_0000h | 3.1.3.6.14/ 76 |
| 7B4 | Hardware Loop n Start (RI5CY_LPSTART1) | 32 | R/W | 0000_0000h | 3.1.3.6.12/ 76 |
| 7B5 | Hardware Loop n End (RI5CY_LPEND1) | 32 | R/W | 0000_0000h | 3.1.3.6.13/ 76 |
| 7B6 | Hardware Loop n Count (RI5CY_LPCOUNT1) | 32 | R/W | 0000_0000h | 3.1.3.6.14/ 76 |
| C10 | Privilege Level (RI5CY_PRIVLV) | 32 | R | 0000_0003h | 3.1.3.6.15/ 77 |

## 3.1.3.6.1 User Status (RI5CY_USTATUS)

Address: 0h base + 0h offset = 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | UPIE | 0 | | | UIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_USTATUS field descriptions**

| Field | Description |
|---|---|
| 31–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>UPIE | User Previous Interrupt Enable<br><br>When an exception is encountered, UPIE will be set to UIE. When the uret instruction is executed, the value of UPIE will be stored to UIE.<br><br>0 Disable the user previous interrupt<br>1 Enable the user previous interrupt |
| 3–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>UIE | User Interrupt Enable<br><br>If you want to enable interrupt handling in your exception handler, set the User Interrupt Enable to 1'b1 inside your handler code.<br><br>0 Disable the user interrupt<br>1 Enable the user interrupt |

## 3.1.3.6.2 User Trap-Vector Base Address (RI5CY_UTVEC)

Address: 0h base + 5h offset = 5h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UTVEC | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_UTVEC field descriptions**

| Field | Description |
|---|---|
| 31–8<br>UTVEC | User Trap-Vector Base Address<br><br>When a user level exception is encountered, the core jumps to the corresponding handler using the content of the UTVEC as base address. |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 3.1.3.6.3 User Exception PC (RI5CY_UEPC)

Address: 0h base + 41h offset = 41h

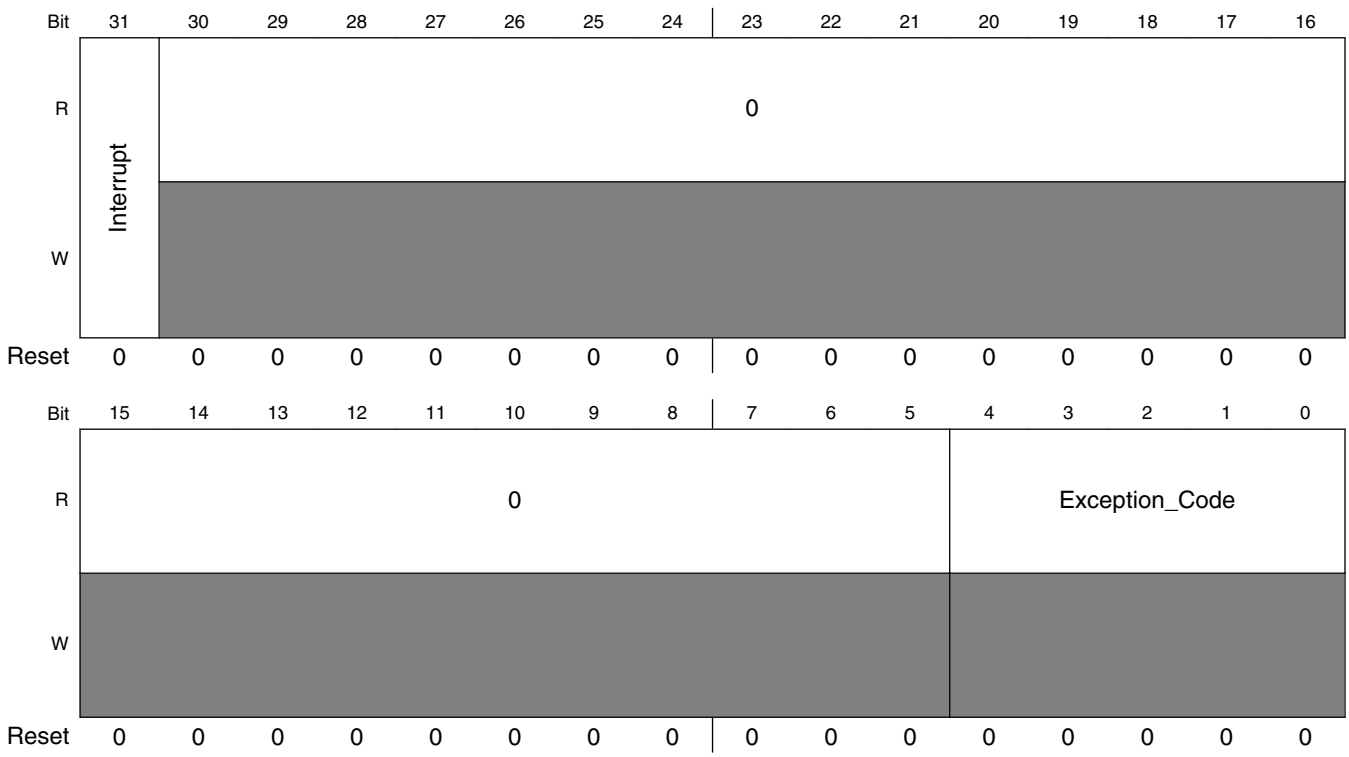
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UEPC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_UEPC field descriptions**

| Field | Description |
|---|---|
| UEPC | User Exception PC<br><br>When a user level exception is encountered, the current program counter is saved in UEPC, and the core jumps to the exception address. When an uret instruction is executed, the value from UEPC replaces the current program counter. |

## 3.1.3.6.4 User Cause (RI5CY_UCAUSE)

Address: 0h base + 42h offset = 42h




**RI5CY_UCAUSE field descriptions**

| Field | Description |
|---|---|
| 31<br>Interrupt | Interrupt<br><br>This bit is set when the exception was triggered by an interrupt. |
| 30–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| Exception_Code | Exception Code<br><br>When User-level exception is triggered by an interrupt, this field is the number of interrupt from 0-31. When it's a core exception, this field is the number of exception.<br><br>00010 Illegal instruction<br>00011 EBREAK |

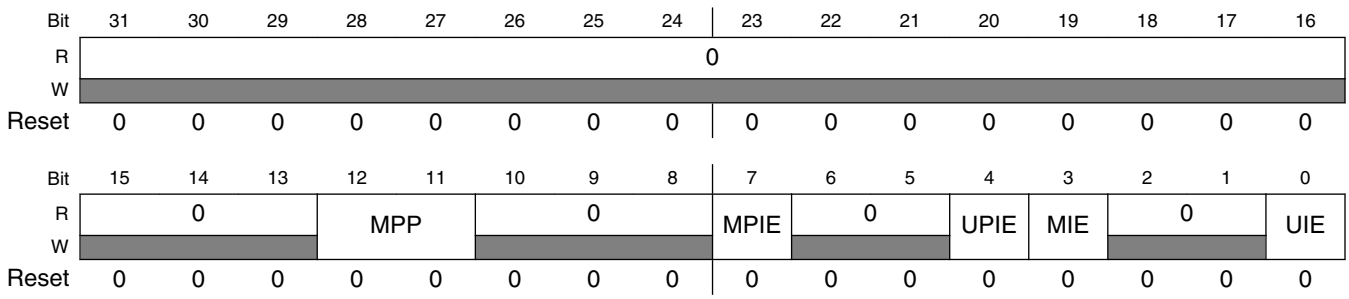
*Table continues on the next page...*

**RI5CY_UCAUSE field descriptions (continued)**

| Field | Description |
|---|---|
| | 00101 LSU load fault<br>00111 LSU store fault<br>01000 ECALL in User mode<br>01011 ECALL in Machine mode |

### 3.1.3.6.5 Machine Status (RI5CY_MSTATUS)

Address: 0h base + 300h offset = 300h




**RI5CY_MSTATUS field descriptions**

| Field | Description |
|---|---|
| 31–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–11<br>MPP | Previous Privilege Mode<br><br>When an exception is encountered, MPP will be set to RI5CY_PRIVLV. When the mret/uret instruction is executed, the value of MPP will be stored to RI5CY_PRIVLV.<br><br>00 User level<br>11 Machine level |
| 10–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>MPIE | Machine Previous Interrupt Enable<br><br>When an exception is encountered, MPIE will be set to MIE. When the mret instruction is executed, the value of MPIE will be stored to MIE.<br><br>0 Disable machine previous interrupt<br>1 Enable machine previous interrupt |
| 6–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 4<br>UPIE | User Previous Interrupt Enable<br><br>When an exception is encountered, UPIE will be set to UIE. When the mret instruction is executed, the value of UPIE will be stored to UIE.<br><br>0 Disable user previous interrupt<br>1 Enable user previous interrupt |

*Table continues on the next page...*

**RI5CY_MSTATUS field descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>MIE | Machine Interrupt Enable<br><br>If you want to enable interrupt handling in your exception handler, set the Interrupt Enable to 1'b1 inside your handler code.<br><br>0 Disable machine interrupt<br>1 Enable machine interrupt |
| 2–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>UIE | User Interrupt Enable<br><br>If you want to enable interrupt handling in your exception handler, set the Interrupt Enable to 1'b1 inside your handler code.<br><br>0 Disable user interrupt<br>1 Enable user interrupt |

## 3.1.3.6.6 Machine Trap-Vector Base Address (RI5CY_MTVEC)

Address: 0h base + 305h offset = 305h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MTVEC | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_MTVEC field descriptions**

| Field | Description |
|---|---|
| 31–8<br>MTVEC | MTVEC<br><br>When an exception is encountered, the core jumps to the corresponding handler using the content of the MTVEC as base address. |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 3.1.3.6.7 Machine Exception PC (RI5CY_MEPC)

Address: 0h base + 341h offset = 341h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MEPC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_MEPC field descriptions**

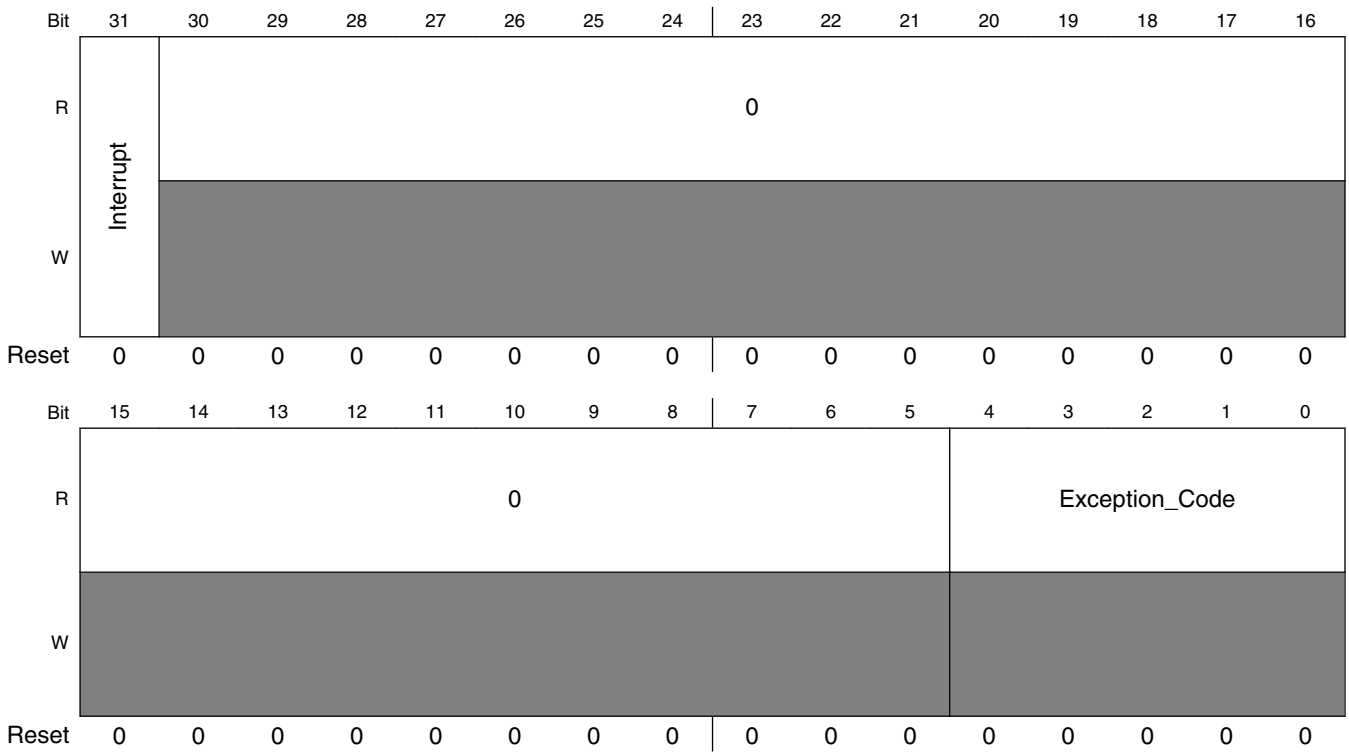
| Field | Description |
|---|---|
| MEPC | Machine Exception PC |

**RI5CY_MEPC field descriptions (continued)**

| Field | Description |
|---|---|
| | When an exception is encountered, the current program counter is saved in MEPC, and the core jumps to the exception address. When an mret instruction is executed, the value from MEPC replaces the current program counter. |

## 3.1.3.6.8 Machine Cause (RI5CY_MCAUSE)

Address: 0h base + 342h offset = 342h




**RI5CY_MCAUSE field descriptions**

| Field | Description |
|---|---|
| 31<br>Interrupt | Interrupt<br><br>This bit is set when the exception was triggered by an interrupt. |
| 30–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| Exception_Code | Exception Code<br><br>When machine level exception is triggered by an interrupt, this field is the number of interrupt from 0-31. When it's a core exception, this field is the number of exception.<br><br>00010 Illegal instruction<br>00011 EBREAK<br>00101 LSU load fault<br>00111 LSU store fault |

*Table continues on the next page...*

**RI5CY_MCAUSE field descriptions (continued)**

| Field | Description |
|---|---|
| | 01000 ECALL in User mode<br>01011 ECALL in Machine mode |

### 3.1.3.6.9 Performance Counter Counter Register n (RI5CY_PCCR*n*)

PCCR registers support both saturating and wrap-around arithmetic. This is controlled by the saturation bit in PCMR.

| Register | Name | Description |
|---|---|---|
| PCCR0 | CYCLES | Counts the number of cycles the core was active (not sleeping) |
| PCCR1 | INSTR | Counts the number of instructions executed |
| PCCR2 | LD_STALL | Number of load data hazards |
| PCCR3 | JR_STALL | Number of jump register data hazards |
| PCCR4 | IMISS | Cycles waiting for instruction fetches, i.e. number of instructions wasted due to non-ideal caching |
| PCCR5 | LD | Number of data memory loads executed. Misaligned accesses are counted twice |
| PCCR6 | ST | Number of data memory stores executed. Misaligned accesses are counted twice |
| PCCR7 | JUMP | Number of unconditional jumps (j, jal, jr, jalr) |
| PCCR8 | BRANCH | Number of branches.<br>Counts taken and not taken branches |
| PCCR9 | BTAKEN | Number of taken branches. |
| PCCR10 | RVC | Number of compressed instructions executed |

In the FPGA, RTL simulation and Virtual-Platform there are individual counters for each event type, i.e. PCCR0-10 each represent a separate register. To save area in the ASIC, there is only one counter and one counter register. Accessing PCCR0-10 will access the same counter register in the ASIC.

The following figure shows how events are first masked with the PCER register and then ORed together to increase the one performance counter PCCR.

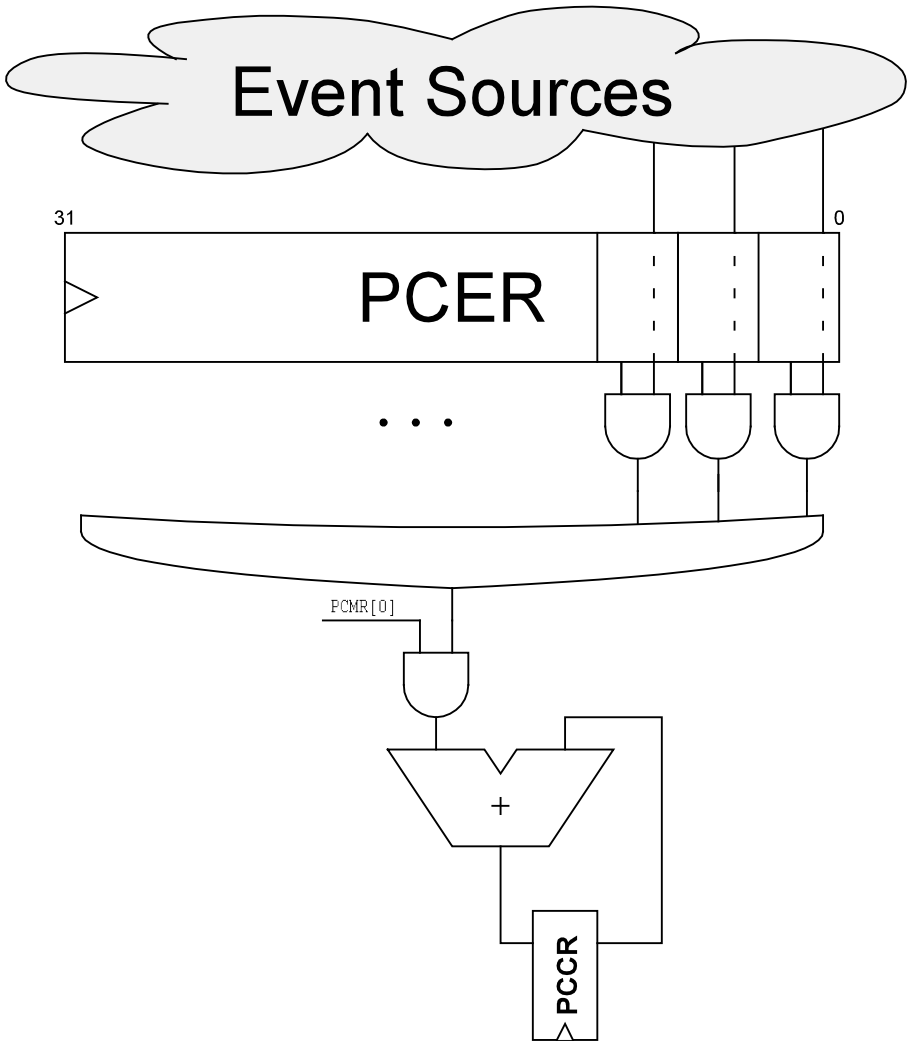



**Figure 3-2. Events and PCCR, PCMR and PCER on the ASIC**

Address: 0h base + 780h offset + (1d × i), where i=0d to 10d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Counter | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## RI5CY_PCCR*n* field descriptions

| Field | Description |
|---|---|
| Counter | Counter<br><br>See register descriptions for details. |

### 3.1.3.6.10 Performance Counter Event Register (RI5CY_PCER)

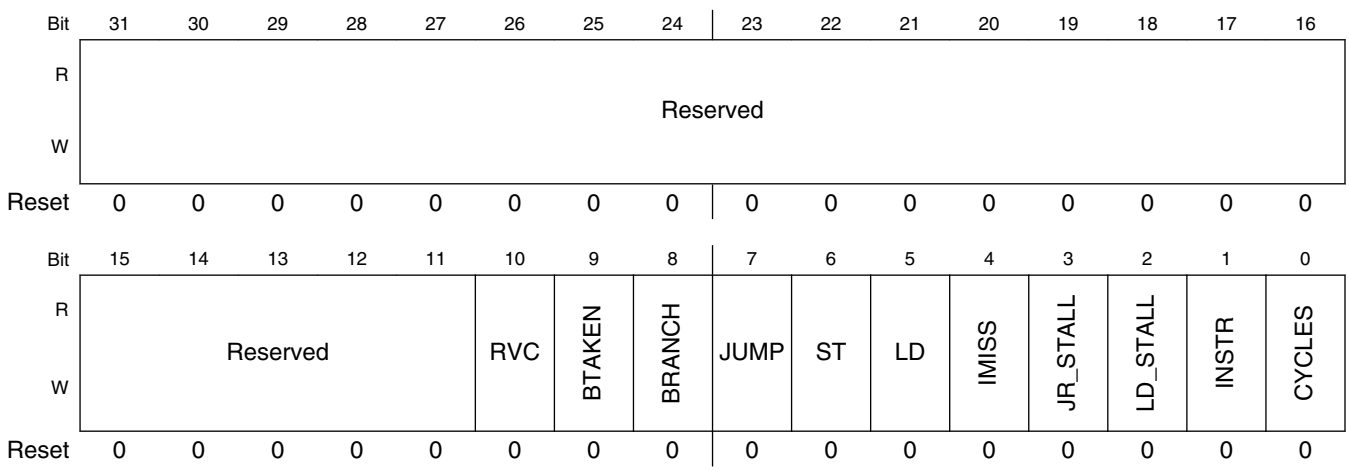
Each bit in the PCER register controls one performance counter. If the bit is 1, the counter is enabled and starts counting events. If it is 0, the counter is disabled and its value won't change.

In the ASIC there is only one counter register, thus all counter events are masked by PCER and ORed together, i.e. if one of the enabled event happens, the counter will be increased. If multiple non-masked events happen at the same time, the counter will only be increased by one.

In order to be able to count separate events on the ASIC, the program can be executed in a loop with different events configured.

In the FPGA or RTL simulation version, each event has its own counter and can be accessed separately.

Address: 0h base + 7A0h offset = 7A0h




**RI5CY_PCER field descriptions**

| Field | Description |
|---|---|
| 31–11<br>Reserved | This field is reserved. |
| 10<br>RVC | RVC<br><br>It counts the number of compressed instructions executed. |
| 9<br>BTAKEN | BTAKEN<br><br>It counts the number of taken branches. |
| 8<br>BRANCH | BRANCH<br><br>It counts the number of branches, no matter taken or not taken. |
| 7<br>JUMP | JUMP |

*Table continues on the next page...*

**RI5CY_PCER field descriptions (continued)**

| Field | Description |
|---|---|
| | It counts the number of unconditional jumps (j, jal, jr, jalr) |
| 6<br>ST | ST<br><br>It counts the number of data memory stores executed. Misaligned accesses are counted twice. |
| 5<br>LD | LD<br><br>It counts the number of data memory loads executed. Misaligned accesses are counted twice. |
| 4<br>IMISS | IMISS<br><br>It counts the cycles waiting for instruction fetches, i.e. number of instructions wasted due to non-ideal caching. |
| 3<br>JR_STALL | JR_STALL<br><br>It counts the number of jump register data hazards. |
| 2<br>LD_STALL | LD_STALL<br><br>It counts the number of load data hazards. |
| 1<br>INSTR | INSTR<br><br>It counts the number of instructions executed. |
| 0<br>CYCLES | CYCLES<br><br>It counts the number of cycles that the core is active (not sleeping). |

## 3.1.3.6.11 Performance Counter Mode Register (RI5CY_PCMR)

Address: 0h base + 7A1h offset = 7A1h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | Saturation | GlobalEnable |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### RI5CY_PCMR field descriptions

| Field | Description |
|---|---|
| 31–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 1<br>Saturation | Saturation<br><br>If this bit is set, saturating arithmetic is used in the performance counter counters. After reset, this bit is set. |
| 0<br>GlobalEnable | Global Enable<br><br>Activate/deactivate all performance counters. If this bit is 0, all performance counters are disabled. After reset, this bit is set. |

## 3.1.3.6.12 Hardware Loop n Start (RI5CY_LPSTART*n*)

Address: 0h base + 7B0h offset + (4d × i), where i=0d to 1d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPSTART | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RI5CY_LPSTART*n* field descriptions

| Field | Description |
|---|---|
| LPSTART | Hardware Loop n Start |

## 3.1.3.6.13 Hardware Loop n End (RI5CY_LPEND*n*)

Address: 0h base + 7B1h offset + (4d × i), where i=0d to 1d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPEND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### RI5CY_LPEND*n* field descriptions

| Field | Description |
|---|---|
| LPEND | Hardware Loop n End |

## 3.1.3.6.14 Hardware Loop n Count (RI5CY_LPCOUNT*n*)

Address: 0h base + 7B2h offset + (4d × i), where i=0d to 1d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPCOUNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RI5CY_LPCOUNT*n* field descriptions**

| Field | Description |
|---|---|
| LPCOUNT | Hardware Loop n Counter<br><br>This counter is decremented every time the loop body is executed. |

## 3.1.3.6.15 Privilege Level (RI5CY_PRIVLV)

Address: 0h base + C10h offset = C10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | PRVLVL | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**RI5CY_PRIVLV field descriptions**

| Field | Description |
|---|---|
| 31–2<br>Reserved | This field is reserved. |
| PRVLVL | Privilege Level<br><br>This field indicates current privilege level of RI5CY core.<br><br>00 User level<br>11 Machine level |

## 3.1.3.7 Exceptions and Interrupts

RI5CY supports interrupts, exceptions on illegal instructions.

**Table 3-1. Interrupt/Exception Offset Vector Table**

| Offset address | Description |
|---|---|
| 0x00-0x7C | Interrupts 0 – 31 |
| 0x80 | Reset |
| 0x84 | Illegal Instruction |
| 0x88 | ECALL Instruction Executed |
| 0x8C | LSU Error (Invalid Memory Access) |

The base address of the interrupt vector table is given by the boot address. The most significant 3 bytes of the boot address given to the core are used for the first instruction fetch of the core and as the basis of the interrupt vector table. The core starts fetching at the address made by concatenating the most significant 3 bytes of the boot address and the reset value (0x80) as the least significant byte. The boot address can be changed after the first instruction was fetched to change the interrupt vector table address. It is assumed that the boot address is supplied via a register to avoid long paths to the instruction fetch unit.

#### 3.1.3.7.1 Interrupts

Interrupts can only be enabled/disabled on a global basis and not individually. It is assumed that there is an event/interrupt controller outside of the core that performs masking and buffering of the interrupt lines. The global interrupt enable is done via the CSR register MSTATUS.

Multiple interrupts requests are assumed to be handled by event/interrupt controller. When an interrupt is taken, the core gives an acknowledge signal to the event/interrupt controller as well as the interrupt id taken.

#### 3.1.3.7.2 Exceptions

The illegal instruction exception and ecall instruction exceptions cannot be disabled and are always active.

#### 3.1.3.7.3 Handling

RI5CY does support nested interrupt/exception handling. Exceptions inside interrupt/ exception handlers cause another exception, thus exceptions during the critical part of your exception handlers, i.e. before having saved the MEPC and MESTATUS registers, will cause those register to be overwritten. Interrupts during interrupt/exception handlers are disabled by default, but can be explicitly enabled if desired.

Upon executing an mret instruction, the core jumps to the program counter saved in the CSR register MEPC and restores the MPIE value of the register MSTATUS to IE. When entering an interrupt/exception handler, the core sets MEPC to the current program counter and saves the current value of MIE in MPIE of the MSTATUS register.

### 3.1.3.8 Debug Unit

### 3.1.3.8.1 Address Map

**Table 3-2. Debug Unit Address Map**

| Address | Name | Description |
|---|---|---|
| 0x0000-0x007F | Debug Registers | Always accessible, even when the core is running |
| 0x400-0x47F | GPR (x0-x31) | General Purpose Registers<br>Only accessible if the core is halted |
| 0x2000-0x20FF | Debug Registers | Only accessible if the core is halted |
| 0x4000-0x7FFF | CSR | Control and Status Registers<br>Only accessible if the core is halted |

**NOTE**

Addresses are intended for a bus system with 32-bit wide words.

Addresses have to be aligned to word-boundaries.

### 3.1.3.8.2 Debug memory map and register descriptions

**Debug memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| F900_0000 | Debug Control (Debug_CTRL) | 32 | R/W | 0000_0000h | 3.1.3.8.2.1/ 80 |
| F900_0004 | Debug Hit (Debug_HIT) | 32 | R/W | 0000_0000h | 3.1.3.8.2.2/ 81 |
| F900_0008 | Debug Interrupt Enable (Debug_IE) | 32 | R/W | 0000_0000h | 3.1.3.8.2.3/ 82 |
| F900_000C | Debug Cause (Debug_CAUSE) | 32 | R | 0000_0000h | 3.1.3.8.2.4/ 83 |
| F900_0040 | Debug Hardware Breakpoint Control Register n (Debug_BPCTRL0) | 32 | R/W | 0000_0000h | 3.1.3.8.2.5/ 84 |
| F900_0044 | Debug Hardware Breakpoint Data Register n (Debug_BPDATA0) | 32 | R/W | 0000_0000h | 3.1.3.8.2.6/ 85 |
| F900_0048 | Debug Hardware Breakpoint Control Register n (Debug_BPCTRL1) | 32 | R/W | 0000_0000h | 3.1.3.8.2.5/ 84 |
| F900_004C | Debug Hardware Breakpoint Data Register n (Debug_BPDATA1) | 32 | R/W | 0000_0000h | 3.1.3.8.2.6/ 85 |
| F900_0050 | Debug Hardware Breakpoint Control Register n (Debug_BPCTRL2) | 32 | R/W | 0000_0000h | 3.1.3.8.2.5/ 84 |
| F900_0054 | Debug Hardware Breakpoint Data Register n (Debug_BPDATA2) | 32 | R/W | 0000_0000h | 3.1.3.8.2.6/ 85 |

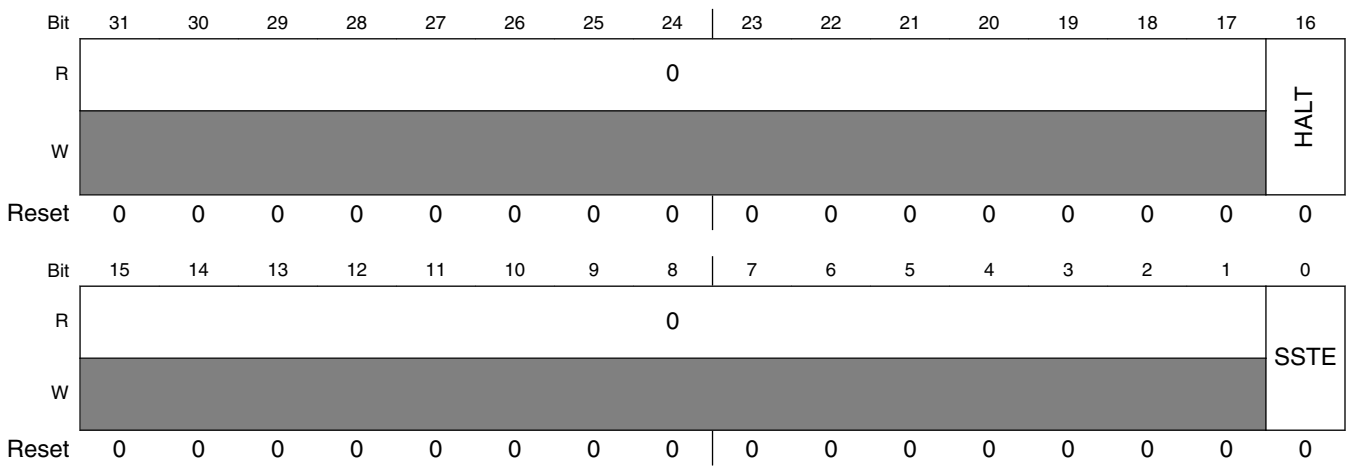
*Table continues on the next page...*

## Debug memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| F900_0058 | Debug Hardware Breakpoint Control Register n (Debug_BPCTRL3) | 32 | R/W | 0000_0000h | 3.1.3.8.2.5/ 84 |
| F900_005C | Debug Hardware Breakpoint Data Register n (Debug_BPDATA3) | 32 | R/W | 0000_0000h | 3.1.3.8.2.6/ 85 |
| F900_2000 | Debug Next Program Counter (Debug_NPC) | 32 | R/W | 0000_0000h | 3.1.3.8.2.7/ 85 |
| F900_2004 | Debug Previous Program Counter (Debug_PPC) | 32 | R | 0000_0000h | 3.1.3.8.2.8/ 86 |

### 3.1.3.8.2.1 Debug Control (Debug_CTRL)

Address: F900_0000h base + 0h offset = F900_0000h

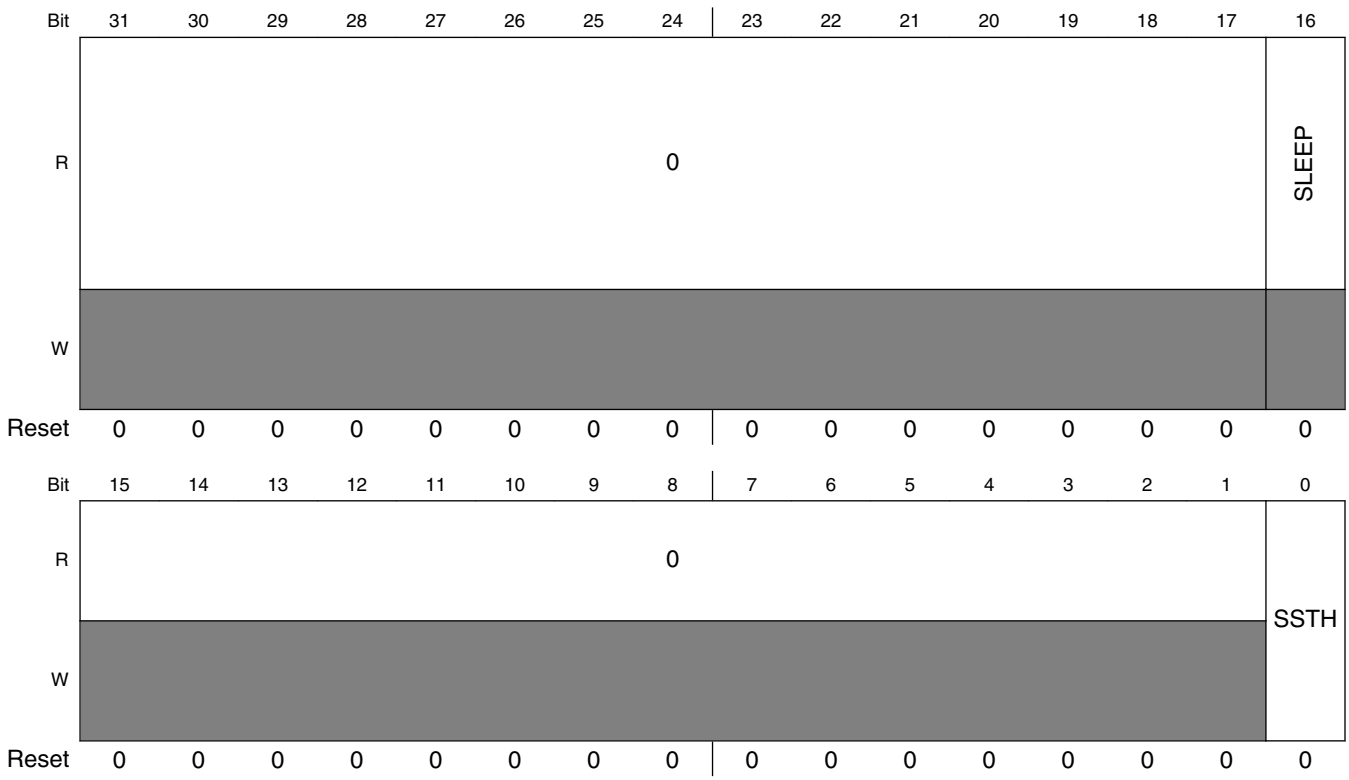



## Debug_CTRL field descriptions

| Field | Description |
|---|---|
| 31–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>HALT | HALT<br><br>When 1 written, core enters debug mode, when 0 written, core exits debug mode. When read, 1 means core is in debug mode<br><br>0 Core is not in debug mode.<br>1 Core is in debug mode. |
| 15–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>SSTE | Single-step enable<br><br>When single-step is enabled, writing 0 to HALT will cause the core to step by one step.<br><br>0 Single-step is disabled.<br>1 Single-step is enabled. |

### 3.1.3.8.2.2 Debug Hit (Debug_HIT)

Address: F900_0000h base + 4h offset = F900_0004h



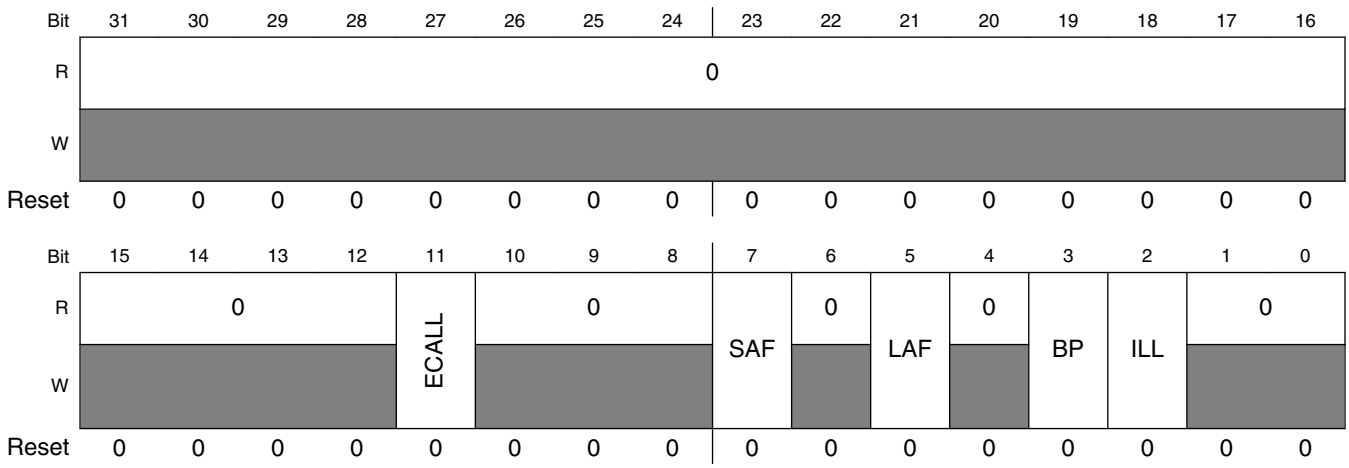

**Debug_HIT field descriptions**

| Field | Description |
|---|---|
| 31–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>SLEEP | SLEEP<br><br>Set when the core is in a sleeping state and waits for an event. |
| 15–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>SSTH | SSTH<br><br>Single-step hit, sticky bit that must be cleared by external debugger. |

### 3.1.3.8.2.3 Debug Interrupt Enable (Debug_IE)

This register provides the ability for interrupts to cause traps. When enabled, exceptions cause traps, otherwise normal exceptions.

Address: F900_0000h base + 8h offset = F900_0008h




**Debug_IE field descriptions**

| Field | Description |
|---|---|
| 31–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11<br>ECALL | ECALL Enable<br><br>0 Environment call from M mode does not cause trap.<br>1 Environment call from M mode causes trap. |
| 10–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>SAF | SAF Enable<br><br>When 1 is written to the LAF, this bit is also set.<br><br>0 Store access fault does not cause trap.<br>1 Store access fault causes trap. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>LAF | LAF Enable<br><br>When 1 is written to the SAF, this bit is also set.<br><br>0 Load access fault does not cause trap.<br>1 Load access fault causes trap. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>BP | BP Enable |

*Table continues on the next page...*

**Debug_IE field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 EBREAK instruction does not cause trap.<br>1 EBREAK instruction causes trap. |
| 2<br>ILL | ILL Enable<br><br>0 Illegal instruction does not cause trap.<br>1 Illegal instruction causes trap. . |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

### 3.1.3.8.2.4 Debug Cause (Debug_CAUSE)

Address: F900_0000h base + Ch offset = F900_000Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQ | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

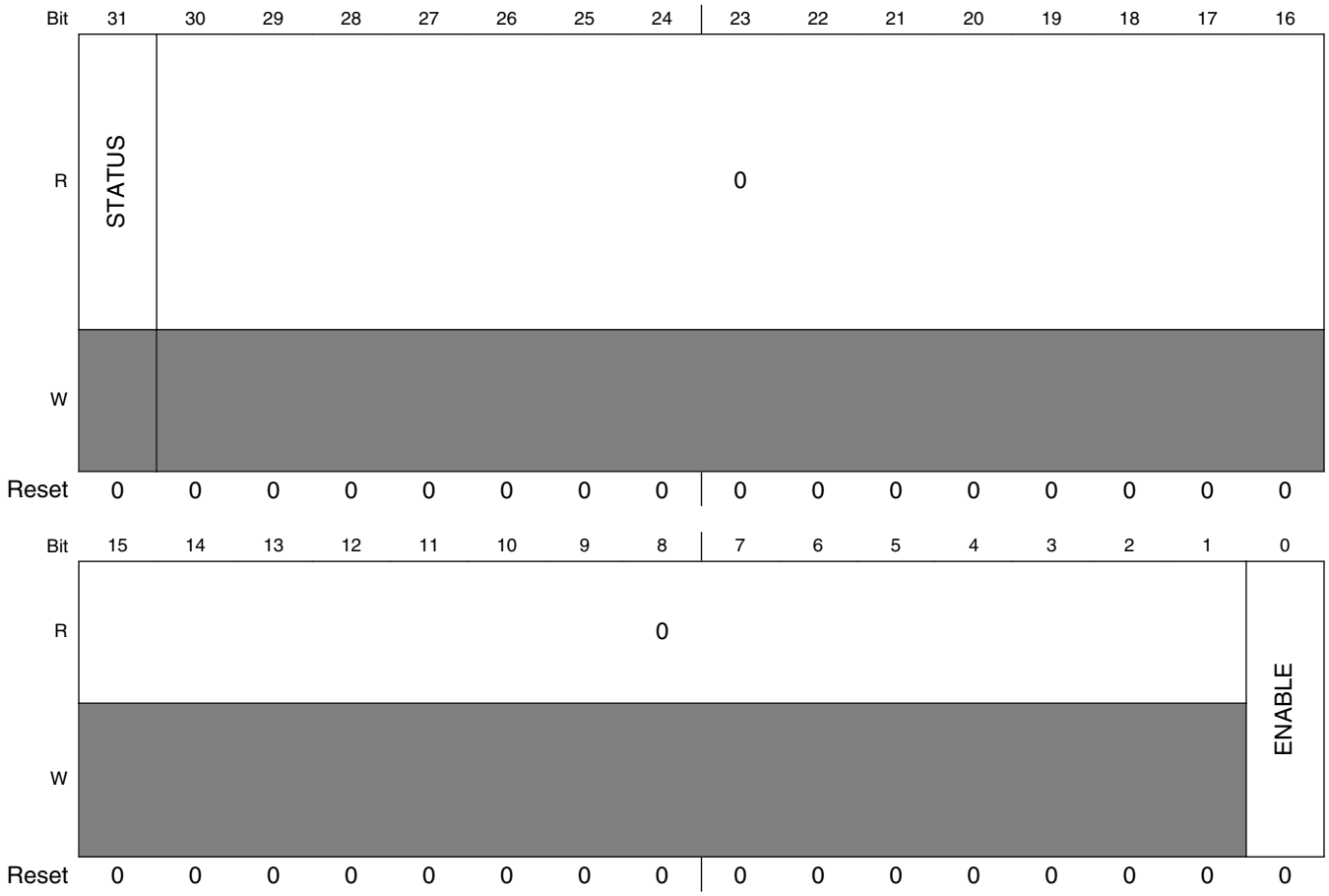
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | CAUSE | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Debug_CAUSE field descriptions**

| Field | Description |
|---|---|
| 31<br>IRQ | IRQ<br><br>0 Core in debug mode is caused by core exceptions.<br>1 Core in debug mode is caused by interrupts. |
| 30–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| CAUSE | Exception/interrupt number<br><br>If IRQ is 1, this field is the number of interrupt that causes core to enter debug mode. If IRQ is 0, this field is the code of exception that causes core to enter debug mode.<br><br>00010 Illegal instruction<br>00011 EBREAK<br>00101 LSU load fault<br>00111 LSU store fault<br>01000 ECALL in User mode<br>01011 ECALL in Machine mode |

### 3.1.3.8.2.5 Debug Hardware Breakpoint Control Register n (Debug_BPCTRL*n*)

Address: F900_0000h base + 40h offset + (8d × i), where i=0d to 3d




**Debug_BPCTRL*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>STATUS | Breakpoint hit status<br><br>0 Breakpoint does not hit.<br>1 Breakpoint hits. |
| 30–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>ENABLE | Breakpoint Enable<br><br>0 Breakpoint is disabled<br>1 Breakpoint is enabled |

### 3.1.3.8.2.6 Debug Hardware Breakpoint Data Register n (Debug_BPDATA*n*)

Address: F900_0000h base + 44h offset + (8d × i), where i=0d to 3d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PC | | | | | | | | | | | | | | | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Debug_BPDATA*n* field descriptions**

| Field | Description |
|---|---|
| 31–1<br>PC | PC<br><br>Set break point PC value. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

### 3.1.3.8.2.7 Debug Next Program Counter (Debug_NPC)

Values of PPC and NPC when entering debug mode:

| Reason | PPC | NPC | GDB Sigval |
|---|---|---|---|
| ebreak | ebreak instruction | next instruction | TRAP |
| ecall | ecall instruction | IVT entry | TRAP |
| illegal instruction | illegal instruction | IVT entry | ILL |
| invalid mem access | load/store instruction | IVT entry | SEGV |
| interrupt | last instruction | IVT entry | INT |
| halt | last instruction | next instruction | TRAP |
| single-step | last instruction | next instruction | TRAP |

Address: F900_0000h base + 2000h offset = F900_2000h

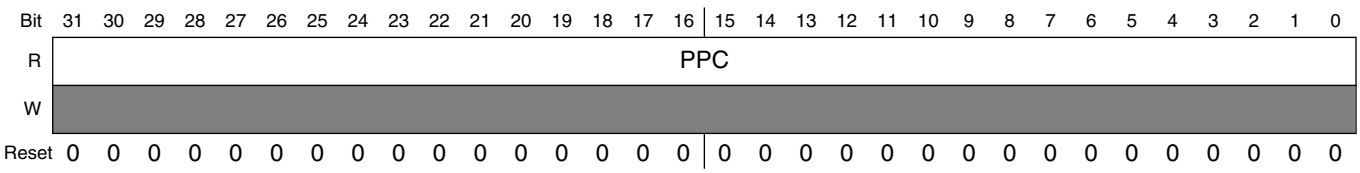
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NPC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Debug_NPC field descriptions**

| Field | Description |
|---|---|
| NPC | NPC<br><br>Next PC to be executed. |

#### 3.1.3.8.2.8 Debug Previous Program Counter (Debug_PPC)

Address: F900_0000h base + 2004h offset = F900_2004h




**Debug_PPC field descriptions**

| Field | Description |
|---|---|
| PPC | PPC<br><br>Previous PC, already executed. |

### 3.1.3.8.3 Control and Status Registers

These registers can only be accessed when core is in debug mode.

| Address | Name | Description |
|---|---|---|
| 0x4000 | CSR 0 = 0x000 | CSR |
| ... | ... | ... |
| 0x7FFC | CSR 4095 = 0xFFF | CSR |

## 3.1.3.9 Instruction Set Extensions

### 3.1.3.9.1 Post-Incrementing Load and Store Instructions

Post-Incrementing load and store instructions perform a load, or a store, respectively, while at the same time incrementing the address that was used for the memory access. Since it is a post-incrementing scheme, the base address is used for the access and the modified address is written back to the register-file. There are versions of those instructions that use immediates and those that use registers as offsets. The base address always comes from a register.

#### 3.1.3.9.1.1 Load operations

**Table 3-3. Load operations**

| Mnemonic | Description |
|---|---|
| **Register-Immediate Loads with Post-Increment** | |

*Table continues on the next page...*

**Table 3-3. Load operations (continued)**

| Mnemonic | Description |
|---|---|
| **p.lb rD, Imm(rs1!)** | rD = Sext(Mem8(rs1)) rs1 += Imm[11:0] |
| **p.lbu rD, Imm(rs1!)** | rD = Zext(Mem8(rs1)) rs1 += Imm[11:0] |
| **p.lh rD, Imm(rs1!)** | rD = Sext(Mem16(rs1)) rs1 += Imm[11:0] |
| **p.lhu rD, Imm(rs1!)** | rD = Zext(Mem16(rs1)) rs1 += Imm[11:0] |
| **p.lw rD, Imm(rs1!)** | rD = Mem32(rs1) rs1 += Imm[11:0] |
| **Register-Register Loads with Post-Increment** | |
| **p.lb rD, rs2(rs1!)** | rD = Sext(Mem8(rs1)) rs1 += rs2 |
| **p.lbu rD, rs2(rs1!)** | rD = Zext(Mem8(rs1)) rs1 += rs2 |
| **p.lh rD, rs2(rs1!)** | rD = Sext(Mem16(rs1)) rs1 += rs2 |
| **p.lhu rD, rs2(rs1!)** | rD = Zext(Mem16(rs1)) rs1 += rs2 |
| **p.lw rD, rs2(rs1!)** | rD = Mem32(rs1) rs1 += rs2 |
| **Register-Register Loads** | |
| **p.lb rD, rs2(rs1)** | rD = Sext(Mem8(rs1 + rs2)) |
| **p.lbu rD, rs2(rs1)** | rD = Zext(Mem8(rs1 + rs2)) |
| **p.lh rD, rs2(rs1)** | rD = Sext(Mem16(rs1 + rs2)) |
| **p.lhu rD, rs2(rs1)** | rD = Zext(Mem16(rs1 + rs2)) |
| **p.lw rD, rs2(rs1)** | rD = Mem32(rs1 + rs2) |

### 3.1.3.9.1.2 Store operations

**Table 3-4. Store operations**

| Mnemonic | Description |
|---|---|
| **Register-Immediate Stores with Post-Increment** | |
| **p.sb rs2, Imm(rs1!)** | Mem8(rs1) = rs2 rs1 += Imm[11:0] |
| **p.sh rs2, Imm(rs1!)** | Mem16(rs1) = rs2 rs1 += Imm[11:0] |
| **p.sw rs2, Imm(rs1!)** | Mem32(rs1) = rs2 rs1 += Imm[11:0] |
| **Register-Register Stores with Post-Increment** | |
| **p.sb rs2, rs3(rs1!)** | Mem8(rs1) = rs2 rs1 += rs3 |
| **p.sh rs2, rs3(rs1!)** | Mem16(rs1) = rs2 rs1 += rs3 |
| **p.sw rs2, rs3(rs1!)** | Mem32(rs1) = rs2 rs1 += rs3 |
| **Register-Register Stores** | |
| **p.sb rs2, rs3(rs1)** | Mem8(rs1 + rs3) = rs2 |
| **p.sh rs2 rs3(rs1)** | Mem16(rs1 + rs3) = rs2 |
| **p.sw rs2, rs3(rs1)** | Mem32(rs1 + rs3) = rs2 |

#### 3.1.3.9.2 Hardware loops

RI5CY supports 2 levels of nested hardware loops. The loop has to be setup before entering the loop body. For this purpose, there are two methods, either the long commands that separately set start- and end-addresses of the loop and the number of iterations, or the short command that does all of this in a single instruction. The short command has a limited range for the number of instructions contained in the loop and the loop must start in the next instruction after the setup instruction.

Loop number 0 has higher priority than loop number 1 in a nested loop configuration, meaning that loop 0 represents the inner loop.

A hardware loop is subject to the following constraints:

- Minimum of 2 instructions in the loop body.
- Loop counter has to be bigger than 0, since the loop body is always entered at least once.

##### 3.1.3.9.2.1 Operations

**Table 3-5. Hardware loops operations**

| Mnemonic | | Description |
|---|---|---|
| **Long Hardware Loop Setup instructions** | | |
| **lp.starti** | **L, uimmL** | lpstart[L] = PC + (uimmL << 1) |
| **lp.endi** | **L, uimmL** | lpend[L] = PC + (uimmL << 1) |
| **lp.count** | **L, rs1** | lpcount[L] = rs1 |
| **lp.counti** | **L, uimmL** | lpcount[L] = uimmL |
| **Short Hardware Loop Setup Instructions** | | |
| **lp.setup** | **L, rs1, uimmL** | lpstart[L] = pc + 4<br>lpend[L] = pc + (uimmL << 1) lpcount[L] = rs1 |
| **lp.setupi** | **L, uimmS, uimmL** | lpstart[L] = pc + 4<br>lpend[L] = pc + (uimmS << 1) lpcount[L] = uimmL |

#### 3.1.3.9.3 ALU

The ALU extensions are split into several subgroups that belong together.

- Bit manipulation instructions are useful to work on single bits or groups of bits within a word, see Bit manipulation operations.

- General ALU instructions try to fuse common used sequences into a single instruction and thus increase the performance of small kernels that use those sequence, see General ALU operations.
- Immediate branching instructions are useful to compare a register with an immediate value before taking or not a branch, see Immediate branching operations.

#### 3.1.3.9.3.1 Bit manipulation operations

**Table 3-6. Bit manipulation operations**

| Mnemonic | | Description |
|---|---|---|
| **p.extract** | **rD, rs1, Is3, Is2** | rD = Sext((rs1 & ((1 << Is3) – 1) << Is2) >> Is2)<br>Note: Is3 + Is2 must be <= 32 |
| **p.extractu** | **rD, rs1, Is3, Is2** | rD = Zext((rs1 & ((1 << Is3) – 1) << Is2) >> Is2)<br>Note: Is3 + Is2 must be <= 32 |
| **p.extractr** | **rD, rs1, rs2** | rD = Sext((rs1 & ((1 << rs2[9:5]) – 1) << rs2[4:0]) >> rs2[4:0])<br>Note: rs2[9:5]+ rs2[4:0] must be <= 32 |
| **p.extractur** | **rD, rs1, rs2** | rD = Zext((rs1 & ((1 << rs2[9:5]) – 1) << rs2[4:0]) >> rs2[4:0])<br>Note: rs2[9:5]+ rs2[4:0] must be <= 32 |
| **p.insert** | **rD, rs1, Is3, Is2** | rD = rD \| (rs1[Is3:0] << Is2)<br>Note: Is3 + Is2 must be <= 32, the rest of the bits of rD are passed through and are not modified |
| **p.insertr** | **rD, rs1, rs2** | rD = rD \| (rs1[Is3:0] << rs2[4:0])<br>Note: rs2[9:5]+ rs2[4:0] must be <= 32, the rest of the bits of rD are passed through and are not modified |
| **p.bclr** | **rD, rs1, Is3, Is2** | rD = rs1 & ~(((1 << (Is3+1)) – 1) << Is2)<br>Note: Is3 + Is2 must be <= 32 |
| **p.bclrr** | **rD, rs1, rs2** | rD = rs1 & ~(((1 << (rs2[9:5]+1)) – 1) << rs2[4:0])<br>Note: rs2[9:5]+ rs2[4:0] must be <= 32 |
| **p.bset** | **rD, rs1, Is3, Is2** | rD = rs1 \| (((1 << (Is3+1)) – 1) << Is2)<br>Note: Is3 + Is2 must be <= 32 |
| **p.bsetr** | **rD, rs1, rs2** | rD = rs1 \| (((1 << (rs2[9:5]+1)) – 1) << rs2[4:0])<br>Note: rs2[9:5]+ rs2[4:0] must be <= 32 |
| **p.ff1** | **rD, rs1** | rD = bit position of the first bit set in rs1, starting from LSB. If bit 0 is set, rD will be 0. If only bit 31 is set, rD will be 31.<br>If rs1 is 0, rD will be 32. |
| **p.fl1** | **rD, rs1** | rD = bit position of the last bit set in rs1, starting from MSB. If bit 31 is set, rD will be 31. If only bit 0 is set, rD will be 0.<br>If rs1 is 0, rD will be 32. |
| **p.clb** | **rD, rs1** | rD = count leading bits of rs1<br>Note: This is the number of consecutive 1's or 0's from MSB.<br>Note: If rs1 is 0, rD will be 0. |
| **p.cnt** | **rD, rs1** | rD = Population count of rs1, i.e. number of bits set in rs1 |
| **p.ror** | **rD, rs1, rs2** | rD = RotateRight(rs1, rs2) |

### 3.1.3.9.3.2 General ALU operations

**Table 3-7. General ALU operations**

| Mnemonic | | Description |
|---|---|---|
| **p.abs** | **rD, rs1** | rD = rs1 < 0 ? –rs1 : rs1 |
| **p.slet** | **rD, rs1, rs2** | rD = rs1 <= rs2 ? 1 : 0 Note: Comparison issigned |
| **p.sletu** | **rD, rs1, rs2** | rD = rs1 <= rs2 ? 1 : 0<br>Note: Comparison is unsigned |
| **p.min** | **rD, rs1, rs2** | rD = rs1 < rs2 ? rs1 : rs2 Note: Comparison is signed |
| **p.minu** | **rD, rs1, rs2** | rD = rs1 < rs2 ? rs1 : rs2 Note: Comparison isunsigned |
| **p.max** | **rD, rs1, rs2** | rD = rs1 < rs2 ? rs2 : rs1 Note: Comparison is signed |
| **p.maxu** | **rD, rs1, rs2** | rD = rs1 < rs2 ? rs2 : rs1 Note: Comparison isunsigned |
| **p.exths** | **rD, rs1** | rD = Sext(rs1[15:0]) |
| **p.exthz** | **rD, rs1** | rD = Zext(rs1[15:0]) |
| **p.extbs** | **rD, rs1** | rD = Sext(rs1[7:0]) |
| **p.extbz** | **rD, rs1** | rD = Zext(rs1[7:0]) |
| **p.clip** | **rD, rs1, Is2** | if rs1 <= -2^(Is2-1), rD = -2^(Is2-1),<br>else if rs1 >= 2^(Is2-1)–1, rD = 2^(Is2-1)-1, else rD = rs1 |
| **p.clipr** | **rD, rs1, rs2** | if rs1 <= -(rs2+1), rD = -(rs2+1), else if rs1 >=rs2, rD = rs2,<br>else rD = rs1 |
| **p.clipu** | **rD, rs1, Is2** | if rs1 <= 0, rD = 0,<br>else if rs1 >= 2^(Is2–1)-1, rD = 2^(Is2-1)-1, else rD = rs1 |
| **p.clipur** | **rD, rs1, rs2** | if rs1 <= 0, rD = 0,<br>else if rs1 >= rs2, rD = rs2, else rD = rs1 |
| **p.addN** | **rD, rs1, rs2, Is3** | rD = (rs1 + rs2) >>> Is3<br>Note: Arithmetic shift right. Setting Is3 to 2 replaces former p.avg |
| **p.adduN** | **rD, rs1, rs2, Is3** | rD = (rs1 + rs2) >> Is3<br>Note: Logical shift right. Setting Is3 to 2 replaces former p.avg |
| **p.addRN** | **rD, rs1, rs2, Is3** | rD = (rs1 + rs2 + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right. |
| **p.adduRN** | **rD, rs1, rs2, Is3** | rD = (rs1 + rs2 + 2^(Is3-1))) >> Is3<br>Note: Logical shift right. |
| **p.addNr** | **rD, rs1, rs2** | rD = (rD + rs1) >>> rs2[4:0]<br>Note: Arithmetic shift right. |
| **p.adduNr** | **rD, rs1, rs2** | rD = (rD + rs1) >> rs2[4:0] |
| **p.addRNr** | **rD, rs1, rs2** | rD = (rD + rs1 + 2^(rs2[4:0])) >>> rs2[4:0]<br>Note: Arithmetic shift right. |
| **p.adduRNr** | **rD, rs1, rs2** | rD = (rD + rs1 + 2^(rs2[4:0]-1))) >> rs2[4:0]<br>Note: Logical shift right. |
| **p.subN** | **rD, rs1, rs2, Is3** | rD = (rs1 - rs2) >>> Is3 Note: Arithmetic shift right. |

*Table continues on the next page...*

**Table 3-7. General ALU operations (continued)**

| Mnemonic | | Description |
|---|---|---|
| **p.subuN** | **rD, rs1, rs2, Is3** | rD = (rs1 - rs2) >> Is3 Note: Logical shift right. |
| **p.subRN** | **rD, rs1, rs2, Is3** | rD = (rs1 - rs2 + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right. |
| **p.subuRN** | **rD, rs1, rs2, Is3** | rD = (rs1 - rs2 + 2^(Is3-1))) >> Is3<br>Note: Logical shift right. |
| **p.subNr** | **rD, rs1, rs2** | rD = (rD – rs1) >>> rs2[4:0]<br>Note: Arithmetic shift right. |
| **p.subuNr** | **rD, rs1, rs2** | rD = (rD – rs1) >> rs2[4:0]<br>Note: Logical shift right. |
| **p.subRNr** | **rD, rs1, rs2** | rD = (rD – rs1+ 2^(rs2[4:0]-1)) >>> rs2[4:0]<br>Note: Arithmetic shift right. |
| **p.subuRNr** | **rD, rs1, rs2** | rD = (rD – rs1+ 2^(rs2[4:0]-1))) >> rs2[4:0]<br>Note: Logical shift right. |

#### 3.1.3.9.3.3 Immediate branching operations

**Table 3-8. Immediate branching operations**

| Mnemonic | Description |
|---|---|
| **p.beqimm rs1, Imm5, Imm12** | Branch to PC + (Imm12 << 1) if rs1 is equal to Imm5. Imm5 is signed. |
| **p.bneimm rs1, Imm5, Imm12** | Branch to PC + (Imm12 << 1) if rs1 is not equal to Imm5.<br>Imm5 is signed. |

### 3.1.3.9.4 Multiply-Accumulate

RI5CY uses a single-cycle 32-bit x 32-bit multiplier with a 32-bit result. All instructions of the RISC-V M instruction set extension are supported.

The multiplications with upper-word result (MSP of 32-bit x 32-bit multiplication), take 4 cycles to compute. The division and remainder instructions take between 2 and 32 cycles. The number of cycles depends on the operand values.

Additionally, RI5CY supports non-standard extensions for multiply-accumulate and half-word multiplications with an optional post-multiplication shift.

### 3.1.3.9.4.1 MAC operations

**Table 3-9. MAC operations**

| Mnemonic | | Description |
|---|---|---|
| 32-Bit x 32-Bit Multiplication Operations | | |
| **p.mac** | **rD, rs1, rs2** | rD = rD + rs1 * rs2 |
| **p.msu** | **rD, rs1, rs2** | rD = rD - rs1 * rs2 |
| 16-Bit x 16-Bit Multiplication | | |
| **p.muls** | **rD, rs1, rs2** | rD[31:0] = Sext(rs1[15:0]) * Sext(rs2[15:0]) |
| **p.mulhhs** | **rD, rs1, rs2** | rD[31:0] = Sext(rs1[31:15]) * Sext(rs2[31:15]) |
| **p.mulsN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[15:0]) * Sext(rs2[15:0])) >>> Is3<br>Note: Arithmetic shift right |
| **p.mulhhsN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[31:15]) * Sext(rs2[31:15])) >>> Is3<br>Note: Arithmetic shift right |
| **p.mulsRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[15:0]) * Sext(rs2[15:0]) + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right |
| **p.mulhhsRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[31:15]) * Sext(rs2[31:15]) + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right |
| **p.mulu** | **rD, rs1, rs2** | rD[31:0] = Zext(rs1[15:0]) * Zext(rs2[15:0]) |
| **p.mulhhu** | **rD, rs1, rs2** | rD[31:0] = Zext(rs1[31:15]) * Zext(rs2[31:15]) |
| **p.muluN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[15:0]) * Zext(rs2[15:0])) >>> Is3<br>Note: Logical shift right |
| **p.mulhhuN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[31:15]) * Zext(rs2[31:15])) >>> Is3<br>Note: Logical shift right |
| **p.muluRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[15:0]) * Zext(rs2[15:0]) + 2^(Is3-1)) >>> Is3<br>Note: Logical shift right |
| **p.mulhhuRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[31:15]) * Zext(rs2[31:15]) + 2^(Is3-1)) >>> Is3<br>Note: Logical shift right |
| 16-Bit x 16-Bit Multiply-Accumulate | | |
| **p.macsN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[15:0]) * Sext(rs2[15:0]) + rD) >>> Is3<br>Note: Arithmetic shift right |
| **p.machhsN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[31:15]) * Sext(rs2[31:15]) + rD) >>> Is3<br>Note: Arithmetic shift right |
| **p.macsRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[15:0]) * Sext(rs2[15:0]) + rD + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right |
| **p.machhsRN** | **, rD, rs1, rs2, Is3** | rD[31:0] = (Sext(rs1[31:15]) * Sext(rs2[31:15]) + rD + 2^(Is3-1)) >>> Is3<br>Note: Arithmetic shift right |
| **p.macuN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[15:0]) * Zext(rs2[15:0]) + rD) >>> Is3<br>Note: Logical shift right |
| **p.machhuN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[31:15]) * Zext(rs2[31:15]) + rD) >>> Is3<br>Note: Logical shift right |
| **p.macuRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[15:0]) * Zext(rs2[15:0]) + rD + 2^(Is3-1)) >>> Is3 |

*Table continues on the next page...*

**Table 3-9. MAC operations (continued)**

| Mnemonic | | Description |
|---|---|---|
| | | Note: Logical shift right |
| **p.machhuRN** | **rD, rs1, rs2, Is3** | rD[31:0] = (Zext(rs1[31:15]) * Zext(rs2[31:15]) + rD + 2^(Is3-1)) >>> Is3<br>Note: Logical shift right |

### 3.1.3.9.5 Vectorial

Vectorial instructions perform operations in a SIMD-like manner on multiple sub-word elements at the same time. This is done by segmenting the data path into smaller parts when 8 or 16-bit operations should be performed.

Vectorial instructions are available in two flavors:

- 8-Bit, to perform four operations on the 4 bytes inside a 32-bit word at the same time
- 16-Bit, to perform two operations on the 2 half-words inside a 32-bit word at the same time

Additionally, there are three modes that influence the second operand:

1. Normal mode, vector-vector operation. Both operands, from rs1 and rs2, are treated as vectors of bytes or half-words.
2. Scalar replication mode (.sc), vector-scalar operation. Operand 1 is treated as a vector, while operand 2 is treated as a scalar and replicated two or four times to form a complete vector. The LSP is used for this purpose.
3. Immediate scalar replication mode (.sci), vector-scalar operation. Operand 1 is treated as vector, while operand 2 is treated as a scalar and comes from an immediate. The immediate is either sign- or zero-extended, depending on the operation. If not specified, the immediate is sign-extended.

#### 3.1.3.9.5.1 Vectorial ALU operations

**Table 3-10. Vectorial ALU operations**

| Mnemonic | Description |
|---|---|
| General ALU Instructions | |
| **pv.add[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] + op2[i] |
| **pv.sub[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] - op2[i] |
| **pv.avg[.sc,.sci]{.h,.b}** | rD[i] = (rs1[i] + op2[i]) >> 1 Note: Arithmetic right shift |
| **pv.avgu[.sc,.sci]{.h,.b}** | rD[i] = (rs1[i] + op2[i]) >> 1 |
| **pv.min[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] < op2[i] ? rs1[i] : op2[i] |
| **pv.minu[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] < op2[i] ? rs1[i] : op2[i]<br>Note: Immediate is zero-extended, comparison is unsigned |

*Table continues on the next page...*

## Table 3-10. Vectorial ALU operations (continued)

| Mnemonic | Description |
|---|---|
| **pv.max[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] > op2[i] ? rs1[i] : op2[i] |
| **pv.maxu[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] > op2[i] ? rs1[i] : op2[i]<br>Note: Immediate is zero-extended, comparison is unsigned |
| **pv.srl[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] >> op2[i]<br>Note: Immediate is zero-extended, shift is logical |
| **pv.sra[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] >>> op2[i]<br>Note: Immediate is zero-extended, shift is arithmetic |
| **pv.sll[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] << op2[i]<br>Note: Immediate is zero-extended, shift is logical |
| **pv.or[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] \| op2[i] |
| **pv.xor[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] ^ op2[i] |
| **pv.and[.sc,.sci]{.h,.b}** | rD[i] = rs1[i] & op2[i] |
| **pv.abs{.h,.b}** | rD[i] = rs1 < 0 ? –rs1 : rs1 |
| **pv.extract.h** | rD = Sext(rs1[((I+1)*16)-1 : I*16]) |
| **pv.extract.b** | rD = Sext(rs1[((I+1)*8)-1 : I*8]) |
| **pv.extractu.h** | rD = Zext(rs1[((I+1)*16)-1 : I*16]) |
| **pv.extractu.b** | rD = Zext(rs1[((I+1)*8)-1 : I*8]) |
| **pv.insert.h** | rD[((I+1)*16-1:I*16] = rs1[15:0]<br>Note: The rest of the bits of rD are untouched and keep their previous value |
| **pv.insert,b** | rD[((I+1)*8-1:I*8] = rs1[7:0]<br>Note: The rest of the bits of rD are untouched and keep their previous value |
| Dot Product Instructions | |
| **pv.dotup[.sc,.sci].h** | rD = rs1[0] * op2[0] + rs1[1] * op2[1] Note: All operations are unsigned |
| **pv.dotup[.sc,.sci].b** | rD = rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: All operations are unsigned |
| **pv.dotusp[.sc,.sci].h** | rD = rs1[0] * op2[0] + rs1[1] * op2[1]<br>Note: rs1 is treated as unsigned, while rs2 is treated as signed |
| **pv.dotusp[.sc,.sci].b** | rD = rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: rs1 is treated as unsigned, while rs2 is treated as signed |
| **pv.dotsp[.sc,.sci].h** | rD = rs1[0] * op2[0] + rs1[1] * op2[1] Note: All operations are signed |
| **pv.dotsp[.sc,.sci].b** | rD = rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: All operations are signed |
| **pv.sdotup[.sc,.sci].h** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1] Note: All operations are unsigned |
| **pv.sdotup[.sc,.sci].b** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: All operations are unsigned |
| **pv.sdotusp[.sc,.sci].h** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1]<br>Note: rs1 is treated as unsigned, while rs2 is treated as signed |
| **pv.sdotusp[.sc,.sci].b** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: rs1 is treated as unsigned, while rs2 is treated as signed |
| **pv.sdotsp[.sc,.sci].h** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1] Note: All operations are signed |

*Table continues on the next page...*

## Table 3-10. Vectorial ALU operations (continued)

| Mnemonic | Description |
|---|---|
| **pv.sdotsp[.sc,.sci].b** | rD = rD + rs1[0] * op2[0] + rs1[1] * op2[1] + rs1[2] * op2[2] + rs1[3] * op2[3] Note: All operations are signed |
| Shuffle and Pack Instructions | |
| **pv.shuffle.h** | rD[31:16] = rs1[rs2[16]*16+15:rs2[16]*16] rD[15:0] = rs1[rs2[0]*16+15:rs2[0]*16] |
| **pv.shuffle.sci.h** | rD[31:16] = rs1[I1*16+15:I1*16] rD[15:0] = rs1[I0*16+15:I0*16]<br>Note: I1 and I0 represent bits 1 and 0 of the immediate |
| **pv.shuffle.b** | rD[31:24] = rs1[rs2[25:24]*8+7:rs2[25:24]*8] rD[23:16] = rs1[rs2[17:16]*8+7:rs2[17:16]*8] rD[15:8] = rs1[rs2[9:8]*8+7:rs2[9:8]*8]<br>rD[7:0] = rs1[rs2[1:0]*8+7:rs2[1:0]*8] |
| **pv.shufflel0.sci.b** | rD[31:24] = rs1[7:0]<br>rD[23:16] = rs1[(I5:I4)*8+7: (I5:I4)*8]<br>rD[15:8] = rs1[(I3:I2)*8+7: (I3:I2)*8]<br>rD[7:0] = rs1[(I1:I0)*8+7:(I1:I0)*8] |
| **pv.shufflel1.sci.b** | rD[31:24] = rs1[15:8]<br>rD[23:16] = rs1[(I5:I4)*8+7: (I5:I4)*8]<br>rD[15:8] = rs1[(I3:I2)*8+7: (I3:I2)*8]<br>rD[7:0] = rs1[(I1:I0)*8+7:(I1:I0)*8] |
| **pv.shufflel2.sci.b** | rD[31:24] = rs1[23:16]<br>rD[23:16] = rs1[(I5:I4)*8+7: (I5:I4)*8]<br>rD[15:8] = rs1[(I3:I2)*8+7: (I3:I2)*8]<br>rD[7:0] = rs1[(I1:I0)*8+7:(I1:I0)*8] |
| **pv.shufflel3.sci.b** | rD[31:24] = rs1[31:24]<br>rD[23:16] = rs1[(I5:I4)*8+7: (I5:I4)*8]<br>rD[15:8] = rs1[(I3:I2)*8+7: (I3:I2)*8]<br>rD[7:0] = rs1[(I1:I0)*8+7:(I1:I0)*8] |
| **pv.shuffle2.h** | rD[31:16] = ((rs2[17] == 1) ? rs1 : rD)[rs2[16]*16+15:rs2[16]*16]<br>rD[15:0] = ((rs2[1] == 1) ? rs1 : rD)[rs2[0]*16+15:rs2[0]*16] |
| **pv.shuffle2.b** | rD[31:24] = ((rs2[26] == 1) ? rs1 : rD)[rs2[25:24]*8+7:rs2[25:24]*8]<br>rD[23:16] = ((rs2[18] == 1) ? rs1 : rD)[rs2[17:16]*8+7:rs2[17:16]*8]<br>rD[15:8] = ((rs2[10] == 1) ? rs1 : rD)[rs2[9:8]*8+7:rs2[9:8]*8]<br>rD[7:0] = ((rs2[2] == 1) ? rs1 : rD)[rs2[1:0]*8+7:rs2[1:0]*8] |
| **pv.pack.h** | rD[31:16] = rs1[15:0]<br>rD[15:0] = rs2[15:0] |
| **pv.packhi.b** | rD[31:24] = rs1[7:0]<br>rD[23:16] = rs2[7:0]<br>Note: The rest of the bits of rD are untouched and keep their previous value |
| **pv.packlo.b** | rD[15:8] = rs1[7:0]<br>rD[7:0] = rs2[7:0]<br>Note: The rest of the bits of rD are untouched and keep their previous value |

#### 3.1.3.9.5.2 Vectorial comparison operations

Vectorial comparisons are done on individual bytes (.b) or half-words (.h), depending on the chosen mode. If the comparison result is true, all bits in the corresponding byte/half-word are set to 1. If the comparison result is false, all bits are set to 0.

The default mode (no .sc, .sci) compares the lowest byte/half-word of the first operand with the lowest byte/half-word of the second operand, and so on. If the mode is set to scalar replication (.sc), always the lowest byte/half-word of the second operand is used for comparisons, thus instead of a vector comparison a scalar comparison is performed. In the immediate scalar replication mode (.sci), the immediate given to the instruction is used for the comparison.

**Table 3-11. Vectorial comparison operations**

| Mnemonic | | Description |
|---|---|---|
| **pv.cmpeq[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] == op2 ? ‘1 : ‘0 |
| **pv.cmpne[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] != op2 ? ‘1 : ‘0 |
| **pv.cmpgt[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] > op2 ? ‘1 : ‘0 |
| **pv.cmpge[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] >=op2 ? ‘1 : ‘0 |
| **pv.cmplt[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] < op2 ? ‘1 : ‘0 |
| **pv.cmple[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] <= op2 ? ‘1 : ‘0 |
| **pv.cmpgtu[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] > op2 ? ‘1 : ‘0 Note: Unsigned comparison |
| **pv.cmpgeu[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] >= op2 ? ‘1 : ‘0 Note: Unsigned comparison |
| **pv.cmpltu[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] < op2 ? ‘1 : ‘0 Note: Unsigned comparison |
| **pv.cmpleu[.sc,.sci]{.h,.b}** | **rD, rs1, {rs2, Imm6}** | rD[i] = rs1[i] <= op2 ? ‘1 : ‘0 Note: Unsigned comparison |

## 3.1.4 RISC-V ZERO-RISCY core

### 3.1.4.1 Introduction

ZERO-RISCY is a 2-stage in-order 32b RISC-V processor core. ZERO-RISCY has been designed to be small and efficient.

The following figure shows a block diagram of the core.

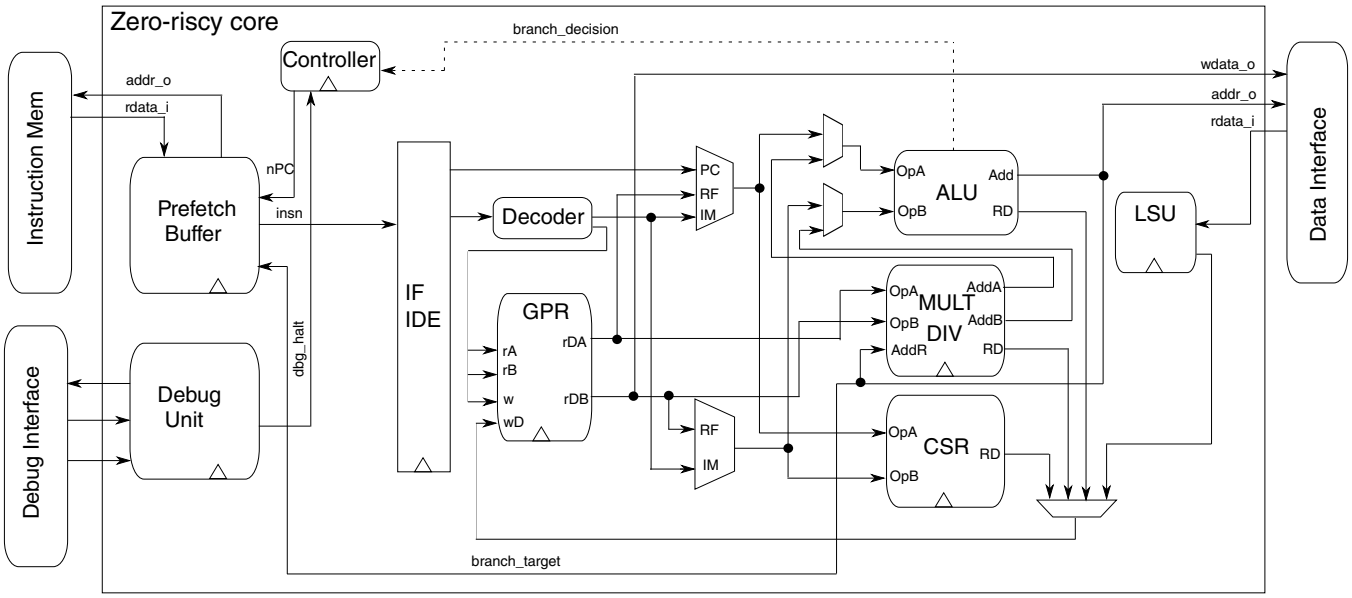



**Figure 3-3. Block diagram**

ZERO-RISCY supports the following instructions:

- Full support for RV32I Base Integer Instruction Set
- Full support for RV32E Base Integer Instruction Set
- Full support for RV32C Standard Extension for Compressed Instructions
- Full support for RV32M Integer Multiplication and Division Instruction Set Extension

## 3.1.4.2 ZERO-RISCY memory map and register descriptions

Performance Counters in ZERO-RISCY are placed inside the Control and Status Registers and can be accessed with csrr and csrw instructions.

**ZERO_RISCY memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 300 | Machine Status (ZERO_RISCY_MSTATUS) | 32 | R/W | 0000_1800h | 3.1.4.2.1/ 98 |
| 305 | Machine Trap-Vector Base Address (ZERO_RISCY_MTVEC) | 32 | R | 0103_FF00h | 3.1.4.2.2/ 99 |
| 341 | Machine Exception PC (ZERO_RISCY_MEPC) | 32 | R/W | 0000_0000h | 3.1.4.2.3/ 100 |
| 342 | Machine Cause (ZERO_RISCY_MCAUSE) | 32 | R/W | 0000_0000h | 3.1.4.2.4/ 100 |
| 780 | Performance Counter Counter Register n (ZERO_RISCY_PCCR0) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |

*Table continues on the next page...*

**ZERO_RISCY memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 781 | Performance Counter Counter Register n (ZERO_RISCY_PCCR1) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 782 | Performance Counter Counter Register n (ZERO_RISCY_PCCR2) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 783 | Performance Counter Counter Register n (ZERO_RISCY_PCCR3) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 784 | Performance Counter Counter Register n (ZERO_RISCY_PCCR4) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 785 | Performance Counter Counter Register n (ZERO_RISCY_PCCR5) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 786 | Performance Counter Counter Register n (ZERO_RISCY_PCCR6) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 787 | Performance Counter Counter Register n (ZERO_RISCY_PCCR7) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 788 | Performance Counter Counter Register n (ZERO_RISCY_PCCR8) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 789 | Performance Counter Counter Register n (ZERO_RISCY_PCCR9) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 78A | Performance Counter Counter Register n (ZERO_RISCY_PCCR10) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 78B | Performance Counter Counter Register n (ZERO_RISCY_PCCR) | 32 | R/W | 0000_0000h | 3.1.4.2.5/ 101 |
| 7A0 | Performance Counter Event Register (ZERO_RISCY_PCER) | 32 | R/W | 0000_0000h | 3.1.4.2.6/ 103 |
| 7A1 | Performance Counter Mode Register (ZERO_RISCY_PCMR) | 32 | R/W | 0000_0003h | 3.1.4.2.7/ 104 |

## 3.1.4.2.1 Machine Status (ZERO_RISCY_MSTATUS)

Address: 0h base + 300h offset = 300h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | MPP | | 0 | | | MPIE | 0 | | | MIE | 0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ZERO_RISCY_MSTATUS field descriptions**

| Field | Description |
|---|---|
| 31–13<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 12–11<br>MPP | Previous Privilege<br><br>ZERO-RISCY core only implements machine mode. So statically 2'b11 and cannot be altered. |
| 10–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>MPIE | Previous Interrupt Enable<br><br>When an exception is encountered, MPIE will be set to MIE. When the mret instruction is executed, the value of MPIE will be stored to MIE.<br><br>0 Disable previous interrupt<br>1 Enable previous interrupt |
| 6–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>MIE | Interrupt Enable<br><br>If you want to enable interrupt handling in your exception handler, set the Interrupt Enable to 1'b1 inside your handler code.<br><br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 3.1.4.2.2 Machine Trap-Vector Base Address (ZERO_RISCY_MTVEC)

Address: 0h base + 305h offset = 305h

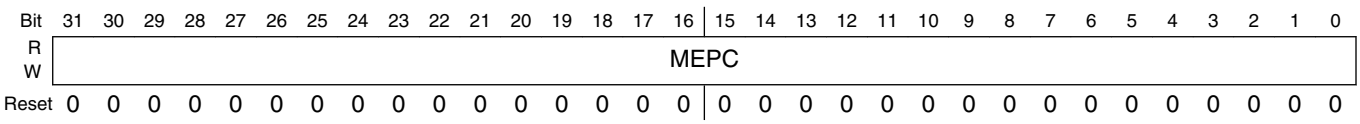
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MTVEC | | | | | | | | | | | | | | | | | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ZERO_RISCY_MTVEC field descriptions**

| Field | Description |
|---|---|
| 31–8<br>MTVEC | MTVEC<br><br>When an exception is encountered, the core jumps to the corresponding handler using the content of the MTVEC as base address. It is a read-only register which contains the boot address. |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

## 3.1.4.2.3 Machine Exception PC (ZERO_RISCY_MEPC)

Address: 0h base + 341h offset = 341h

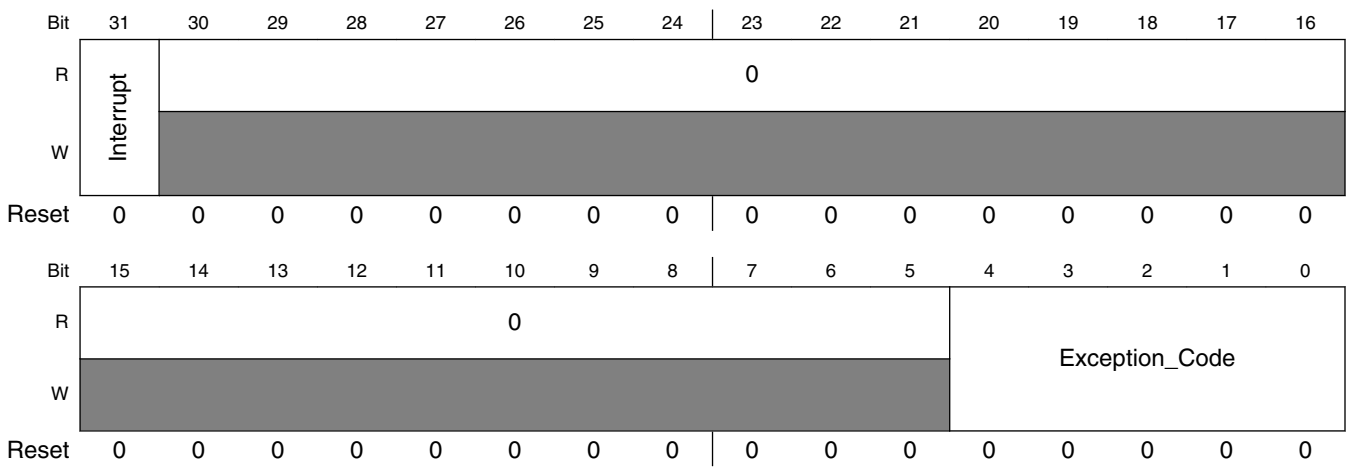



### ZERO_RISCY_MEPC field descriptions

| Field | Description |
|---|---|
| MEPC | Machine Exception PC<br><br>When an exception is encountered, the current program counter is saved in MEPC, and the core jumps to the exception address. When an mret instruction is executed, the value from MEPC replaces the current program counter. |

## 3.1.4.2.4 Machine Cause (ZERO_RISCY_MCAUSE)

Address: 0h base + 342h offset = 342h




### ZERO_RISCY_MCAUSE field descriptions

| Field | Description |
|---|---|
| 31<br>Interrupt | Interrupt<br><br>This bit is set when the exception was triggered by an interrupt.<br><br>0 Core in debug mode is caused by core exceptions.<br>1 Core in debug mode is caused by interrupts. . |
| 30–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| Exception_Code | Exception Code<br><br>When exception is triggered by an interrupt, this field is the number of interrupt from 0-31. When it's a core exception, this field is the number of exception. |

*Table continues on the next page...*

**ZERO_RISCY_MCAUSE field descriptions (continued)**

| Field | Description |
|---|---|
| | 00010 Illegal instruction<br>00011 EBREAK<br>00101 LSU load fault<br>00111 LSU store fault<br>01011 ECALL |

### 3.1.4.2.5 Performance Counter Counter Register n (ZERO_RISCY_PCCR*n*)

PCCR registers support both saturating and wrap-around arithmetic. This is controlled by the saturation bit in PCMR.

**Table 3-12. PCCR registers**

| Register | Name | Description |
|---|---|---|
| PCCR0 | CYCLES | Counts the number of cycles the core is active (not sleeping) |
| PCCR1 | INSTR | Counts the number of instructions executed |
| PCCR2 | - | Reserved |
| PCCR3 | - | Reserved |
| PCCR4 | IMISS | Cycles waiting for instruction fetches, i.e. number of instructions wasted due to non-ideal caching |
| PCCR5 | LD | Number of data memory loads executed. Misaligned accesses are counted twice |
| PCCR6 | ST | Number of data memory stores executed. Misaligned accesses are counted twice |
| PCCR7 | JUMP | Number of unconditional jumps (j, jal, jr, jalr) |
| PCCR8 | BRANCH | Number of branches.<br>Counts taken and not taken branches |
| PCCR9 | BTAKEN | Number of taken branches. |
| PCCR10 | RVC | Number of compressed instructions executed |

In the FPGA, RTL simulation and Virtual-Platform there are individual counters for each event type, i.e. PCCR0-10 each represent a separate register. To save area in the ASIC, there is only one counter and one counter register. Accessing PCCR0-10 will access the same counter register in the ASIC.

The following figure shows how events are first masked with the PCER register and then ORed together to increase the one performance counter PCCR.

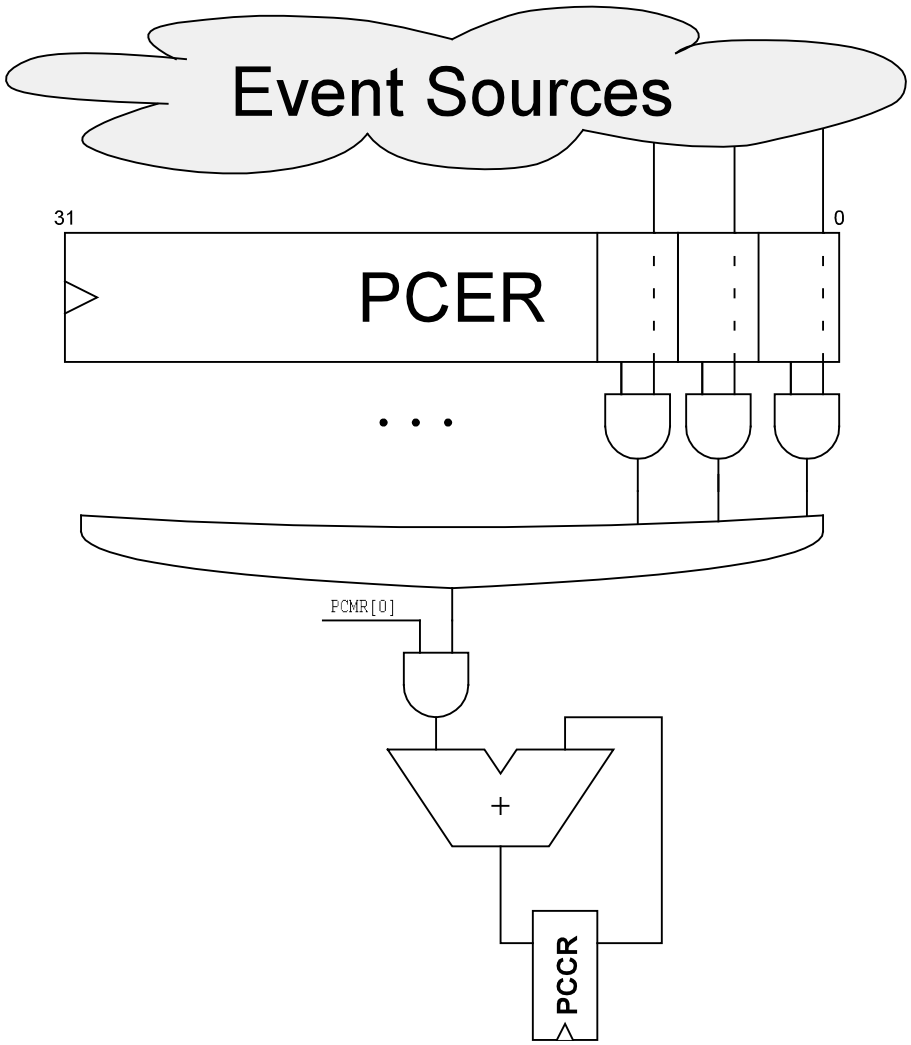



**Figure 3-4. Events and PCCR, PCMR and PCER on the ASIC**

Address: 0h base + 780h offset + (1d × i), where i=0d to 31d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Counter | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## ZERO_RISCY_PCCR*n* field descriptions

| Field | Description |
|---|---|
| Counter | Counter<br><br>See register descriptions for details. |

### 3.1.4.2.6 Performance Counter Event Register (ZERO_RISCY_PCER)

Each bit in the PCER register controls one performance counter. If the bit is 1, the counter is enabled and starts counting events. If it is 0, the counter is disabled and its value won't change.

In the ASIC there is only one counter register, thus all counter events are masked by PCER and ORed together, i.e. if one of the enabled event happens, the counter will be increased. If multiple non-masked events happen at the same time, the counter will only be increased by one.

In order to be able to count separate events on the ASIC, the program can be executed in a loop with different events configured.

In the FPGA or RTL simulation version, each event has its own counter and can be accessed separately.

See PCCR Registers table for details of each bit field in this register.

Address: 0h base + 7A0h offset = 7A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | RVC | BTAKEN | BRANCH | JUMP | ST | LD | IMISS | Reserved | | INSTR | CYCLES |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ZERO_RISCY_PCER field descriptions**

| Field | Description |
|---|---|
| 31–11<br>Reserved | This field is reserved. |
| 10<br>RVC | RVC<br><br>It counts the number of compressed instructions executed. |
| 9<br>BTAKEN | WBRANCH<br><br>It counts the number of taken branches. |
| 8<br>BRANCH | BRANCH<br><br>It counts the number of branches, no matter taken or not taken. |

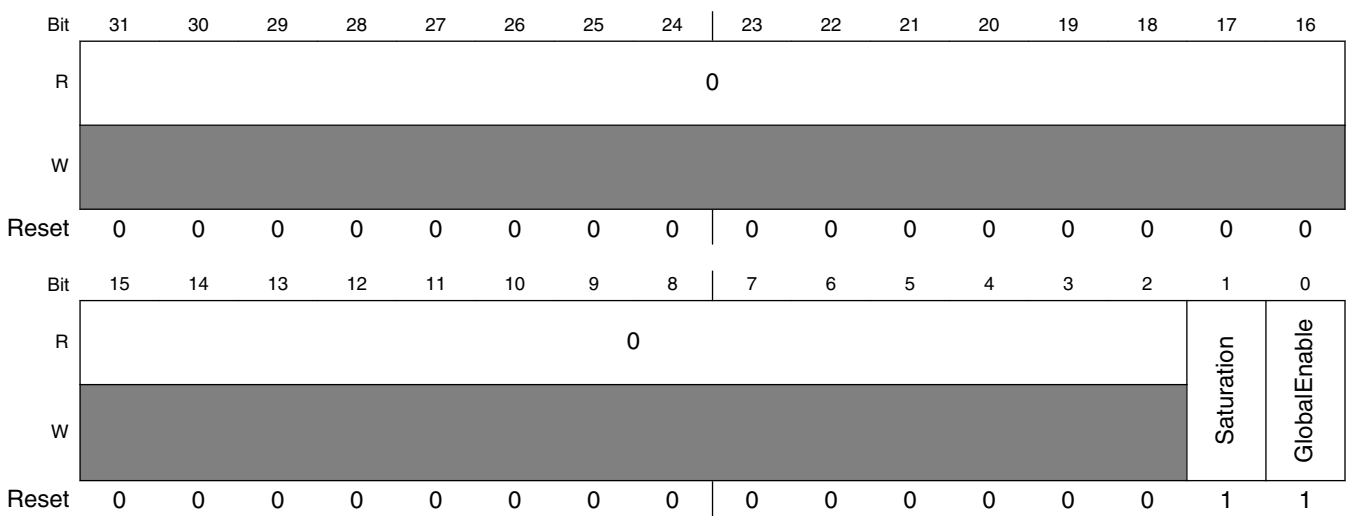
*Table continues on the next page...*

**ZERO_RISCY_PCER field descriptions (continued)**

| Field | Description |
|---|---|
| 7<br>JUMP | JUMP<br><br>It counts the number of unconditional jumps (j, jal, jr,jalr) |
| 6<br>ST | ST<br><br>It counts the number of data memory stores executed. Misaligned accesses are counted twice |
| 5<br>LD | LD<br><br>It counts the number of data memory loads executed. Misaligned accesses are counted twice |
| 4<br>IMISS | IMISS<br><br>It counts the cycles waiting for instruction fetches, i.e.the number of instructions wasted due to non-ideal caching |
| 3–2<br>Reserved | This field is reserved. |
| 1<br>INSTR | INSTR<br><br>It counts the number of instructions executed. |
| 0<br>CYCLES | CYCLES<br><br>It counts the number of cycles that the core is active (not sleeping) |

## 3.1.4.2.7 Performance Counter Mode Register (ZERO_RISCY_PCMR)

Address: 0h base + 7A1h offset = 7A1h




**ZERO_RISCY_PCMR field descriptions**

| Field | Description |
|---|---|
| 31–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**ZERO_RISCY_PCMR field descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>Saturation | Saturation<br><br>If this bit is set, saturating arithmetic is used in the performance counter counters. After reset, this bit is set. |
| 0<br>GlobalEnable | Global Enable<br><br>Activate/deactivate all performance counters. If this bit is 0, all performance counters are disabled. After reset, this bit is set. |

### 3.1.4.3 Exceptions and Interrupts

ZERO-RISCY supports interrupts, exceptions on illegal instructions.

**Table 3-13. Interrupt/Exception Offset Vector Table**

| Address | Description |
|---|---|
| 0x00-0x7C | Interrupts 0 – 31 |
| 0x80 | Reset |
| 0x84 | Illegal Instruction |
| 0x88 | ECALL Instruction Executed |
| 0x8C | LSU Error (Invalid Memory Access) |

The base address of the interrupt vector table is given by the boot address. The most significant 3 bytes of the boot address given to the core are used for the first instruction fetch of the core and as the basis of the interrupt vector table. The core starts fetching at the address made by concatenating the most significant 3 bytes of the boot address and the reset value (0x80) as the least significant byte. The boot address can be changed after the first instruction was fetched to change the interrupt vector table address. It is assumed that the boot address is supplied via a register to avoid long paths to the instruction fetch unit.

#### 3.1.4.3.1 Interrupts

Interrupts can only be enabled/disabled on a global basis and not individually. It is assumed that there is an event/interrupt controller outside of the core that performs masking and buffering of the interrupt lines. The global interrupt enable is done via the CSR register MSTATUS.

Multiple interrupts requests are assumed to be handled by event/interrupt controller. When an interrupt is taken, the core gives an acknowledge signal to the event/interrupt controller as well as the interrupt id taken.

#### 3.1.4.3.2 Exceptions

The illegal instruction exception and ecall instruction exceptions cannot be disabled and are always active.

#### 3.1.4.3.3 Handling

ZERO-RISCY does support nested interrupt/exception handling. Exceptions inside interrupt/exception handlers cause another exception, thus exceptions during the critical part of your exception handlers, i.e. before having saved the MEPC and MESTATUS registers, will cause those register to be overwritten.

Interrupts during interrupt/exception handlers are disabled by default, but can be explicitly enabled if desired.

Upon executing an mret instruction, the core jumps to the program counter saved in the CSR register MEPC and restores the MPIE value of the register MSTATUS to IE. When entering an interrupt/exception handler, the core sets MEPC to the current program counter and saves the current value of MIE in MPIE of the MSTATUS register.

### 3.1.4.4 Debug Unit

#### 3.1.4.4.1 Address Map

**Table 3-14. Debug Unit Address Map**

| Address | Name | Description |
|---|---|---|
| 0x0000-0x007F | Debug Registers | Always accessible, even when the core is running |
| 0x400-0x47F | GPR (x0-x31) | General Purpose Registers<br>Only accessible if the core is halted |
| 0x2000-0x20FF | Debug Registers | Only accessible if the core is halted |
| 0x4000-0x7FFF | CSR | Control and Status Registers<br>Only accessible if the core is halted |

**NOTE**

Addresses are intended for a bus system with 32-bit wide words.

Addresses have to be aligned to word-boundaries.

### 3.1.4.4.2 ZERO-RISCY memory map and register descriptions

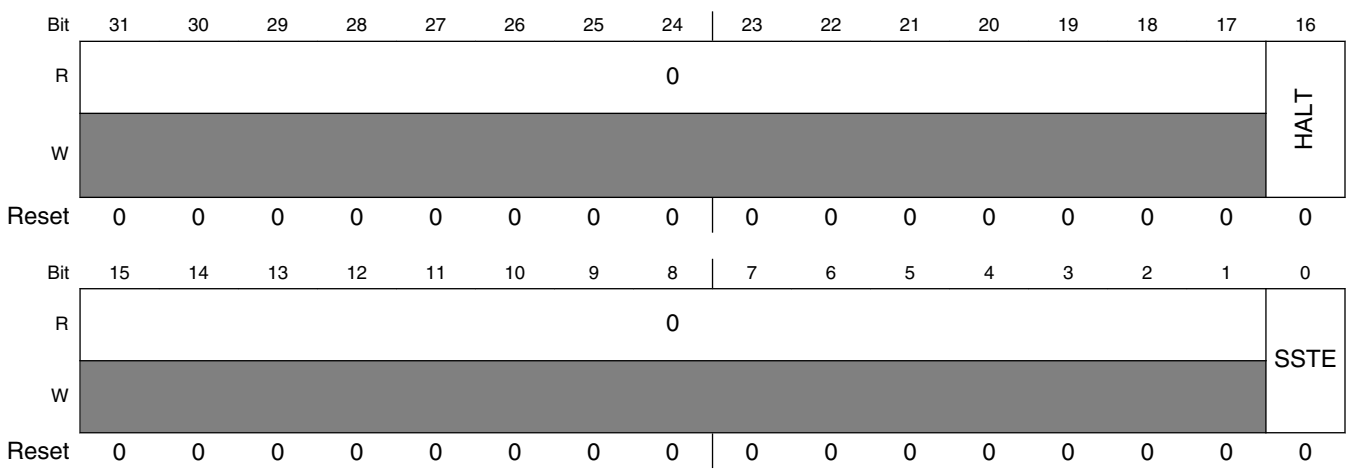
Performance Counters in ZERO-RISCY are placed inside the Control and Status Registers and can be accessed with csrr and csrw instructions.

**DBG memory map**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| F900_8000 | Debug Control (DBG_CTRL) | 32 | R/W | 0000_0000h | 3.1.4.4.2.1/ 107 |
| F900_8004 | Debug Hit (DBG_HIT) | 32 | R/W | 0000_0000h | 3.1.4.4.2.2/ 108 |
| F900_8008 | Debug Interrupt Enable (DBG_IE) | 32 | R/W | 0000_0000h | 3.1.4.4.2.3/ 109 |
| F900_800C | Debug Cause (DBG_CAUSE) | 32 | R | 0000_0000h | 3.1.4.4.2.4/ 110 |
| F900_8040 | Debug Hardware Breakpoint Control Register n (DBG_BPCTRL0) | 32 | R/W | 0000_0000h | 3.1.4.4.2.5/ 111 |
| F900_8044 | Debug Hardware Breakpoint Data Register n (DBG_BPDATA0) | 32 | R/W | 0000_0000h | 3.1.4.4.2.6/ 112 |
| F900_8048 | Debug Hardware Breakpoint Control Register n (DBG_BPCTRL1) | 32 | R/W | 0000_0000h | 3.1.4.4.2.5/ 111 |
| F900_804C | Debug Hardware Breakpoint Data Register n (DBG_BPDATA1) | 32 | R/W | 0000_0000h | 3.1.4.4.2.6/ 112 |
| F900_A000 | Debug Next Program Counter (DBG_NPC) | 32 | R/W | 0000_0000h | 3.1.4.4.2.7/ 112 |
| F900_A004 | Debug Previous Program Counter (DBG_PPC) | 32 | R | 0000_0000h | 3.1.4.4.2.8/ 113 |

### 3.1.4.4.2.1 Debug Control (DBG_CTRL)

Address: F900_8000h base + 0h offset = F900_8000h

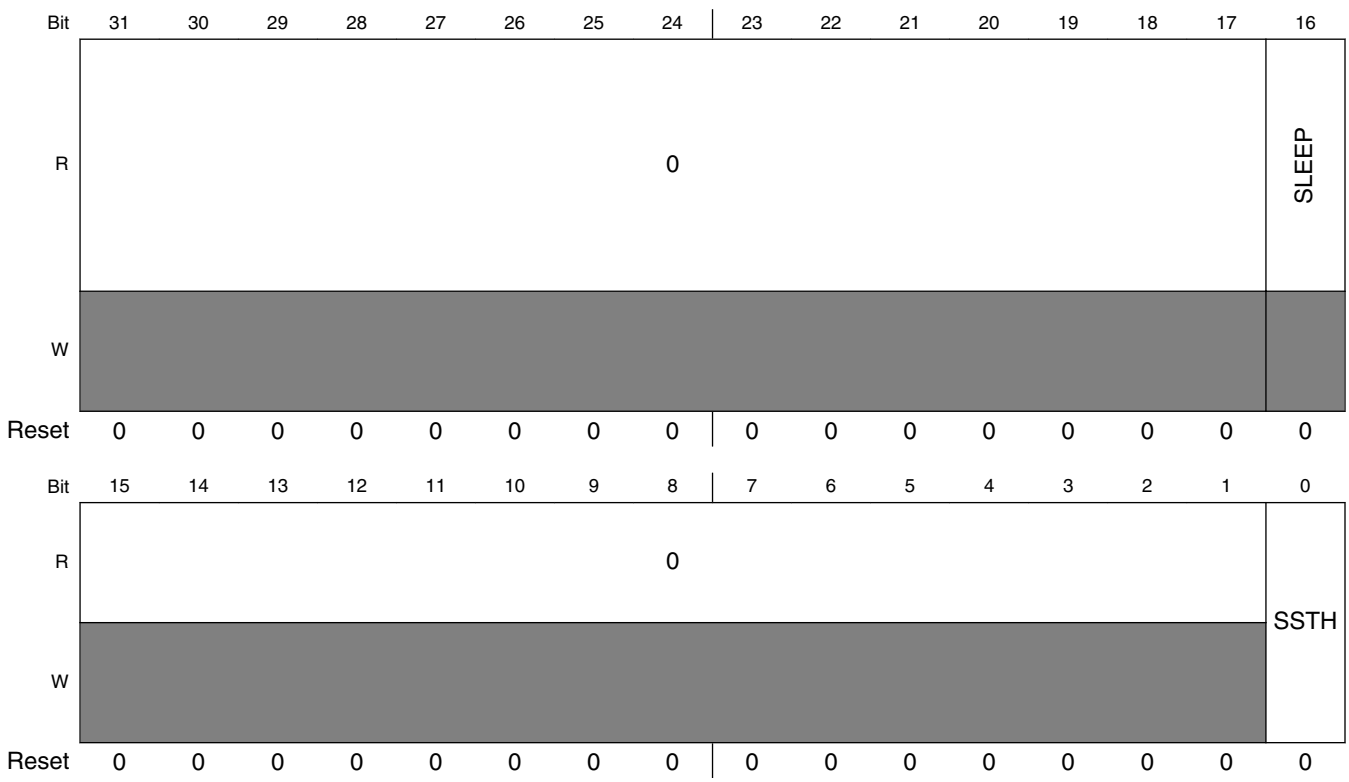


**DBG_CTRL field descriptions**

| Field | Description |
|---|---|
| 31–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>HALT | HALT<br><br>When 1 written, core enters debug mode, when 0 written, core exits debug mode. When read, 1 means core is in debug mode<br><br>0 Core is not in debug mode.<br>1 Core is in debug mode. |
| 15–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>SSTE | Single-step enable<br><br>When single-step is enabled, writing 0 to HALT will cause the core to step by one step.<br><br>0 Single-step is disabled.<br>1 Single-step is enabled. |

## 3.1.4.4.2.2 Debug Hit (DBG_HIT)

Address: F900_8000h base + 4h offset = F900_8004h



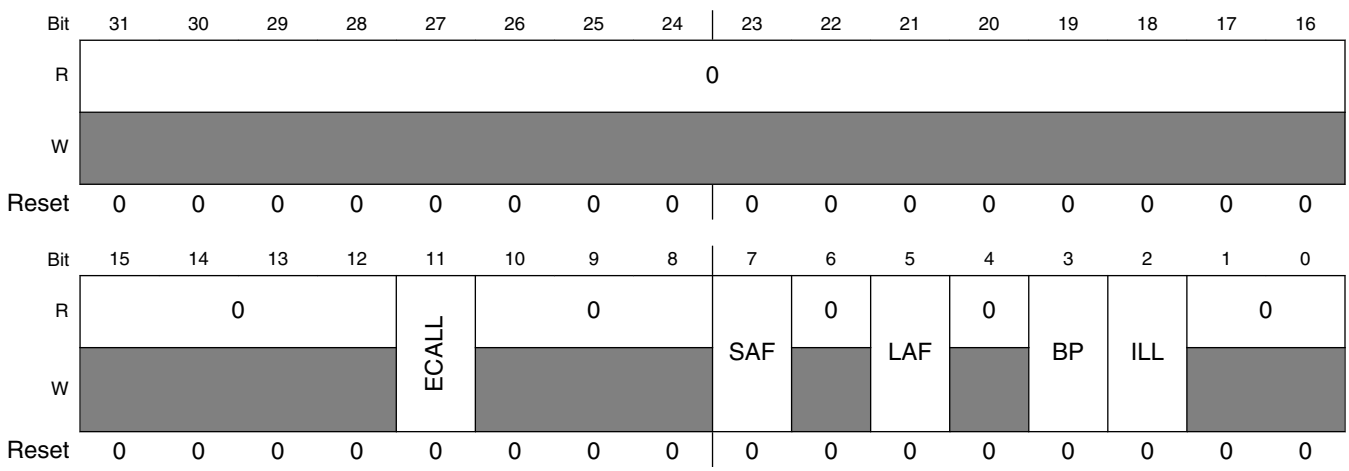
**DBG_HIT field descriptions**

| Field | Description |
|---|---|
| 31–17<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 16<br>SLEEP | SLEEP<br><br>Set when the core is in a sleeping state and waits for an event. |
| 15–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>SSTH | SSTH<br><br>Single-step hit, sticky bit that must be cleared by external debugger. |

### 3.1.4.4.2.3 Debug Interrupt Enable (DBG_IE)

This register provides the ability for interrupts to cause traps. When enabled, exceptions cause traps, otherwise normal exceptions.

Address: F900_8000h base + 8h offset = F900_8008h




**DBG_IE field descriptions**

| Field | Description |
|---|---|
| 31–12<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 11<br>ECALL | ECALL Enable<br><br>0 Environment call does not cause trap.<br>1 Environment call causes trap. |
| 10–8<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 7<br>SAF | SAF<br><br>When 1 is written to the LAF, this bit is also set. |

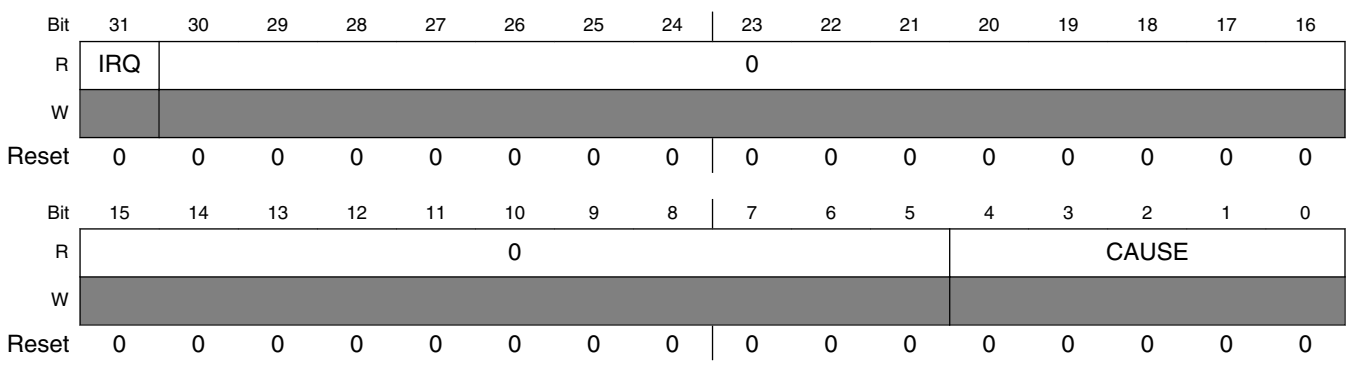
*Table continues on the next page...*

**DBG_IE field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Store access fault does not cause trap.<br>1 Store access fault causes trap. |
| 6<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 5<br>LAF | LAF<br><br>When 1 is written to the SAF, this bit is also set.<br><br>0 Load access fault does not cause trap.<br>1 Load access fault causes trap. |
| 4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 3<br>BP | BP<br><br>0 EBREAK instruction does not cause trap.<br>1 EBREAK instruction causes trap. |
| 2<br>ILL | ILL<br><br>0 Illegal instruction does not cause trap.<br>1 Illegal instruction causes trap. . |
| Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

### 3.1.4.4.2.4 Debug Cause (DBG_CAUSE)

Address: F900_8000h base + Ch offset = F900_800Ch




**DBG_CAUSE field descriptions**

| Field | Description |
|---|---|
| 31<br>IRQ | IRQ<br><br>0 Core in debug mode is caused by core exceptions.<br>1 Core in debug mode is caused by interrupts. . |
| 30–5<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

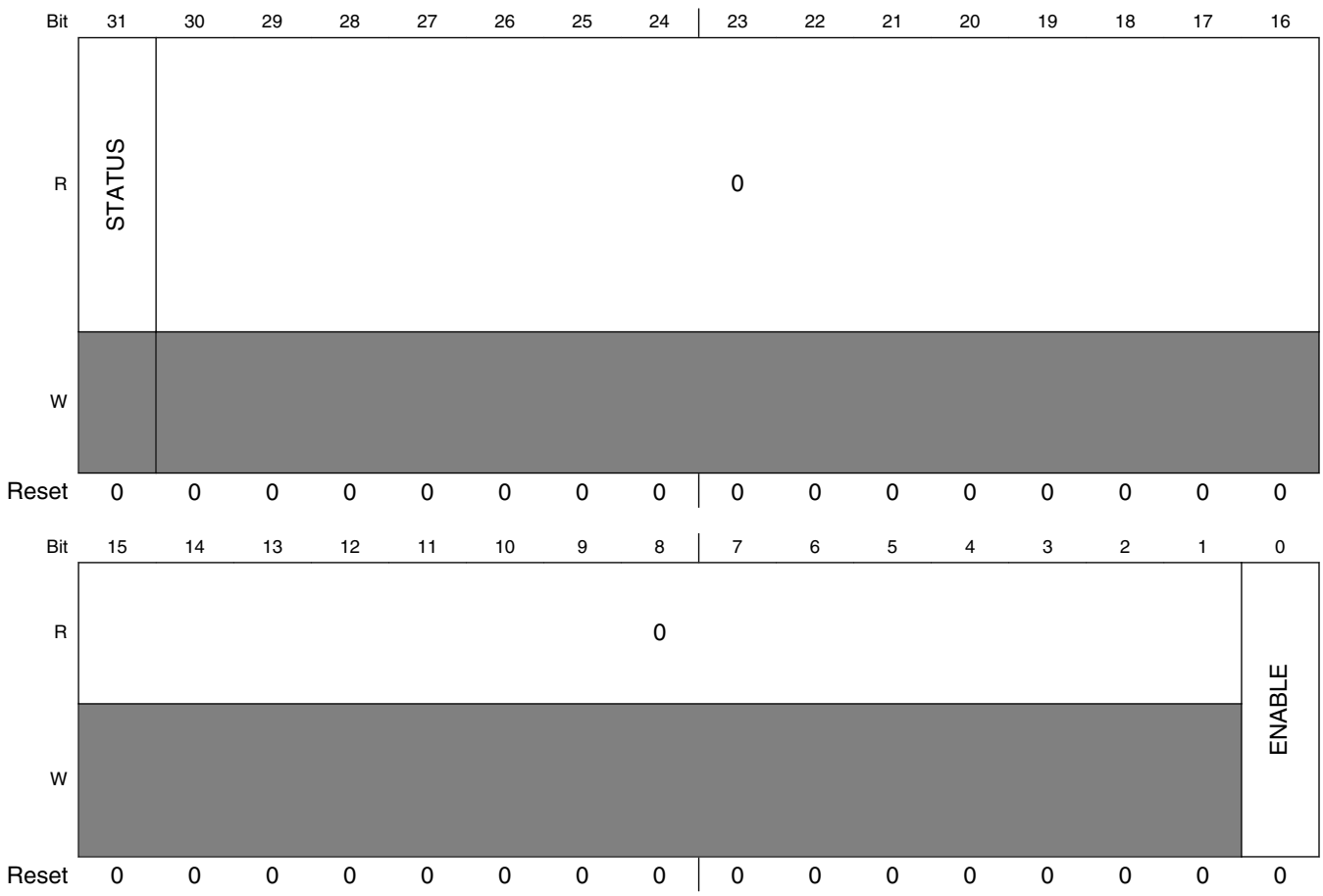
*Table continues on the next page...*

**DBG_CAUSE field descriptions (continued)**

| Field | Description |
|---|---|
| CAUSE | Exception/interrupt number<br><br>If IRQ is 1, this field is the number of interrupt that causes core to enter debug mode. If IRQ is 0, this field is the code of exception that causes core to enter debug mode.<br><br>00010 Illegal instruction<br>00011 EBREAK<br>00101 LSU load fault<br>00111 LSU store fault<br>01011 ECALL |

### 3.1.4.4.2.5 Debug Hardware Breakpoint Control Register n (DBG_BPCTRL*n*)

Address: F900_8000h base + 40h offset + (8d × i), where i=0d to 1d




**DBG_BPCTRL*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>STATUS | STATUS<br><br>High when break point hits. |

*Table continues on the next page...*

**DBG_BPCTRL*n* field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Breakpoint does not hit.<br>1 Breakpoint hits. |
| 30–1<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 0<br>ENABLE | Break point enable bit<br><br>When 1 is written and core's PC hits DBG_BPDATA, core enters debug mode.<br><br>0 Breakpoint is disabled.<br>1 Breakpoint is enabled. |

### 3.1.4.4.2.6 Debug Hardware Breakpoint Data Register n (DBG_BPDATA*n*)

Address: F900_8000h base + 44h offset + (8d × i), where i=0d to 1d

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PC | | | | | | | | | | | | | | | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DBG_BPDATA*n* field descriptions**

| Field | Description |
|---|---|
| 31–1<br>PC | PC<br><br>Set break point PC value. |
| 0<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

### 3.1.4.4.2.7 Debug Next Program Counter (DBG_NPC)

Values of PPC and NPC when entering debug mode:

| Reason | PPC | NPC | GDB Sigval |
|---|---|---|---|
| ebreak | ebreak instruction | next instruction | TRAP |
| ecall | ecall instruction | IVT entry | TRAP |
| illegal instruction | illegal instruction | IVT entry | ILL |
| invalid mem access | load/store instruction | IVT entry | SEGV |
| interrupt | last instruction | IVT entry | INT |
| halt | last instruction | next instruction | TRAP |
| single-step | last instruction | next instruction | TRAP |

Address: F900_8000h base + 2000h offset = F900_A000h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NPC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DBG_NPC field descriptions**

| Field | Description |
|---|---|
| NPC | NPC<br><br>Next PC to be executed. When written core jumps to PC. |

### 3.1.4.4.2.8 Debug Previous Program Counter (DBG_PPC)

Address: F900_8000h base + 2004h offset = F900_A004h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DBG_PPC field descriptions**

| Field | Description |
|---|---|
| PPC | PPC<br><br>Previous PC, already executed. |

### 3.1.4.4.3 Control and Status Registers

These registers can only be accessed when core is in debug mode.

| Address | Name | Description |
|---|---|---|
| 0x4000 | CSR 0 = 0x000 | CSR |
| ... | ... | ... |
| 0x7FFC | CSR 4095 = 0xFFF | CSR |

## 3.2 Nested Vectored Interrupt Controller (NVIC) modules

This section introduces the NVIC modules in this device.

## 3.2.1 Cortex-M0+ Nested Vectored Interrupt Controller (NVIC)

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by Arm and can be found at **arm.com**.

### 3.2.1.1 Cortex M0+ interrupt priority levels

This device supports 4 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 2 bits. For example, IPR0 is shown below:

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQ3 | | 0 | 0 | 0 | 0 | 0 | 0 | IRQ2 | | 0 | 0 | 0 | 0 | 0 | 0 | IRQ1 | | 0 | 0 | 0 | 0 | 0 | 0 | IRQ0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### 3.2.1.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external NMI_b signal. The pin the NMI_b signal is multiplexed on, must be configured for the Non-Maskable Interrupt function to generate the non-maskable interrupt request.

### 3.2.1.3 Cortex-M0+ interrupt channel assignments

The interrupt source assignments are defined in the following table.

- Vector number — the value stored on the stack when an interrupt is serviced.
- IRQ number — non-core interrupt source count, which is the vector number minus 16.

The IRQ number is used within ARM's NVIC documentation.

**Table 3-16. Interrupt vector assignments**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| **ARM Core System Handler Vectors** | | | | | | |
| 0x0000_0000 | 0 | – | – | – | ARM Cortex-M0+ | Initial Stack Pointer |
| 0x0000_0004 | 1 | – | – | – | ARM Cortex-M0+ | Initial Program Counter |
| 0x0000_0008 | 2 | – | – | – | ARM Cortex-M0+ | Non-Maskable Interrupt (NMI) |

*Table continues on the next page...*

## Table 3-16. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_000C | 3 | – | – | – | ARM Cortex-M0+ | Hard Fault |
| 0x0000_0010 | 4 | – | – | – | — | — |
| 0x0000_0014 | 5 | – | – | – | — | — |
| 0x0000_0018 | 6 | – | – | – | — | — |
| 0x0000_001C | 7 | – | – | – | — | — |
| 0x0000_0020 | 8 | – | – | – | — | — |
| 0x0000_0024 | 9 | – | – | – | — | — |
| 0x0000_0028 | 10 | – | – | – | — | — |
| 0x0000_002C | 11 | – | – | – | ARM Cortex-M0+ | Supervisor Call (SVCall) |
| 0x0000_0030 | 12 | – | – | – | — | — |
| 0x0000_0034 | 13 | – | – | – | — | — |
| 0x0000_0038 | 14 | – | – | – | ARM Cortex-M0+ | Pendable request for system service (PendableSrvReq) |
| 0x0000_003C | 15 | – | – | – | ARM Cortex-M0+ | System Tick timer (SysTick) |
| **Non-Core Vectors** | | | | | | |
| 0x0000_0040 | 16 | 0 | 0 | 0 | Cross Trigger Interface 1 (CTI1) | CTI1 interrupt |
| 0x0000_0044 | 17 | 1 | 0 | 0 | Direct Memory Controller 1 (DMA1) | DMA1 channel 0 or 4 transfer complete |
| 0x0000_0048 | 18 | 2 | 0 | 0 | Direct Memory Controller 1 (DMA1) | DMA1 channel 1 or 5 transfer complete |
| 0x0000_004C | 19 | 3 | 0 | 0 | Direct Memory Controller 1 (DMA1) | DMA1 channel 2 or 6 transfer complete |
| 0x0000_0050 | 20 | 4 | 0 | 1 | Direct Memory Controller 1 (DMA1) | DMA1 channel 3 or 7 transfer complete |
| 0x0000_0054 | 21 | 5 | 0 | 1 | Direct Memory Controller 1 (DMA1) | DMA1 error interrupt (channels 0 - 7) |
| 0x0000_0058 | 22 | 6 | 0 | 1 | Multi System Mode Controller (MSMC) | MSMC (SMC1) interrupt |
| 0x0000_005C | 23 | 7 | 0 | 1 | Low Leakage Wake-Up 1 (LLWU1) | LLWU1 interrupt[4] |
| 0x0000_0060 | 24 | 8 | 0 | 2 | Messaging Unit - Side B (MU-B) | MU-B interrupt |
| 0x0000_0064 | 25 | 9 | 0 | 2 | Watchdog 1 (WDOG1) | WDOG1 interrupt |

*Table continues on the next page...*

## Table 3-16. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0068 | 26 | 10 | 0 | 2 | Cryptographic Acceleration Unit version 3 (CAU3) | Task Complete |
| 0x0000_006C | 27 | 11 | 0 | 2 | Cryptographic Acceleration Unit version 3 (CAU3) | Security Violation |
| 0x0000_0070 | 28 | 12 | 0 | 3 | True Random Number Generator (TRNG) | TRNG interrupt |
| 0x0000_0074 | 29 | 13 | 0 | 3 | Low-Power Periodic Interrupt Timer 1 (LPIT1) | LPIT1 interrupt |
| 0x0000_0078 | 30 | 14 | 0 | 3 | Low-Power Timer 2 (LPTMR2) | LPTMR2 interrupt |
| 0x0000_007C | 31 | 15 | 0 | 3 | Timer / Pulse Width Modulation Module 3 (TPM3) | TPM3 interrupt |
| 0x0000_0080 | 32 | 16 | 0 | 4 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) | LPI2C3 interrupt |
| 0x0000_0084 | 33 | 17 | 0 | 4 | 2.4GHz Radio (RF0) | RF0 Programmable Interrupt 0 |
| 0x0000_0088 | 34 | 18 | 0 | 4 | 2.4GHz Radio (RF0) | RF0 Programmable Interrupt 1 |
| 0x0000_008C | 35 | 19 | 0 | 4 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | LPSPI3 interrupt |
| 0x0000_0090 | 36 | 20 | 0 | 5 | Low-Power Universal Asynchronous Receiver/ Transmitter 3 (LPUART3) | LPUART3 interrupt |
| 0x0000_0094 | 37 | 21 | 0 | 5 | PORTE | PORTE interrupt |
| 0x0000_0098 | 38 | 22 | 0 | 5 | Low-Power Comparator 1 (LPCMP1) | LPCMP1 interrupt |
| 0x0000_009C | 39 | 23 | 0 | 5 | Real Time Clock (RTC) - RTC and Tamper Functions | RTC interrupt |
| 0x0000_00A0 | 40 | 24 | 0 | 6 | Interrupt Multiplexer 0 (INTMUX0) | INTMUX0 Source |

*Table continues on the next page...*

**Table 3-16. Interrupt vector assignments (continued)**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_00A4 | 41 | 25 | 0 | 6 | Interrupt Multiplexer 1 (INTMUX1) | INTMUX1 Source |
| 0x0000_00A8 | 42 | 26 | 0 | 6 | Interrupt Multiplexer 2 (INTMUX2) | INTMUX2 Source |
| 0x0000_00AC | 43 | 27 | 0 | 6 | Interrupt Multiplexer 3 (INTMUX3) | INTMUX3 Source |
| 0x0000_00B0 | 44 | 28 | 0 | 7 | Interrupt Multiplexer 4 (INTMUX4) | INTMUX4 Source |
| 0x0000_00B4 | 45 | 29 | 0 | 7 | Interrupt Multiplexer 5 (INTMUX5) | INTMUX5 Source |
| 0x0000_00B8 | 46 | 30 | 0 | 7 | Interrupt Multiplexer 6 (INTMUX6) | INTMUX6 Source |
| 0x0000_00BC | 47 | 31 | 0 | 7 | Interrupt Multiplexer 7 (INTMUX7) | INTMUX7 Source |

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is: IRQ div 32
3. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4
4. The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery.

**NOTE**

Interrupt Vector handlers must be placed in Flash / RAM memory local to the core that will execute the functions. Vector fetches from the non-local memory are possible but not recommended. Under certain conditions, a hard fault exception can be thrown if these conditions are not met.

### 3.2.1.4 Interrupt Multiplexer n (INTMUXn) input assignments

In systems with a larger number of peripherals, the number of interrupts allowed for the ARM Cortex-M0+ Nested Vectored Interrupt Controller (NVIC) is not sufficient. In order to provide the interrupts from all of the implemented peripherals to the M0+ NVIC

module, an interrupt multiplexer is instantiated. This module takes the interrupts that will not fit in the M0+ NVIC and muxes them onto dedicated NVIC channels. This allows the remainging interrupts to be selected and sent to the NVIC for processing.

The following sections describe the Interrupt Multiplexer (INTMUX) assignments for this device. These assignments are used for all of the implemented interrupt multplexers for this device.

**Table 3-17. INTMUXn input assignments**

| INTMUXn Input | Source module | Source description |
|---|---|---|
| 0 | External Watchdog Monitor (EWM) | EWM interrupt |
| 1 | Flash Controller (FTFE) | Command Complete |
| 2 | Flash Controller (FTFE) | Read Collision |
| 3 | System Power Manager (SPM) | SPM interrupt |
| 4 | System Clock Generator (SCG) | SCG interrupt |
| 5 | Low-Power Periodic Interrupt Timer 0 (LPIT0) | LPIT0 interrupt |
| 6 | Low-Power Timer 0 (LPTMR0) | LPTMR0 interrupt |
| 7 | Low-Power Timer 1 (LPTMR1) | LPTMR1 interrupt |
| 8 | Timer/PWM Module 0 (TPM0 - (6ch)) | TPM0 interrupt |
| 9 | Timer/PWM Module 1 (TPM1 - (2ch)) | TPM1 interrupt |
| 10 | Timer/PWM Module 2 (TPM2 - (6ch)) | TPM2 interrupt |
| 11 | Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) | EMVSIM0 interrupt |
| 12 | Flexible Input/Output 0 (FlexIO0) | FlexIO0 Interrupt |
| 13 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) | LPI2C0 interrupt |
| 14 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) | LPI2C1 interrupt |
| 15 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) | LPI2C2 interrupt |
| 16 | Inter-IC Sound 0 (I2S0) | I2S0 interrupt |
| 17 | Ultra Secured Digital Host Controller 0 (uSDHC0) | uSDHC0 interrupt |
| 18 | Low-Power Serial Peripheral Interface 0 (LPSPI0) | LPSPI0 interrupt |
| 19 | Low-Power Serial Peripheral Interface 1 (LPSPI1) | LPSPI1 interrupt |
| 20 | Low-Power Serial Peripheral Interface 2 (LPSPI2) | LPSPI2 interrupt |
| 21 | Low-Power Universal Asynchronous Receiver/ Transmitter 0 (LPUART0) | LPUART0 interrupt |
| 22 | Low-Power Universal Asynchronous Receiver/ Transmitter 1 (LPUART1) | LPUART1 interrupt |
| 23 | Low-Power Universal Asynchronous Receiver/ Transmitter 2 (LPUART2) | LPUART2 interrupt |
| 24 | Universal Serial Bus 0 (USB0) | USB0 interrupt |
| 25 | PORTA | PORTA interrupt |
| 26 | PORTB | PORTB interrupt |

*Table continues on the next page...*

**Table 3-17. INTMUXn input assignments (continued)**

| INTMUXn Input | Source module | Source description |
|---|---|---|
| 27 | PORTC | PORTC Interrupt |
| 28 | PORTD | PORTD interrupt |
| 29 | Low Power Analog-to-Digital Converter 0 (LPADC0) | LPADC0 interrupt |
| 30 | Low Power Comparator 0 (LPCMP0) | LPCMP0 interrupt |
| 31 | 12-bit Low Power Digital-to-Analog Converter 0 (LPDAC0) | LPDAC0 interrupt |

## 3.2.2 Cortex-M4 Nested Vectored Interrupt Controller (NVIC) configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by Arm and can be found at **arm.com**.

### 3.2.2.1 Cortex M4F interrupt priority levels

This device supports 8 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 3 bits. For example, IPR0 is shown below:

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQ3 | | | 0 | 0 | 0 | 0 | 0 | IRQ2 | | | 0 | 0 | 0 | 0 | 0 | IRQ1 | | | 0 | 0 | 0 | 0 | 0 | IRQ0 | | | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### 3.2.2.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external NMI_b signal. The pin the NMI_b signal is multiplexed on, must be configured for the Non-Maskable Interrupt function to generate the non-maskable interrupt request.

### 3.2.2.3 Cortex-M4F interrupt channel assignments

The interrupt source assignments are defined in the following table.

- Vector number — the value stored on the stack when an interrupt is serviced.
- IRQ number — non-core interrupt source count, which is the vector number minus 16.

The IRQ number is used within ARM's NVIC documentation.

**Table 3-19. Interrupt vector assignments**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| **ARM Core System Handler Vectors** | | | | | | |
| 0x0000_0000 | 0 | – | – | – | ARM Cortex-M4 | Initial Stack Pointer |
| 0x0000_0004 | 1 | – | – | – | ARM Cortex-M4 | Initial Program Counter |
| 0x0000_0008 | 2 | – | – | – | ARM Cortex-M4 | Non-Maskable Interrupt (NMI) |
| 0x0000_000C | 3 | – | – | – | ARM Cortex-M4 | Hard Fault |
| 0x0000_0010 | 4 | – | – | – | ARM Cortex-M4 | MemManage Fault |
| 0x0000_0014 | 5 | – | – | – | ARM Cortex-M4 | Bus Fault |
| 0x0000_0018 | 6 | – | – | – | ARM Cortex-M4 | Usage Fault |
| 0x0000_001C | 7 | – | – | – | — | — |
| 0x0000_0020 | 8 | – | – | – | — | — |
| 0x0000_0024 | 9 | – | – | – | — | — |
| 0x0000_0028 | 10 | – | – | – | — | — |
| 0x0000_002C | 11 | – | – | – | ARM Cortex-M4 | Supervisor Call (SVCall) |
| 0x0000_0030 | 12 | – | – | – | ARM Cortex-M4 | Debug Monitor |
| 0x0000_0034 | 13 | – | – | – | — | — |
| 0x0000_0038 | 14 | – | – | – | ARM Cortex-M4 | Pendable request for system service (PendableSrvReq) |
| 0x0000_003C | 15 | – | – | – | ARM Cortex-M4 | System Tick timer (SysTick) |
| **Non-Core Vectors** | | | | | | |
| 0x0000_0040 | 16 | 0 | 0 | 0 | Cross Trigger Interface 0 (CTI0) / Miscellaneous Control Module (MCM) | CTI0 / MCM interrupt |
| 0x0000_0044 | 17 | 1 | 0 | 0 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 0 transfer complete |
| 0x0000_0048 | 18 | 2 | 0 | 0 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 1 transfer complete |
| 0x0000_004C | 19 | 3 | 0 | 0 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 2 transfer complete |

*Table continues on the next page...*

## Table 3-19. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0050 | 20 | 4 | 0 | 1 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 3 transfer complete |
| 0x0000_0054 | 21 | 5 | 0 | 1 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 4 transfer complete |
| 0x0000_0058 | 22 | 6 | 0 | 1 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 5 transfer complete |
| 0x0000_005C | 23 | 7 | 0 | 1 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 6 transfer complete |
| 0x0000_0060 | 24 | 8 | 0 | 2 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 7 transfer complete |
| 0x0000_0064 | 25 | 9 | 0 | 2 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 8 transfer complete |
| 0x0000_0068 | 26 | 10 | 0 | 2 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 9 transfer complete |
| 0x0000_006C | 27 | 11 | 0 | 2 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 10 transfer complete |
| 0x0000_0070 | 28 | 12 | 0 | 3 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 11 transfer complete |
| 0x0000_0074 | 29 | 13 | 0 | 3 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 12 transfer complete |
| 0x0000_0078 | 30 | 14 | 0 | 3 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 13 transfer complete |
| 0x0000_007C | 31 | 15 | 0 | 3 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 14 transfer complete |
| 0x0000_0080 | 32 | 16 | 0 | 4 | Direct Memory Access Controller 0 (DMA0) | DMA0 channel 15 transfer complete |
| 0x0000_0084 | 33 | 17 | 0 | 4 | Direct Memory Access Controller 0 (DMA0) | DMA0 error interrupt channels 0-15 |
| 0x0000_0088 | 34 | 18 | 0 | 4 | Multi System Mode Controller (MSMC) | MSMC (SMC0) interrupt |

*Table continues on the next page...*

## Table 3-19. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_008C | 35 | 19 | 0 | 4 | External Watchdog Module (EWM) | EWM interrupt |
| 0x0000_0090 | 36 | 20 | 0 | 5 | Flash Memory Controller (FTFE) | Command Complete |
| 0x0000_0094 | 37 | 21 | 0 | 5 | Flash Memory Controller (FTFE) | Read Collision |
| 0x0000_0098 | 38 | 22 | 0 | 5 | Low Leakage Wake-up Unit 0 (LLWU0) | LLWU0 interrupt[4] |
| 0x0000_009C | 39 | 23 | 0 | 5 | Messaging Unit - Side A (MU-A) | MU-A interrupt |
| 0x0000_00A0 | 40 | 24 | 0 | 6 | System Power Manager (SPM) | SPM interrupt |
| 0x0000_00A4 | 41 | 25 | 0 | 6 | Watchdog 0 (WDOG0) | WDOG0 interrupt |
| 0x0000_00A8 | 42 | 26 | 0 | 6 | System Clock Generator (SCG) | SCG interrupt |
| 0x0000_00AC | 43 | 27 | 0 | 6 | Low-Power Periodic Interrupt Timer 0 (LPIT0) | LPIT0 interrupt |
| 0x0000_00B0 | 44 | 28 | 0 | 7 | Real Time Clock (RTC) | RTC and Tamper interrupts |
| 0x0000_00B4 | 45 | 29 | 0 | 7 | Low-Power Timer 0 (LPTMR0) | LPTMR0 interrupt |
| 0x0000_00B8 | 46 | 30 | 0 | 7 | Low-Power Timer 1 (LPTMR1) | LPTMR1 interrupt |
| 0x0000_00BC | 47 | 31 | 0 | 7 | Timer/PWM Module 0 (TPM0) - 6ch | TPM0 interrupt |
| 0x0000_00C0 | 48 | 32 | 1 | 8 | Timer/PWM Module 1 (TPM1) - 2ch | TPM1 interrupt |
| 0x0000_00C4 | 49 | 33 | 1 | 8 | Timer/PWM 2 (TPM2) - 6ch | TPM2 interrupt |
| 0x0000_00C8 | 50 | 34 | 1 | 8 | Smart Card Interface Module 0 (EMVSIM0) | EMVSIM0 interrupt |
| 0x0000_00CC | 51 | 35 | 1 | 8 | Flexible Input / Output 0 (FlexIO0) | FlexIO0 interrupt |
| 0x0000_00D0 | 52 | 36 | 1 | 9 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) | LPI2C0 interrupt |

*Table continues on the next page...*

## Table 3-19. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_00D4 | 53 | 37 | 1 | 9 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) | LPI2C1 interrupt |
| 0x0000_00D8 | 54 | 38 | 1 | 9 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) | LPI2C2 interrupt |
| 0x0000_00DC | 55 | 39 | 1 | 9 | Inter-IC Sound 0 (I2S0) | I2S0 interrupt |
| 0x0000_00E0 | 56 | 40 | 1 | 10 | Ultra Secured Digital Host Controller 0 (uSDHC0) | uSDHC0 interrupt |
| 0x0000_00E4 | 57 | 41 | 1 | 10 | Low-Power Serial Peripheral Interface 0 (LPSPI0) | LPSPI0 interrupt |
| 0x0000_00E8 | 58 | 42 | 1 | 10 | Low-Power Serial Peripheral Interface 1 (LPSPI1) | LPSPI1 interrupt |
| 0x0000_00EC | 59 | 43 | 1 | 10 | Low-Power Serial Peripheral Interface 2 (LPSPI2) | LPSPI2 interrupt |
| 0x0000_00F0 | 60 | 44 | 1 | 11 | Low-Power Universal Asynchronous Receiver/ Transmitter 0 (LPUART0) | LPUART0 interrupt |
| 0x0000_00F4 | 61 | 45 | 1 | 11 | Low-Power Universal Asynchronous Receiver/ Transmitter 1 (LPUART1) | LPUART1 interrupt |
| 0x0000_00F8 | 62 | 46 | 1 | 11 | Low-Power Universal Asynchronous Receiver/ Transmitter 2 (LPUART2) | LPUART2 interrupt |
| 0x0000_00FC | 63 | 47 | 1 | 11 | Universal Serial Bus 0 (USB0) | USB0 interrupt |
| 0x0000_0100 | 64 | 48 | 1 | 12 | PORTA | PORTA interrupt |
| 0x0000_0104 | 65 | 49 | 1 | 12 | PORTB | PORTB interrupt |
| 0x0000_0108 | 66 | 50 | 1 | 12 | PORTC | PORTC interrupt |
| 0x0000_010C | 67 | 51 | 1 | 12 | PORTD | PORTD interrupt |

*Table continues on the next page...*

## Table 3-19. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_0110 | 68 | 52 | 1 | 13 | Low Power Ananlog-to-Digital Converter 0 (LPADC0) | LPADC0 interrupt |
| 0x0000_0114 | 69 | 53 | 1 | 13 | Low-Power Comparator 0 (LPCMP0) | LPCMP0 interrupt |
| 0x0000_0118 | 70 | 54 | 1 | 13 | 12-bit Low Power Digital-to-Analog Converter 0 (LPDAC0) | LPDAC0 interrupt |
| 0x0000_011C | 71 | 55 | 1 | 13 | Cryptographic Acceleration Unit version 3 (CAU3) | Task Completion |
| 0x0000_0120 | 72 | 56 | 1 | 14 | Cryptographic Acceleration Unit version 3 (CAU3) | Security Violation |
| 0x0000_0124 | 73 | 57 | 1 | 14 | True Random Number Generator (TRNG) | TRNG interrupt |
| 0x0000_0128 | 74 | 58 | 1 | 14 | Low-Power Periodic Interrupt Timer 1 (LPIT1) | LPIT1 interrupt |
| 0x0000_012C | 75 | 59 | 1 | 14 | Low-Power Timer 2 (LPTMR2) | LPTMR2 interrupt |
| 0x0000_0130 | 76 | 60 | 1 | 15 | Timer/PWM Module 3 (TPM3) - 2ch | TPM3 interrupt |
| 0x0000_0134 | 77 | 61 | 1 | 15 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) | LPI2C3 interrupt |
| 0x0000_0138 | 78 | 62 | 1 | 15 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | LPSPI3 interrupt |
| 0x0000_013C | 79 | 63 | 1 | 15 | Low-Power Universal Asynchronous Receiver/ Transmitter 3 (LPUART3) | LPUART3 interrupt |
| 0x0000_0140 | 80 | 64 | 2 | 16 | PORTE | PORTE interrupt |
| 0x0000_0144 | 81 | 65 | 2 | 16 | Low-Power Comparator 1 (LPCMP1) | LPCMP1 interrupt |
| 0x0000_0148 | 82 | 66 | 2 | 16 | — | — |

*Table continues on the next page...*

## Table 3-19. Interrupt vector assignments (continued)

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_014C | 83 | 67 | 2 | 16 | — | — |
| 0x0000_0150 | 84 | 68 | 2 | 17 | — | — |
| 0x0000_0154 | 85 | 69 | 2 | 17 | — | — |
| 0x0000_0158 | 86 | 70 | 2 | 17 | — | — |
| 0x0000_015C | 87 | 71 | 2 | 17 | — | — |
| 0x0000_0160 | 88 | 72 | 2 | 18 | — | — |
| 0x0000_0164 | 89 | 73 | 2 | 18 | — | — |
| 0x0000_0168 | 90 | 74 | 2 | 18 | — | — |
| 0x0000_016C | 91 | 75 | 2 | 18 | — | — |
| 0x0000_0170 | 92 | 76 | 2 | 19 | — | — |
| 0x0000_0174 | 93 | 77 | 2 | 19 | — | — |
| 0x0000_0178 | 94 | 78 | 2 | 19 | — | — |
| 0x0000_017C | 95 | 79 | 2 | 19 | — | — |
| 0x0000_0180 | 96 | 80 | 2 | 20 | — | — |
| 0x0000_0184 | 97 | 81 | 2 | 20 | — | — |
| 0x0000_0188 | 98 | 82 | 2 | 20 | — | — |
| 0x0000_018C | 99 | 83 | 2 | 20 | — | — |
| 0x0000_0190 | 100 | 84 | 2 | 21 | — | — |
| 0x0000_0194 | 101 | 85 | 2 | 21 | — | — |
| 0x0000_0198 | 102 | 86 | 2 | 21 | — | — |
| 0x0000_019C | 103 | 87 | 2 | 21 | — | — |
| 0x0000_01A0 | 104 | 88 | 2 | 22 | — | — |
| 0x0000_01A4 | 105 | 89 | 2 | 22 | — | — |
| 0x0000_01A8 | 106 | 90 | 2 | 22 | — | — |
| 0x0000_01AC | 107 | 91 | 2 | 22 | — | — |
| 0x0000_01B0 | 108 | 92 | 2 | 23 | — | — |
| 0x0000_01B4 | 109 | 93 | 2 | 23 | — | — |
| 0x0000_01B8 | 110 | 94 | 2 | 23 | — | — |
| 0x0000_01BC | 111 | 95 | 2 | 23 | — | — |
| 0x0000_01C0 | 112 | 96 | 3 | 24 | — | — |
| 0x0000_01C4 | 113 | 97 | 3 | 24 | — | — |
| 0x0000_01C8 | 114 | 98 | 3 | 24 | — | — |
| 0x0000_01CC | 115 | 99 | 3 | 24 | — | — |
| 0x0000_01D0 | 116 | 100 | 3 | 25 | — | — |
| 0x0000_01D4 | 117 | 101 | 3 | 25 | — | — |
| 0x0000_01D8 | 118 | 102 | 3 | 25 | — | — |
| 0x0000_01DC | 119 | 103 | 3 | 25 | — | — |

*Table continues on the next page...*

**Table 3-19. Interrupt vector assignments (continued)**

| Address | Vector | IRQ[1] | NVIC non-IPR register number [2] | NVIC IPR register number [3] | Source module | Source description |
|---|---|---|---|---|---|---|
| 0x0000_01E0 | 120 | 104 | 3 | 26 | — | — |
| 0x0000_01E4 | 121 | 105 | 3 | 26 | — | — |
| 0x0000_01E8 | 122 | 106 | 3 | 26 | — | — |

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is: IRQ div 32
3. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4
4. The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery.

**NOTE**

Interrupt Vector handlers must be placed in Flash / RAM memory local to the core that will execute the functions. Vector fetches from the non-local memory are possible but not recommended. Under certain conditions, a hard fault exception can be thrown if these conditions are not met.

## 3.3 Asynchronous Wake-up Interrupt Controller (AWIC)

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by Arm and can be found at **arm.com**.

### 3.3.1 AWIC overview

The AWIC block is used to detect asynchronous wake-up events in Stop and VLPS modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing.

### 3.3.2 Wake-up sources

The device uses the following internal and external inputs to the AWIC module. AWIC0 is AWIC in CM4F domain while AWIC1 is AWIC in CM0+ domain.

## Table 3-20. AWIC0 Stop and VLPS wakeup sources

| Wake-up source | Description |
|---|---|
| System resets | All available system resets sources are describted in SMC0 under MSMC chapter |
| SPM | All available interrupt in SPM, such as low/high voltage detect interrupt, low voltage detect/warnning interrupt, and etc. |
| Pin | PTA, PTB, PTC, PTD, PTE pin interrupts |
| LPUART0~3 | Functional when using clock source which is active in Stop and VLPS modes |
| LPI2C0~3 | Address match wakeup |
| LPSPI0~3 | Slave mode interrupt |
| I2S | Functional when using an external bit clock or external master clock |
| EMVSIM | Any enabled interrupt can be a source as long as the module remains clocked. |
| USB Controller | Wakeup |
| Secure Digital Hardware Controller (SDHC) | Wakeup |
| FlexIO | Functional when using clock source which is active in Stop and VLPS modes |
| LPTMR0~2 | Functional when using clock source which is active in Stop, VLPS and LLS/VLLS modes |
| TPM0~3 | Functional when using clock source which is active in Stop and VLPS modes |
| RTC | Functional in Stop, VLPS, LLS and VLLSx modes |
| LPIT0/1 | Any enabled interrupt can be a source as long as the module remains clocked. |
| Tamper detect | Interrupt |
| NMI | NMI is routed to CM4F or CM0+ automatically, only boot core has NMI |
| LPCMP0/1 | Interrupt in normal or trigger mode |
| LPDAC | Any enabled interrupt can be a source as long as the module remains clocked. |
| LPADC | The LPADC is functional when using internal clock source |
| LLWU0 | Any enabled interrupt can be a source as long as the module remains clocked. |
| MUA | Any enabled interrupt can be a source as long as the module remains clocked. |
| WDOG0 | Watchdog0 Interrupt |

## Table 3-21. AWIC1 Stop and VLPS wakeup sources

| Wake-up source | Description |
|---|---|
| System resets | All available system resets sources are describted in SMC1 under MSMC chapter |
| PortE | PTE pin interrupts |

*Table continues on the next page...*

**Table 3-21. AWIC1 Stop and VLPS wakeup sources (continued)**

| Wake-up source | Description |
|---|---|
| LPUART3 | Functional when using clock source which is active in Stop and VLPS modes |
| LPI2C03 | Address match wakeup |
| LPSPI3 | Any enabled interrupt can be a source as long as the module remains clocked. |
| 2.4GHz Radio | RF Programmable Interrupt 0 |
| 2.4GHz Radio | RF Programmable Interrupt 1 |
| LPTMR2 | Functional when using clock source which is active in Stop, VLPS and LLS/VLLS modes |
| TPM3 | Functional when using clock source which is active in Stop and VLPS modes |
| RTC | Functional in Stop,VLPS,LLS and VLLSx modes |
| LPIT1 | Any enabled interrupt can be a source as long as the module remains clocked. |
| NMI | NMI is routed to CM4F or CM0+ automatically, only boot core has NMI |
| LPCMP1 | Interrupt in normal or trigger mode |
| TRNG | TRNG has no stop wakeup capability |
| LLWU1 | Any enabled interrupt can be a source as long as the module remains clocked. |
| MUB | Any enabled interrupt can be a source as long as the module remains clocked. |
| WDOG1 | Watchdog1 Interrupt |
| INTMUX0~7 | All other wakeup sources availabe in AWIC0 can be selected through INTMUXn. Please refer to the INTMUXn description. |

# 3.4 Event unit

## 3.4.1 Introduction

RI5CY and ZERO-RISCY core both have an event unit to support vectorized interrupts of up to 32 lines and event triggering of up to 32 input lines. The interrupt and event lines are separately masked and buffered.

This is block diagram of event unit.

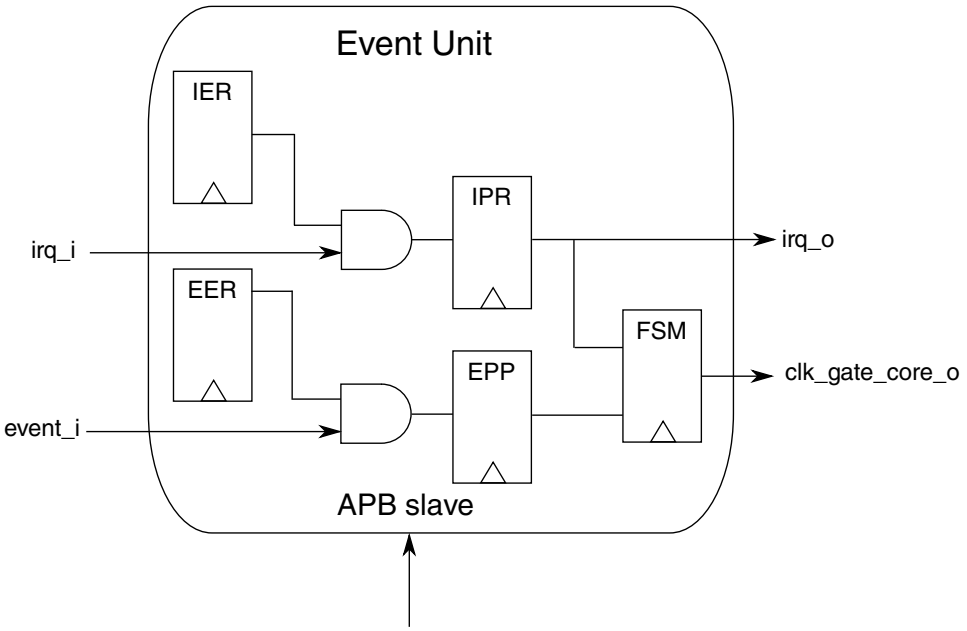



**Figure 3-5. Block diagram**

## 3.4.2 Memory map and register descriptions

### EVENT memory map

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4101_F000 | Interrupt Enable Register (EVENT1_INTPTEN) | 32 | R/W | 0000_0000h | 3.4.2.1/131 |
| 4101_F004 | Interrupt Pending Register (EVENT1_INTPTPEND) | 32 | R/W | 0000_0000h | 3.4.2.2/131 |
| 4101_F008 | Set Interrupt Pending Register (EVENT1_INTPTPENDSET) | 32 | w1c | 0000_0000h | 3.4.2.3/132 |
| 4101_F00C | Clear Interrupt Pending Register (EVENT1_INTPTPENDCLEAR) | 32 | w1c | 0000_0000h | 3.4.2.4/132 |
| 4101_F010 | Interrupt Secure Register (EVENT1_INTPTSECURE) | 32 | R/W | 0000_0000h | 3.4.2.5/133 |
| 4101_F014 | Interrupt Priority 0 Register (EVENT1_INTPTPRI0) | 32 | R/W | 0000_0000h | 3.4.2.6/133 |
| 4101_F018 | Interrupt Priority 1 Register (EVENT1_INTPTPRI1) | 32 | R/W | 0000_0000h | 3.4.2.7/135 |
| 4101_F01C | Interrupt Priority 2 Register (EVENT1_INTPTPRI2) | 32 | R/W | 0000_0000h | 3.4.2.8/136 |
| 4101_F020 | Interrupt Priority 3 Register (EVENT1_INTPTPRI3) | 32 | R/W | 0000_0000h | 3.4.2.9/137 |
| 4101_F024 | Interrupt Priority Base (EVENT1_INTPRIBASE) | 32 | R/W | 0000_0008h | 3.4.2.10/ 138 |
| 4101_F028 | Interrupt Active Register (EVENT1_INTPTENACTIVE) | 32 | R | 0000_0000h | 3.4.2.11/ 139 |
| 4101_F02C | Interrupt Active Priority 0 Register (EVENT1_INTACTPRI0) | 32 | R | 0000_0000h | 3.4.2.12/ 139 |
| 4101_F030 | Interrupt Active Priority 1 Register (EVENT1_INTACTPRI1) | 32 | R | 0000_0000h | 3.4.2.13/ 141 |

*Table continues on the next page...*

## EVENT memory map (continued)

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| 4101_F034 | Interrupt Active Priority 2 Register (EVENT1_INTACTPRI2) | 32 | R | 0000_0000h | 3.4.2.14/ 142 |
| 4101_F038 | Interrupt Active Priority 3 Register (EVENT1_INTACTPRI3) | 32 | R | 0000_0000h | 3.4.2.15/ 143 |
| 4101_F040 | Event Enable Register (EVENT1_EVENTEN) | 32 | R/W | 0000_0000h | 3.4.2.16/ 145 |
| 4101_F044 | Event Pending Register (EVENT1_EVENTPEND) | 32 | R/W | 0000_0000h | 3.4.2.17/ 145 |
| 4101_F048 | Set Event Pending Register (EVENT1_EVTPENDSET) | 32 | w1c | 0000_0000h | 3.4.2.18/ 146 |
| 4101_F04C | Clear Event Pending Register (EVENT1_EVTPENDCLEAR) | 32 | w1c | 0000_0000h | 3.4.2.19/ 146 |
| 4101_F080 | Sleep Control Register (EVENT1_SLPCTRL) | 32 | R/W | 0000_0000h | 3.4.2.20/ 147 |
| 4101_F084 | Sleep Status Register (EVENT1_SLPSTATUS) | 32 | R/W | 0000_0000h | 3.4.2.21/ 147 |
| E004_1000 | Interrupt Enable Register (EVENT0_INTPTEN) | 32 | R/W | 0000_0000h | 3.4.2.1/131 |
| E004_1004 | Interrupt Pending Register (EVENT0_INTPTPEND) | 32 | R/W | 0000_0000h | 3.4.2.2/131 |
| E004_1008 | Set Interrupt Pending Register (EVENT0_INTPTPENDSET) | 32 | w1c | 0000_0000h | 3.4.2.3/132 |
| E004_100C | Clear Interrupt Pending Register (EVENT0_INTPTPENDCLEAR) | 32 | w1c | 0000_0000h | 3.4.2.4/132 |
| E004_1010 | Interrupt Secure Register (EVENT0_INTPTSECURE) | 32 | R/W | 0000_0000h | 3.4.2.5/133 |
| E004_1014 | Interrupt Priority 0 Register (EVENT0_INTPTPRI0) | 32 | R/W | 0000_0000h | 3.4.2.6/133 |
| E004_1018 | Interrupt Priority 1 Register (EVENT0_INTPTPRI1) | 32 | R/W | 0000_0000h | 3.4.2.7/135 |
| E004_101C | Interrupt Priority 2 Register (EVENT0_INTPTPRI2) | 32 | R/W | 0000_0000h | 3.4.2.8/136 |
| E004_1020 | Interrupt Priority 3 Register (EVENT0_INTPTPRI3) | 32 | R/W | 0000_0000h | 3.4.2.9/137 |
| E004_1024 | Interrupt Priority Base (EVENT0_INTPRIBASE) | 32 | R/W | 0000_0008h | 3.4.2.10/ 138 |
| E004_1028 | Interrupt Active Register (EVENT0_INTPTENACTIVE) | 32 | R | 0000_0000h | 3.4.2.11/ 139 |
| E004_102C | Interrupt Active Priority 0 Register (EVENT0_INTACTPRI0) | 32 | R | 0000_0000h | 3.4.2.12/ 139 |
| E004_1030 | Interrupt Active Priority 1 Register (EVENT0_INTACTPRI1) | 32 | R | 0000_0000h | 3.4.2.13/ 141 |
| E004_1034 | Interrupt Active Priority 2 Register (EVENT0_INTACTPRI2) | 32 | R | 0000_0000h | 3.4.2.14/ 142 |
| E004_1038 | Interrupt Active Priority 3 Register (EVENT0_INTACTPRI3) | 32 | R | 0000_0000h | 3.4.2.15/ 143 |
| E004_1040 | Event Enable Register (EVENT0_EVENTEN) | 32 | R/W | 0000_0000h | 3.4.2.16/ 145 |
| E004_1044 | Event Pending Register (EVENT0_EVENTPEND) | 32 | R/W | 0000_0000h | 3.4.2.17/ 145 |

*Table continues on the next page...*

**EVENT memory map (continued)**

| Absolute address (hex) | Register name | Width (in bits) | Access | Reset value | Section/ page |
|---|---|---|---|---|---|
| E004_1048 | Set Event Pending Register (EVENT0_EVTPENDSET) | 32 | w1c | 0000_0000h | 3.4.2.18/ 146 |
| E004_104C | Clear Event Pending Register (EVENT0_EVTPENDCLEAR) | 32 | w1c | 0000_0000h | 3.4.2.19/ 146 |
| E004_1080 | Sleep Control Register (EVENT0_SLPCTRL) | 32 | R/W | 0000_0000h | 3.4.2.20/ 147 |
| E004_1084 | Sleep Status Register (EVENT0_SLPSTATUS) | 32 | R/W | 0000_0000h | 3.4.2.21/ 147 |

## 3.4.2.1 Interrupt Enable Register (EVENTx_INTPTEN)

Address: Base address + 0h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IEN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTEN field descriptions**

| Field | Description |
|---|---|
| IEN | Interrupt n Enable<br><br>This bit enables interrupt. Each interrupt takes one bit. For example, interrupt 0 is in bit 0 while interrupt 31 is in bit 31.<br><br>0 Interrupt n is disabled.<br>1 Interrupt n is enabled. |

## 3.4.2.2 Interrupt Pending Register (EVENTx_INTPTPEND)

This register indicates the interrupts pending status. It can also be used to set or clear pending.

Address: Base address + 4h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IPEND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPEND field descriptions**

| Field | Description |
|---|---|
| IPEND | Interrupt n Pending |

**EVENTx_INTPTPEND field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit indicates whether interrupt is in pending. Each interrupt takes one bit. For example, interrupt 0 is in bit 0 while interrupt 31 is in bit 31.<br><br>0 Interrupt n is not pending.<br>1 Interrupt n is pending. |

### 3.4.2.3 Set Interrupt Pending Register (EVENTx_INTPTPENDSET)

This register sets the interrupts in pending status.

Address: Base address + 8h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IPENDSET | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPENDSET field descriptions**

| Field | Description |
|---|---|
| IPENDSET | Set Interrupt n Pending<br><br>This bit reads as 0 and write 1 to set interrupt in pending status. Each interrupt takes one bit. For example, interrupt 0 is in bit 0 while interrupt 31 is in bit 31.<br><br>0 Not set interrupt n in pending status<br>1 Set interrupt n in pending status. |

### 3.4.2.4 Clear Interrupt Pending Register (EVENTx_INTPTPENDCLEAR)

This register clear the interrupts out of pending status.

Address: Base address + Ch offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IPENDCLEAR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPENDCLEAR field descriptions**

| Field | Description |
|---|---|
| IPENDCLEAR | Clear Interrupt n out of Pending<br><br>This bit reads as 0 and write 1 to clear interrupt out of pending status. Each interrupt takes one bit. For example, interrupt 0 is in bit 0 while interrupt 31 is in bit 31. |

**EVENTx_INTPTPENDCLEAR field descriptions (continued)**

| Field | Description |
|---|---|
| | 0 Not clear interrupt n out of pending status<br>1 Clear interrupt n out of pending status. |

### 3.4.2.5 Interrupt Secure Register (EVENTx_INTPTSECURE)

This register sets the secure feature for interrupt.

Address: Base address + 10h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ISECURE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTSECURE field descriptions**

| Field | Description |
|---|---|
| ISECURE | Set secure feature of Interrupt n<br><br>0 Set interrupt n out of security<br>1 Set interrupt n in secruity. |

### 3.4.2.6 Interrupt Priority 0 Register (EVENTx_INTPTPRI0)

This register sets 8-level priority of interrupt 0-7.

Address: Base address + 14h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI7 | | | 0 | IPRI6 | | | 0 | IPRI5 | | | 0 | IPRI4 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI3 | | | 0 | IPRI2 | | | 0 | IPRI1 | | | 0 | IPRI0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPRI0 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IPRI7 | Interrupt 7 Priority<br><br>This bit sets the interrupt 7 priority |

*Table continues on the next page...*

## EVENTx_INTPTPRI0 field descriptions (continued)

| Field | Description |
|---|---|
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IPRI6 | Interrupt 6 Priority<br><br>This bit sets the interrupt 6 priority |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IPRI5 | Interrupt 5 Priority<br><br>This bit sets the interrupt 5 priority |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IPRI4 | Interrupt 4 Priority<br><br>This bit sets the interrupt 4 priority |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IPRI3 | Interrupt 3 Priority<br><br>This bit sets the interrupt 3 priority |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IPRI2 | Interrupt 2 Priority<br><br>This bit sets the interrupt 2 priority |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IPRI1 | Interrupt 1 Priority<br><br>This bit sets the interrupt 1 priority |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IPRI0 | Interrupt 0 Priority<br><br>This bit sets the interrupt 0 priority |

### 3.4.2.7 Interrupt Priority 1 Register (EVENTx_INTPTPRI1)

This register sets 8-level priority of interrupt 8-15.

Address: Base address + 18h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI15 | | | 0 | IPRI14 | | | 0 | IPRI13 | | | 0 | IPRI12 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI11 | | | 0 | IPRI10 | | | 0 | IPRI9 | | | 0 | IPRI8 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPRI1 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IPRI15 | Interrupt 15 Priority<br><br>This bit sets the interrupt 15 priority |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IPRI14 | Interrupt 14 Priority<br><br>This bit sets the interrupt 14 priority |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IPRI13 | Interrupt 13 Priority<br><br>This bit sets the interrupt 13 priority |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IPRI12 | Interrupt 12 Priority<br><br>This bit sets the interrupt 12 priority |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IPRI11 | Interrupt 11 Priority<br><br>This bit sets the interrupt 11 priority |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IPRI10 | Interrupt 10 Priority<br><br>This bit sets the interrupt 10 priority |

*Table continues on the next page...*

**EVENTx_INTPTPRI1 field descriptions (continued)**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IPRI9 | Interrupt 9 Priority<br><br>This bit sets the interrupt 9 priority |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IPRI8 | Interrupt 8 Priority<br><br>This bit sets the interrupt 8 priority |

## 3.4.2.8 Interrupt Priority 2 Register (EVENTx_INTPTPRI2)

This register sets 8-level priority of interrupt 16-23.

Address: Base address + 1Ch offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI23 | | | 0 | IPRI22 | | | 0 | IPRI21 | | | 0 | IPRI20 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI19 | | | 0 | IPRI18 | | | 0 | IPRI17 | | | 0 | IPRI16 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPRI2 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IPRI23 | Interrupt 23 Priority<br><br>This bit sets the interrupt 23 priority |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IPRI22 | Interrupt 22 Priority<br><br>This bit sets the interrupt 22 priority |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IPRI21 | Interrupt 21 Priority<br><br>This bit sets the interrupt 21 priority |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |

*Table continues on the next page...*

**EVENTx_INTPTPRI2 field descriptions (continued)**

| Field | Description |
|---|---|
| 18–16<br>IPRI20 | Interrupt 20 Priority<br><br>This bit sets the interrupt 20 priority |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IPRI19 | Interrupt 19 Priority<br><br>This bit sets the interrupt 19 priority |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IPRI18 | Interrupt 18 Priority<br><br>This bit sets the interrupt 18 priority |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IPRI17 | Interrupt 17 Priority<br><br>This bit sets the interrupt 17 priority |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IPRI16 | Interrupt 16 Priority<br><br>This bit sets the interrupt 16 priority |

## 3.4.2.9 Interrupt Priority 3 Register (EVENTx_INTPTPRI3)

This register sets 8-level priority of interrupt 24-31.

Address: Base address + 20h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI31 | | | 0 | IPRI30 | | | 0 | IPRI29 | | | 0 | IPRI28 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IPRI27 | | | 0 | IPRI26 | | | 0 | IPRI25 | | | 0 | IPRI24 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTPRI3 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IPRI31 | Interrupt 31 Priority |

*Table continues on the next page...*

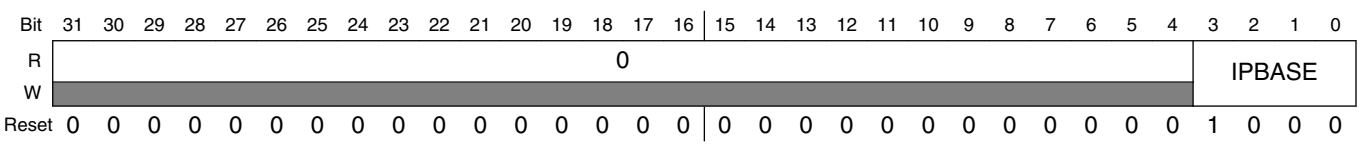
**EVENTx_INTPTPRI3 field descriptions (continued)**

| Field | Description |
|---|---|
| | This bit sets the interrupt 31 priority |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IPRI30 | Interrupt 30 Priority<br><br>This bit sets the interrupt 30 priority |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IPRI29 | Interrupt 29 Priority<br><br>This bit sets the interrupt 29 priority |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IPRI28 | Interrupt 28 Priority<br><br>This bit sets the interrupt 28 priority |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IPRI27 | Interrupt 27 Priority<br><br>This bit sets the interrupt 27 priority |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IPRI26 | Interrupt 26 Priority<br><br>This bit sets the interrupt 26 priority |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IPRI25 | Interrupt 25 Priority<br><br>This bit sets the interrupt 25 priority |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IPRI24 | Interrupt 24 Priority<br><br>This bit sets the interrupt 24 priority |

## 3.4.2.10 Interrupt Priority Base (EVENTx_INTPRIBASE)

This register sets base priority, the interrupt with which priority is lower than base priority cannot be handled.

Address: Base address + 24h offset



**EVENTx_INTPRIBASE field descriptions**

| Field | Description |
|---|---|
| 31–4<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IPBASE | Set interrupt base priority. |

## 3.4.2.11 Interrupt Active Register (EVENTx_INTPTENACTIVE)

Address: Base address + 28h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IACTIVE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTPTENACTIVE field descriptions**

| Field | Description |
|---|---|
| IACTIVE | Interrupt n Enable<br><br>This bit indicates whether each interrupt is active. Each interrupt takes one bit. For example, interrupt 0 is in bit 0 while interrupt 31 is in bit 31.<br><br>0 Interrupt n is not active.<br>1 Interrupt n is active.. |

## 3.4.2.12 Interrupt Active Priority 0 Register (EVENTx_INTACTPRI0)

This register indicates the priority of the active interrupts 0-7.

Address: Base address + 2Ch offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI7 | | | 0 | IAPRI6 | | | 0 | IAPRI5 | | | 0 | IAPRI4 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI3 | | | 0 | IAPRI2 | | | 0 | IAPRI1 | | | 0 | IAPRI0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## EVENTx_INTACTPRI0 field descriptions

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IAPRI7 | Interrupt 7 Active Priority<br><br>This bit indicates the interrupt 7 priority if it is active. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IAPRI6 | Interrupt 6 Active Priority<br><br>This bit sets the interrupt 6 priority if it is active. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IAPRI5 | Interrupt 5 Active Priority<br><br>This bit indicates the interrupt 5 priority if it is active. |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IAPRI4 | Interrupt 4 Active Priority<br><br>This bit indicates the interrupt 4 priority if it is active. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IAPRI3 | Interrupt 3 Active Priority<br><br>This bit indicates the interrupt 3 priority if it is active. |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IAPRI2 | Interrupt 2 Active Priority<br><br>This bit indicates the interrupt 2 priority if it is active. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IAPRI1 | Interrupt 1 Active Priority<br><br>This bit indicates the interrupt 1 priority if it is active. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IAPRI0 | Interrupt 0 Active Priority<br><br>This bit indicates the interrupt 0 priority if it is active. |

### 3.4.2.13 Interrupt Active Priority 1 Register (EVENTx_INTACTPRI1)

This register indicates the priority of the active interrupts 8-15.

Address: Base address + 30h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI15 | | | 0 | IAPRI14 | | | 0 | IAPRI13 | | | 0 | IAPRI12 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI11 | | | 0 | IAPRI10 | | | 0 | IAPRI9 | | | 0 | IAPRI8 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTACTPRI1 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IAPRI15 | Interrupt 15 Active Priority<br><br>This bit indicates the interrupt 15 priority if it is active. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IAPRI14 | Interrupt 14 Active Priority<br><br>This bit sets the interrupt 14 priority if it is active. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IAPRI13 | Interrupt 13 Active Priority<br><br>This bit indicates the interrupt 13 priority if it is active. |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IAPRI12 | Interrupt 12 Active Priority<br><br>This bit indicates the interrupt 12 priority if it is active. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IAPRI11 | Interrupt 11 Active Priority<br><br>This bit indicates the interrupt 11 priority if it is active. |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IAPRI10 | Interrupt 10 Active Priority<br><br>This bit indicates the interrupt 10 priority if it is active. |

*Table continues on the next page...*

**EVENTx_INTACTPRI1 field descriptions (continued)**

| Field | Description |
|---|---|
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IAPRI9 | Interrupt 9 Active Priority<br><br>This bit indicates the interrupt 9 priority if it is active. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IAPRI8 | Interrupt 8 Active Priority<br><br>This bit indicates the interrupt 8 priority if it is active. |

## 3.4.2.14 Interrupt Active Priority 2 Register (EVENTx_INTACTPRI2)

This register indicates the priority of the active interrupts 16-23.

Address: Base address + 34h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI23 | | | 0 | IAPRI22 | | | 0 | IAPRI21 | | | 0 | IAPRI20 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI19 | | | 0 | IAPRI18 | | | 0 | IAPRI17 | | | 0 | IAPRI16 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_INTACTPRI2 field descriptions**

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IAPRI23 | Interrupt 23 Active Priority<br><br>This bit indicates the interrupt 23 priority if it is active. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IAPRI22 | Interrupt 22 Active Priority<br><br>This bit sets the interrupt 22 priority if it is active. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IAPRI21 | Interrupt 21 Active Priority<br><br>This bit indicates the interrupt 21 priority if it is active. |

*Table continues on the next page...*

**EVENTx_INTACTPRI2 field descriptions (continued)**

| Field | Description |
|---|---|
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IAPRI20 | Interrupt 20 Active Priority<br><br>This bit indicates the interrupt 20 priority if it is active. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IAPRI19 | Interrupt 19 Active Priority<br><br>This bit indicates the interrupt 19 priority if it is active. |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IAPRI18 | Interrupt 18 Active Priority<br><br>This bit indicates the interrupt 18 priority if it is active. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IAPRI17 | Interrupt 17 Active Priority<br><br>This bit indicates the interrupt 17 priority if it is active. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IAPRI16 | Interrupt 16 Active Priority<br><br>This bit indicates the interrupt 16 priority if it is active. |

## 3.4.2.15 Interrupt Active Priority 3 Register (EVENTx_INTACTPRI3)

This register indicates the priority of the active interrupts 24-31.

Address: Base address + 38h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI31 | | | 0 | IAPRI30 | | | 0 | IAPRI29 | | | 0 | IAPRI28 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IAPRI27 | | | 0 | IAPRI26 | | | 0 | IAPRI25 | | | 0 | IAPRI24 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## EVENTx_INTACTPRI3 field descriptions

| Field | Description |
|---|---|
| 31<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 30–28<br>IAPRI31 | Interrupt 31 Active Priority<br><br>This bit indicates the interrupt 31 priority if it is active. |
| 27<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 26–24<br>IAPRI30 | Interrupt 30 Active Priority<br><br>This bit sets the interrupt 30 priority if it is active. |
| 23<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 22–20<br>IAPRI29 | Interrupt 29 Active Priority<br><br>This bit indicates the interrupt 29 priority if it is active. |
| 19<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 18–16<br>IAPRI28 | Interrupt 28 Active Priority<br><br>This bit indicates the interrupt 28 priority if it is active. |
| 15<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 14–12<br>IAPRI27 | Interrupt 27 Active Priority<br><br>This bit indicates the interrupt 27 priority if it is active. |
| 11<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 10–8<br>IAPRI26 | Interrupt 26 Active Priority<br><br>This bit indicates the interrupt 26 priority if it is active. |
| 7<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| 6–4<br>IAPRI25 | Interrupt 25 Active Priority<br><br>This bit indicates the interrupt 25 priority if it is active. |
| 3<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| IAPRI24 | Interrupt 24 Active Priority<br><br>This bit indicates the interrupt 24 priority if it is active. |

### 3.4.2.16 Event Enable Register (EVENTx_EVENTEN)

Address: Base address + 40h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EEN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_EVENTEN field descriptions**

| Field | Description |
|---|---|
| EEN | Event n Enable<br><br>This bit enables event. Each event takes one bit. For example, event 0 is in bit 0 while event 31 is in bit 31.<br><br>0 Event n is disabled.<br>1 Event n is enabled. |

### 3.4.2.17 Event Pending Register (EVENTx_EVENTPEND)

This register indicates the event pending status. It can also be used to set or clear events.

Address: Base address + 44h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EPEND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_EVENTPEND field descriptions**

| Field | Description |
|---|---|
| EPEND | Event n Pending<br><br>This bit indicates whether event is in pending. Each event takes one bit. For example, event 0 is in bit 0 while event 31 is in bit 31.<br><br>0 Event n is not pending.<br>1 Event n is pending. |

### 3.4.2.18 Set Event Pending Register (EVENTx_EVTPENDSET)

This register sets the event in pending status.

Address: Base address + 48h offset

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EPENDSET | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_EVTPENDSET field descriptions**

| Field | Description |
|---|---|
| EPENDSET | Set Event n Pending<br><br>This bit reads as 0 and write 1 to set event in pending status. Each event takes one bit. For example, event 0 is in bit 0 while event 31 is in bit 31.<br><br>0 Not set event n in pending status<br>1 Set event n in pending status. |

### 3.4.2.19 Clear Event Pending Register (EVENTx_EVTPENDCLEAR)

This register clear the events out of pending status.

Address: Base address + 4Ch offset

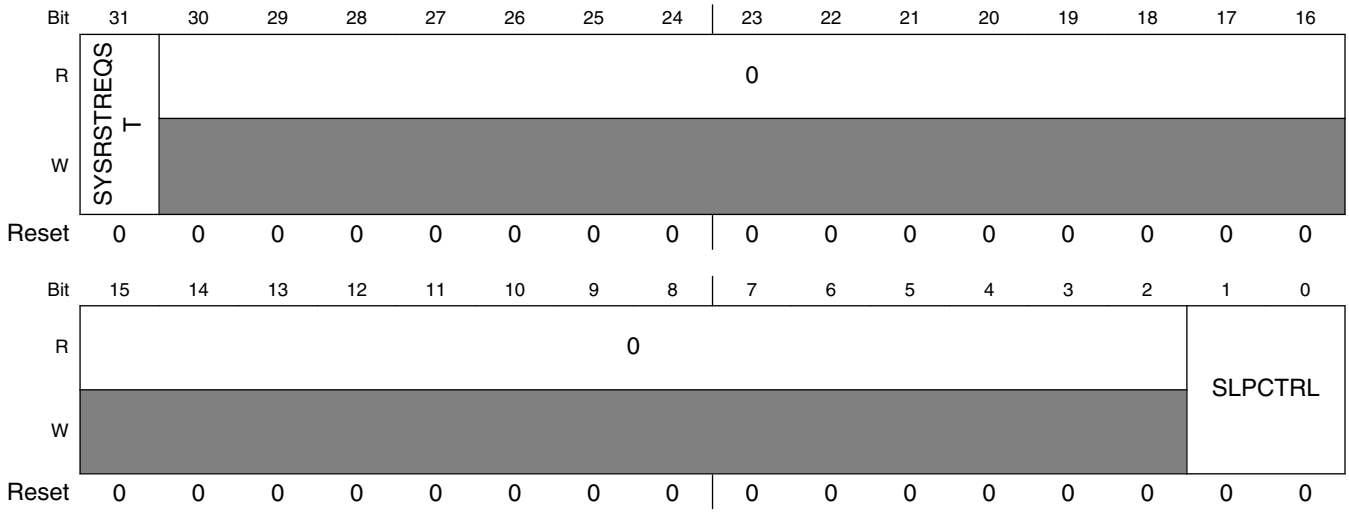
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EPENDCLEAR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENTx_EVTPENDCLEAR field descriptions**

| Field | Description |
|---|---|
| EPENDCLEAR | Clear Event n out of Pending<br><br>This bit reads as 0 and write 1 to clear event out of pending status. Each event takes one bit. For example, event 0 is in bit 0 while event 31 is in bit 31.<br><br>0 Not clear event n out of pending status<br>1 Clear event n out of pending status. |

## 3.4.2.20 Sleep Control Register (EVENTx_SLPCTRL)

Address: Base address + 80h offset

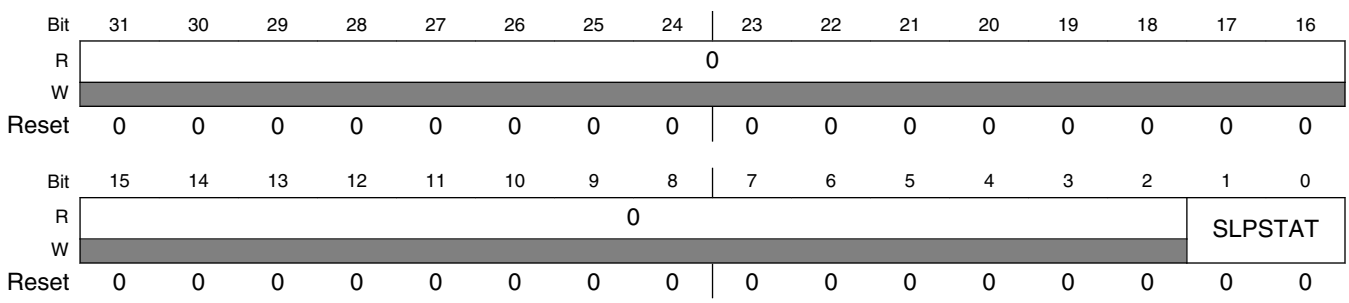



**EVENTx_SLPCTRL field descriptions**

| Field | Description |
|---|---|
| 31<br>SYSRSTREQST | System Reset Request<br><br>0 Do not send system reset request.<br>1 Send system reset request |
| 30–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| SLPCTRL | Sleep Mode Control<br><br>01 Sleep enable<br>10 Deep sleep enable |

## 3.4.2.21 Sleep Status Register (EVENTx_SLPSTATUS)

Address: Base address + 84h offset



**EVENTx_SLPSTATUS field descriptions**

| Field | Description |
|---|---|
| 31–2<br>Reserved | This field is reserved.<br>This read-only field is reserved and always has the value 0. |
| SLPSTAT | Sleep Status<br><br>01 In sleep mode<br>10 In deep sleep mode |

## 3.4.3 Interrupt assignment to RI5CY

**Table 3-22. RI5CY interrupt assignment**

| Address | Vector | IRQ | Module | Description |
|---|---|---|---|---|
| **0x000F_FF00** | 0 | 0 | DMA0 | DMA0 Channel 0, 4, 8, 12 transfer complete |
| **0x000F_FF04** | 1 | 1 | DMA0 | DMA0 Channel 1, 5, 9, 13 transfer complete |
| **0x000F_FF08** | 2 | 2 | DMA0 | DMA0 Channel 2, 6, 10, 14 transfer complete |
| **0x000F_FF0C** | 3 | 3 | DMA0 | DMA0 Channel 3, 7, 11, 15 transfer complete |
| **0x000F_FF10** | 4 | 4 | DMA0 | DMA0 error interrupt channels 0-15 |
| **0x000F_FF14** | 5 | 5 | CMC | |
| **0x000F_FF18** | 6 | 6 | MU-A | |
| **0x000F_FF1C** | 7 | 7 | USB0 | |
| **0x000F_FF20** | 8 | 8 | SDHC0 | |
| **0x000F_FF24** | 9 | 9 | SAI0 | |
| **0x000F_FF28** | 10 | 10 | FLEXIO0 | |
| **0x000F_FF2C** | 11 | 11 | EMVSIM0 | |
| **0x000F_FF30** | 12 | 12 | LPIT0 | |
| **0x000F_FF34** | 13 | 13 | LPSPI0 | |
| **0x000F_FF38** | 14 | 14 | LPSPI1 | |
| **0x000F_FF3C** | 15 | 15 | LPI2C0 | |
| **0x000F_FF40** | 16 | 16 | LPI2C1 | |
| **0x000F_FF44** | 17 | 17 | LPUART0 | |
| **0x000F_FF48** | 18 | 18 | PORTA | |
| **0x000F_FF4C** | 19 | 19 | TPM0 | |
| **0x000F_FF50** | 20 | 20 | DAC0 | |
| **0x000F_FF54** | 21 | 21 | LPADC0 | |
| **0x000F_FF58** | 22 | 22 | LPCMP0 | |
| **0x000F_FF5C** | 23 | 23 | RTC | |
| **0x000F_FF60** | 24 | 24 | RI5CY INTMUX | INTMUX Channel 0 |
| **0x000F_FF64** | 25 | 25 | RI5CY INTMUX | INTMUX Channel 1 |

*Table continues on the next page...*

**Table 3-22. RI5CY interrupt assignment (continued)**

| Address | Vector | IRQ | Module | Description |
|---|---|---|---|---|
| **0x000F_FF68** | 26 | 26 | RI5CY INTMUX | INTMUX Channel 2 |
| **0x000F_FF6C** | 27 | 27 | RI5CY INTMUX | INTMUX Channel 3 |
| **0x000F_FF70** | 28 | 28 | RI5CY INTMUX | INTMUX Channel 4 |
| **0x000F_FF74** | 29 | 29 | RI5CY INTMUX | INTMUX Channel 5 |
| **0x000F_FF78** | 30 | 30 | RI5CY INTMUX | INTMUX Channel 6 |
| **0x000F_FF7C** | 31 | 31 | RI5CY INTMUX | INTMUX Channel 7 |
| **0x000F_FF80** | 32 | - | RI5CY | Reset, Initial PC |
| **0x000F_FF84** | 33 | - | RI5CY | Illegal Instruction |
| **0x000F_FF88** | 34 | - | RI5CY | ECALL Instruction Executed |
| **0x000F_FF8C** | 35 | - | RI5CY | Load Store Unit Error |

Vector0-31: external interrupts, 8-level programmable priority.

Vector32-35: internal exceptions, fixed priority, the lower the vector number, the higher the priority.

The priority of the internal exception is higher than the external interrupts.

RI5CY INTMUX Source assignment:

**Table 3-23. RI5CY INTMUX Source assignment**

| INTMUX Input | Source Module | Description |
|---|---|---|
| 0 | EWM | |
| 1 | FTFE | FTFE command complete |
| 2 | FTFE | FTFE read collision |
| 3 | LLWU0 | |
| 4 | SPM | |
| 5 | WDOG0 | |
| 6 | SCG | |
| 7 | LPTMR0 | |
| 8 | LPTMR1 | |
| 9 | TPM1 | |
| 10 | TPM2 | |
| 11 | LPI2C2 | |
| 12 | LPSPI2 | |
| 13 | LPUART1 | |

*Table continues on the next page...*

**Table 3-23. RI5CY INTMUX Source assignment (continued)**

| INTMUX Input | Source Module | Description |
|---|---|---|
| 14 | LPUART2 | |
| 15 | PORTB | |
| 16 | PORTC | |
| 17 | PORTD | |
| 18 | CAU3 | CAU3 task completion |
| 19 | CAU3 | CAU3 security violation |
| 20 | TRNG | |
| 21 | LPIT1 | |
| 22 | LPTMR2 | |
| 23 | TPM3 | |
| 24 | LPI2C3 | |
| 25 | LPSPI3 | |
| 26 | LPUART3 | |
| 27 | PORTE | |
| 28 | LPCMP1 | |
| 29 | RF0 | RF0 programmable interrupt 0 |
| 30 | RF0 | RF0 programmable interrupt 1 |
| 31 | | |

## 3.4.4 Interrupt assignment to ZERO-RISCY

**Table 3-24. ZERO-RISCY interrupt assignment**

| Address | Vector | IRQ | Module | Description |
|---|---|---|---|---|
| **0x0103_FF00 - 0x0103_FF7C** | 0 - 31 | 0 - 31 | same to vector16-47 of Cortex-M0+ | |
| **0x0103_FF80** | 32 | - | RI5CY | Reset, Initial PC |
| **0x0103_FF84** | 33 | - | RI5CY | Illegal Instruction |
| **0x0103_FF88** | 34 | - | RI5CY | ECALL Instruction Executed |
| **0x0103_FF8C** | 35 | - | RI5CY | Load Store Unit Error |

Vector0-31: external interrupts, 8-level programmable priority. Interrupt source module is the same to vector16-47 of Cortex-M0+.

Vector32-35: internal exceptions, fixed priority, the lower the vector number, the higher the priority.

The priority of the internal exception is higher than the external interrupts.

# Chapter 4
# Peripheral Bridge (AIPS-Lite)

## 4.1 Chip-specific AIPS-Lite information

This device contains two peripheral bridges.

### 4.1.1 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See AIPS0 Memory Map and AIPS1 Memory Map for the memory slot assignment for each module.

## 4.2 Memory map/register definition

The AIPS modules on this device do not contain any user-programmable registers.

## 4.3 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

### 4.3.1 Access support

Aligned and misaligned 32-bit, 16-bit, and byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. All accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted that is larger than the targeted port, an error response is generated.

# Chapter 5
# Crossbar Switch (AXBS)

## 5.1 Chip-specific AXBS information

### 5.1.1 Crossbar switch master assignments - with system MPU

The masters connected to the crossbar switch are assigned as follows:

**Table 5-1. CM4F/RI5CY crossbar master assignment**

| Master port number | Master module | Default Priority |
|---|---|---|
| 0 | ARM/RISC-V core code/data bus | 4 |
| 1 | ARM/RISC-V core system bus | 3 |
| 2 | DMA0 | 2 |
| 3 | uSDHC | 1 |
| 4 | USB | 0 |
| 5 | CM0+/ZERO-RISCY Domain Crossbar S3 | 5 |

**Table 5-2. CM0+/ZERO-RISCY crossbar master assignment**

| Master port number | Master module | Default Priority |
|---|---|---|
| 0 | ARM/RISC-V core unified bus | 2 |
| 1 | DMA1 | 1 |
| 2 | CAU3 | 0 |
| 3 | CM4F/RI5CY Domain Crossbar S4 | 4 |
| 4 | ZERO-RISCY data bus | 3 |

### 5.1.2 Crossbar switch slave assignments - with system MPU

The slaves connected to the crossbar switch are assigned as follows:

**Table 5-3. CM4F crossbar slave assignment**

| Slave port number | Slave module |
|---|---|
| 0 | Flash Memory Controller |
| 1 | SRAM backdoor/Boot ROM |
| 2 | Peripheral bridge 0 |
| 3 | FlexBus |
| 4 | CM0+ Domain Crossbar M3 |

**Table 5-4. CM0+ crossbar slave assignment**

| Slave port number | Slave module |
|---|---|
| 0 | Flash Memory Controller |
| 1 | SRAM Controller |
| 2 | Peripheral bridge 1/USB SRAM/RGPIO A~D |
| 3 | CM4F Domain Crossbar M5 |

## 5.2 Introduction

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

### 5.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave-by-slave basis

- Up to single-clock 32-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support

## 5.3 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

**NOTE**

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

### 5.3.1 AXBS0 register descriptions

#### 5.3.1.1 AXBS0 Memory map

AXBS0 base address: 4000_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Priority Slave Registers (PRS0) | 32 | RW | 0050_1234h |
| 10h | Control Register (CRS0) | 32 | RW | 0000_0000h |
| 100h | Priority Slave Registers (PRS1) | 32 | RW | 0050_1234h |
| 110h | Control Register (CRS1) | 32 | RW | 0000_0000h |
| 200h | Priority Slave Registers (PRS2) | 32 | RW | 0050_1234h |
| 210h | Control Register (CRS2) | 32 | RW | 0000_0000h |
| 300h | Priority Slave Registers (PRS3) | 32 | RW | 0050_1234h |
| 310h | Control Register (CRS3) | 32 | RW | 0000_0000h |
| 400h | Priority Slave Registers (PRS4) | 32 | RW | 0050_1234h |
| 410h | Control Register (CRS4) | 32 | RW | 0000_0000h |
| 800h | Master General Purpose Control Register (MGPCR0) | 32 | RW | 0000_0000h |
| 900h | Master General Purpose Control Register (MGPCR1) | 32 | RW | 0000_0000h |
| A00h | Master General Purpose Control Register (MGPCR2) | 32 | RW | 0000_0000h |
| B00h | Master General Purpose Control Register (MGPCR3) | 32 | RW | 0000_0000h |
| C00h | Master General Purpose Control Register (MGPCR4) | 32 | RW | 0000_0000h |
| D00h | Master General Purpose Control Register (MGPCR5) | 32 | RW | 0000_0000h |

## 5.3.1.2 Priority Slave Registers (PRS0 - PRS4)

### 5.3.1.2.1 Offset

| Register | Offset |
|---|---|
| PRS0 | 0h |
| PRS1 | 100h |
| PRS2 | 200h |
| PRS3 | 300h |
| PRS4 | 400h |

### 5.3.1.2.2 Function

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

**NOTE**

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains fewer than three masters, only one bit is valid.
- If the chip contains fewer than five masters, only two bits are valid.
- If five or more masters are present, all three bits of the priority field are used.

### 5.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | Reserved | | | 0 | Reserved | | | 0 | M5 | | | 0 | M4 | | |
| W | | - | | | | - | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | M3 | | | 0 | M2 | | | 0 | M1 | | | 0 | M0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

### 5.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26-24<br>— | Reserved |
| 23<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 22-20<br>M5 | Master 5 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port.<br>111b - This master has level 8, or lowest, priority when accessing the slave port. |
| 19<br>— | Reserved |
| 18-16<br>M4 | Master 4 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port.<br>111b - This master has level 8, or lowest, priority when accessing the slave port. |
| 15<br>— | Reserved |
| 14-12<br>M3 | Master 3 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port.<br>111b - This master has level 8, or lowest, priority when accessing the slave port. |
| 11<br>— | Reserved |
| 10-8<br>M2 | Master 2 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port.<br>111b - This master has level 8, or lowest, priority when accessing the slave port. |
| 7<br>— | Reserved |
| 6-4<br>M1 | Master 1 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 111b - This master has level 8, or lowest, priority when accessing the slave port. |
| 3<br>— | Reserved |
| 2-0<br>M0 | Master 0 Priority. Sets the arbitration priority for this port on the associated slave port.<br>000b - This master has level 1, or highest, priority when accessing the slave port.<br>001b - This master has level 2 priority when accessing the slave port.<br>010b - This master has level 3 priority when accessing the slave port.<br>011b - This master has level 4 priority when accessing the slave port.<br>100b - This master has level 5 priority when accessing the slave port.<br>101b - This master has level 6 priority when accessing the slave port.<br>110b - This master has level 7 priority when accessing the slave port.<br>111b - This master has level 8, or lowest, priority when accessing the slave port. |

## 5.3.1.3 Control Register (CRS0 - CRS4)

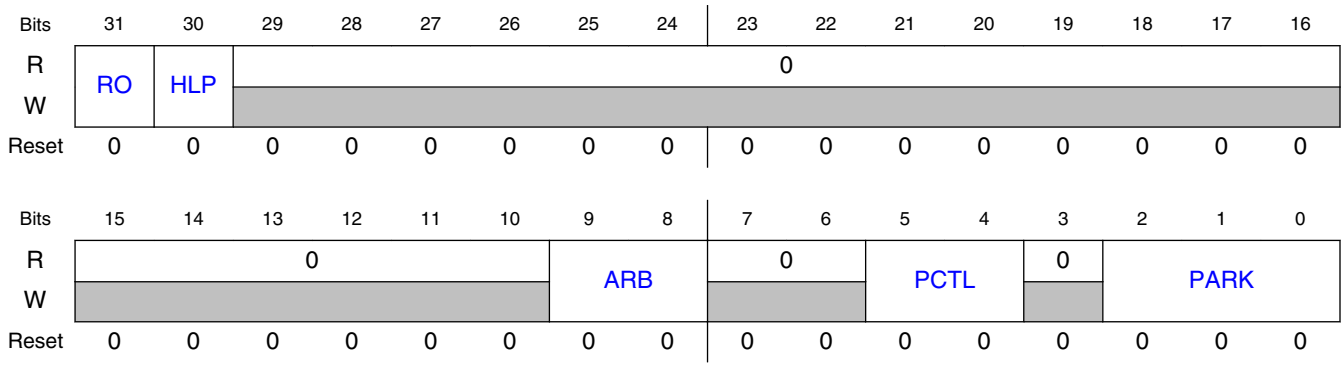
### 5.3.1.3.1 Offset

| Register | Offset |
|---|---|
| CRS0 | 10h |
| CRS1 | 110h |
| CRS2 | 210h |
| CRS3 | 310h |
| CRS4 | 410h |

### 5.3.1.3.2 Function

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

## 5.3.1.3.3 Diagram




## 5.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>RO | Read Only<br>Forces the slave port's CSRn and PRSn registers to be read-only. After set, only a hardware reset clears it.<br>0b - The slave port's registers are writeable<br>1b - The slave port's registers are read-only and cannot be written. Attempted writes have no effect on the registers and result in a bus error response. |
| 30<br>HLP | Halt Low Priority<br>Sets the initial arbitration priority for low power mode requests . Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports.<br>0b - The low power mode request has the highest priority for arbitration on this slave port<br>1b - The low power mode request has the lowest initial priority for arbitration on this slave port |
| 29-10<br>— | Reserved |
| 9-8<br>ARB | Arbitration Mode<br>Selects the arbitration policy for the slave port.<br>00b - Fixed priority<br>01b - Round-robin, or rotating, priority<br>10b - Reserved<br>11b - Reserved |
| 7-6<br>— | Reserved |
| 5-4<br>PCTL | Parking Control<br>Determines the slave port's parking' control. The low-power park feature results in an overall power savings if the slave port is not saturated. However, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master.<br>00b - When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK field<br>01b - When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port<br>10b - When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 11b - Reserved |
| 3<br>— | Reserved |
| 2-0<br>PARK | Park<br>Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared.<br>**NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur.<br>000b - Park on master port M0<br>001b - Park on master port M1<br>010b - Park on master port M2<br>011b - Park on master port M3<br>100b - Park on master port M4<br>101b - Park on master port M5<br>110b - Park on master port M6<br>111b - Park on master port M7 |

### 5.3.1.4 Master General Purpose Control Register (MGPCR0 - MGPCR5)

#### 5.3.1.4.1 Offset

| Register | Offset |
|---|---|
| MGPCR0 | 800h |
| MGPCR1 | 900h |
| MGPCR2 | A00h |
| MGPCR3 | B00h |
| MGPCR4 | C00h |
| MGPCR5 | D00h |

#### 5.3.1.4.2 Function

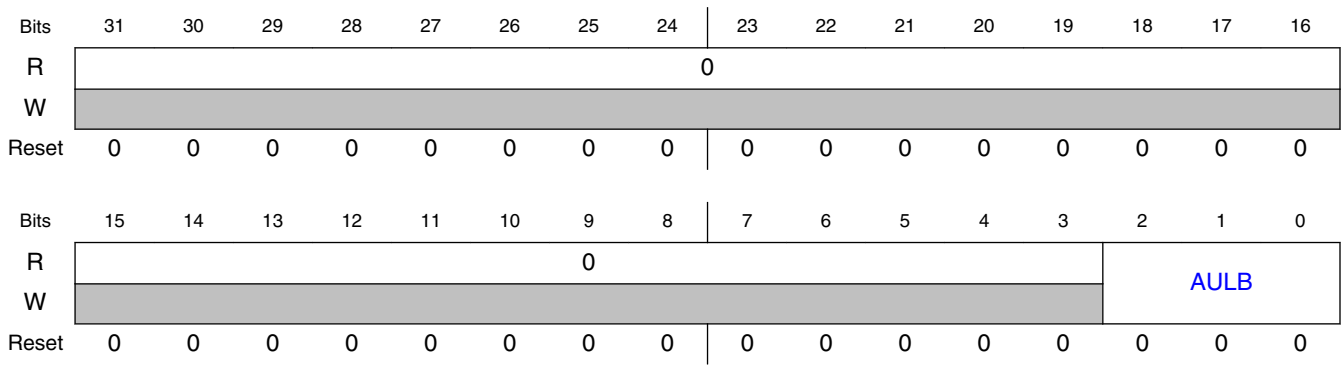
The MGPCR controls only whether the master's undefined length burst accesses are allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters. The MGPCR can be accessed only in Supervisor mode with 32-bit accesses.

**NOTE**

If there are fewer than eight master ports, only the registers associated with those masters are present. Register addresses

associated with master ports that are not present are reserved. Please see this module's chip-specific information for which master ports are present on this module.

#### 5.3.1.4.3 Diagram




#### 5.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>AULB | Arbitrates On Undefined Length Bursts<br>Determines whether, and when, the crossbar switch arbitrates away the slave port the master owns when the master is performing undefined length burst accesses.<br>000b - No arbitration is allowed during an undefined length burst<br>001b - Arbitration is allowed at any time during an undefined length burst<br>010b - Arbitration is allowed after four beats of an undefined length burst<br>011b - Arbitration is allowed after eight beats of an undefined length burst<br>100b - Arbitration is allowed after 16 beats of an undefined length burst<br>101b - Reserved<br>110b - Reserved<br>111b - Reserved |

## 5.4 Functional Description

Information about general operation and arbitration can be found here.

### 5.4.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed- or undefined-length burst transfer it retains control of the slave port until that transfer completes. Based on MGPCR[AULB], the master either retains control of the slave port when doing undefined-length, incrementing burst transfers or loses the bus to a higher-priority master.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK] . This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

### 5.4.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

The MGPCRx[AULB] bits are the exception to this rule. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the slave bus cycle to write them will have already terminated successfully. If the MGPCRx[AULB] bits are written between two burst accesses, the new AULB encodings do not take effect until an IDLE cycle is initiated by the master on that master port.

## 5.4.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

### 5.4.3.1 Arbitration during undefined length bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR[AULB] field setting. When a defined length is imposed on the burst via the AULB bits, the undefined length burst is treated as a single or series of single back-to-back fixed-length burst accesses.

The following figure illustrates an example:

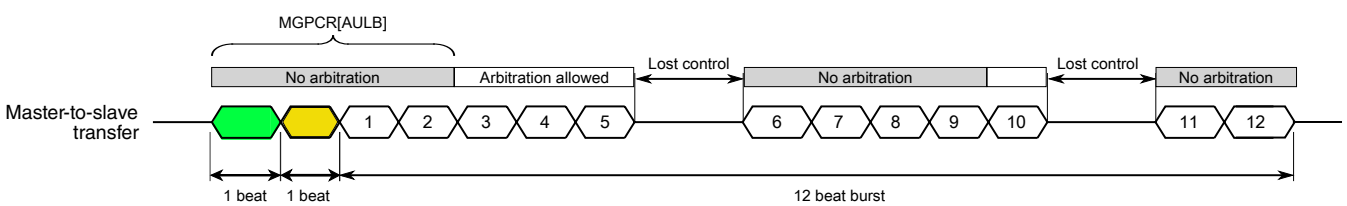



**Figure 5-1. Undefined length burst example**

In this example, a master runs an undefined length burst and the MGPCR[AULB] bits indicate arbitration occurs after the fourth beat of the burst. The master runs two sequential beats and then starts what will be a 12-beat undefined length burst access to a new address within the same slave port region as the previous access. The crossbar does not allow an arbitration point until the fourth overall access, or the second beat of the second burst. At that point, all remaining accesses are open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. After the master regains control of the slave port no arbitration point is available until after the master has run four more beats of its burst. After the fourth beat of the now

continued burst, or the ninth beat of the second burst from the master's perspective, is taken, all beats of the burst are once again open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third now continued burst, or the 10th beat of the second burst from the master's perspective. After the master regains control of the slave port, it is allowed to complete its final two beats of its burst without facing arbitration.

**Note**

Fixed-length burst accesses are not affected by the AULB bits. All fixed-length burst accesses lock out arbitration until the last beat of the fixed-length burst.

### 5.4.3.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

**NOTE**

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

**Table 5-5. How the Crossbar Switch grants control of a slave port to a master**

| When | Then the Crossbar Switch grants control to the requesting master |
|---|---|
| Both of the following are true:<br>• The current master is not running a transfer.<br>• The new requesting master's priority level is higher than that of the current master. | At the next clock edge |
| Both of the following are true: | At the end of the burst transfer or locked transfer |

*Table continues on the next page...*

**Table 5-5. How the Crossbar Switch grants control of a slave port to a master (continued)**

| When | Then the Crossbar Switch grants control to the requesting master |
|---|---|
| • The current master is running a fixed length burst transfer or a locked transfer.<br>• The requesting master's priority level is higher than that of the current master. | |
| Both of the following are true:<br>• The current master is running an undefined length burst transfer.<br>• The requesting master's priority level is higher than that of the current master. | At the next arbitration point for the undefined length burst transfer<br><br>**NOTE:** Arbitration points for an undefined length burst are defined in the MGPCR for each master. |
| The requesting master's priority level is lower than the current master. | At the conclusion of one of the following cycles:<br>• An IDLE cycle<br>• A non-IDLE cycle to a location other than the current slave port |

### 5.4.3.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

#### 5.4.3.4 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

## 5.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures that all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

**NOTE**

- During configuration of the crossbar switch, all other masters must be idle.
- To prevent reconfiguration of the crossbar switch, write 1 to CRS*n*[RO].

# Chapter 6
# Miscellaneous Control Module for CM4F (MCM0)

## 6.1 Introduction

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

### 6.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration and revision
- Error status and interrupts for the core's floating-point unit (FPU)

## 6.2 Memory map/register descriptions

The memory map and register descriptions below describe the registers using byte addresses.

### 6.2.1 MCM register descriptions

#### 6.2.1.1 MCM Memory map

MCM0 base address: E008_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | Crossbar Switch (AXBS) Slave Configuration (PLASC) | 16 | RO | 001Fh |
| Ah | Crossbar Switch (AXBS) Master Configuration (PLAMC) | 16 | RO | 003Fh |
| Ch | Core Platform Control Register (CPCR) | 32 | RW | 0000_0000h |
| 10h | Interrupt Status and Control Register (ISCR) | 32 | RW | 0000_0000h |
| 34h | Core Platform Control Register 2 (CPCR2) | 32 | RW | 0000_0030h |
| 40h | Compute Operation Control Register (CPO) | 32 | RW | 0000_0000h |

## 6.2.1.2 Crossbar Switch (AXBS) Slave Configuration (PLASC)

### 6.2.1.2.1 Offset

| Register | Offset |
|---|---|
| PLASC | 8h |

### 6.2.1.2.2 Function

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

### 6.2.1.2.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | ASC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

### 6.2.1.2.4 Fields

| Field | Function |
|---|---|
| 15-8<br>— | Reserved |
| 7-0<br>ASC | Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port.<br>0b - A bus slave connection to AXBS input port n is absent.<br>1b - A bus slave connection to AXBS input port n is present. |

## 6.2.1.3 Crossbar Switch (AXBS) Master Configuration (PLAMC)

### 6.2.1.3.1 Offset

| Register | Offset |
|---|---|
| PLAMC | Ah |

### 6.2.1.3.2 Function

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

### 6.2.1.3.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | AMC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

### 6.2.1.3.4 Fields

| Field | Function |
|---|---|
| 15-8<br>— | Reserved |
| 7-0<br>AMC | Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.<br>0b - A bus master connection to AXBS input port n is absent.<br>1b - A bus master connection to AXBS input port n is present. |

## 6.2.1.4 Core Platform Control Register (CPCR)

### 6.2.1.4.1 Offset

| Register | Offset |
|---|---|
| CPCR | Ch |

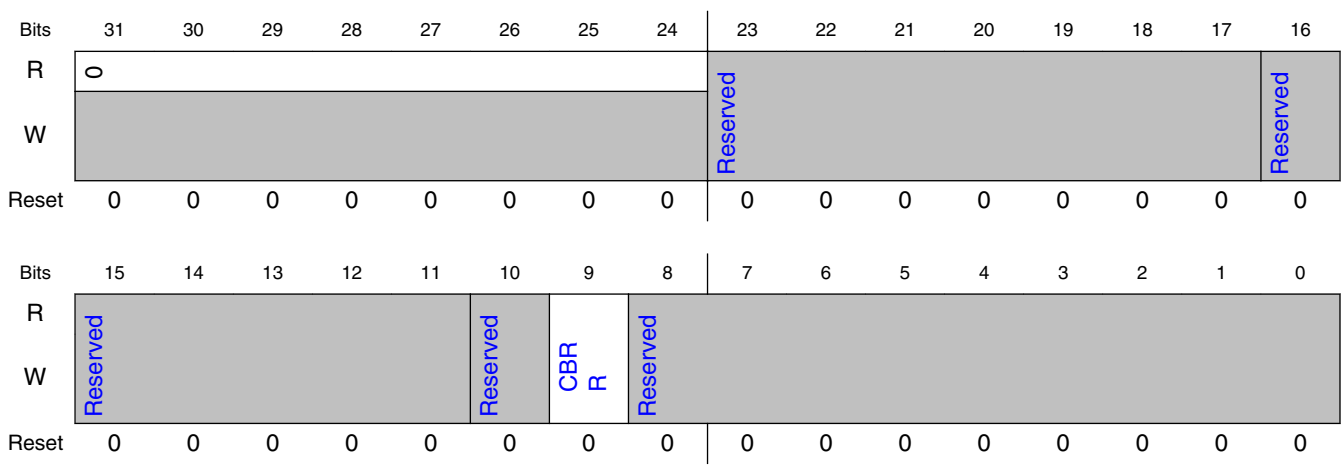
### 6.2.1.4.2 Function

Control Register defines the arbitration and protection schemes for the two system RAM arrays.

**NOTE**

Bits 23-22, 19-0 are undefined after reset.

### 6.2.1.4.3 Diagram




### 6.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-11<br>— | Reserved |
| 10 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 9<br>CBRR | Crossbar round-robin arbitration enable<br>Configures the crossbar slave ports to fixed-priority or round-robin arbitration.<br>0b - Fixed-priority arbitration<br>1b - Round-robin arbitration |
| 8-0<br>— | Reserved |

### 6.2.1.5 Interrupt Status and Control Register (ISCR)

#### 6.2.1.5.1 Offset

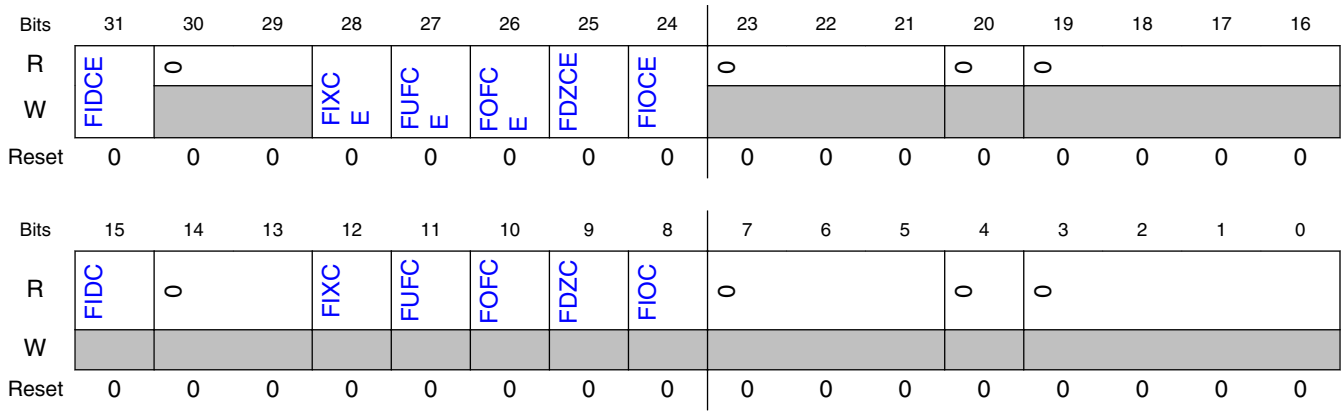
| Register | Offset |
|---|---|
| ISCR | 10h |

#### 6.2.1.5.2 Function

The MCM_ISCR register defines the configuration and reports status for a number of core-related interrupt exception conditions. It includes the enable and status bits associated with the core's floating-point exceptions, and bus errors associated with the core's cache write buffer. The individual event indicators are first qualified with their exception enables and then logically summed to form an interrupt request sent to the core's NVIC.

Bits 15-8 are read-only indicator flags based on the processor's FPSCR register. Attempted writes to these bits are ignored. Once set, the flags remain asserted until software clears the corresponding FPSCR bit.

### 6.2.1.5.3 Diagram




### 6.2.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>FIDCE | FPU input denormal interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 30-29<br>— | Reserved |
| 28<br>FIXCE | FPU inexact interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 27<br>FUFCE | FPU underflow interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 26<br>FOFCE | FPU overflow interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 25<br>FDZCE | FPU divide-by-zero interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 24<br>FIOCE | FPU invalid operation interrupt enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| 23-21<br>— | Reserved |
| 20<br>— | Reserved |
| 19-16<br>— | Reserved |
| 15<br>FIDC | FPU input denormal interrupt status |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This read-only bit is a copy of the core's FPSCR[IDC] bit and signals input denormalized number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IDC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 14-13<br>— | Reserved |
| 12<br>FIXC | FPU inexact interrupt status<br>This read-only bit is a copy of the core's FPSCR[IXC] bit and signals an inexact number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IXC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 11<br>FUFC | FPU underflow interrupt status<br>This read-only bit is a copy of the core's FPSCR[UFC] bit and signals an underflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[UFC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 10<br>FOFC | FPU overflow interrupt status<br>This read-only bit is a copy of the core's FPSCR[OFC] bit and signals an overflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[OFC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 9<br>FDZC | FPU divide-by-zero interrupt status<br>This read-only bit is a copy of the core's FPSCR[DZC] bit and signals a divide by zero has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[DZC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 8<br>FIOC | FPU invalid operation interrupt status<br>This read-only bit is a copy of the core's FPSCR[IOC] bit and signals an illegal operation has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IOC] bit.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 7-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3-0<br>— | Reserved |

## 6.2.1.6 Core Platform Control Register 2 (CPCR2)

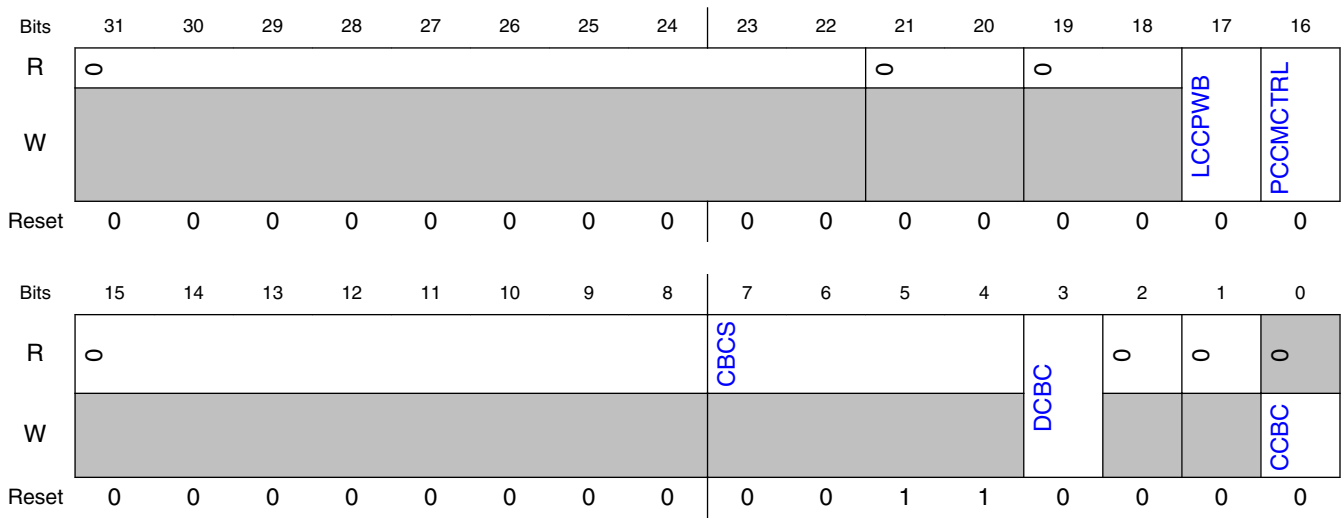
### 6.2.1.6.1 Offset

| Register | Offset |
|---|---|
| CPCR2 | 34h |

### 6.2.1.6.2 Function

Control Register 2 defines cache related properties.

### 6.2.1.6.3 Diagram




### 6.2.1.6.4 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |
| 21-20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17<br>LCCPWB | Limit code cache peripheral write buffering<br>0b - Code cache peripheral write buffering is not limited<br>1b - Code cache peripheral write buffering is limited |
| 16<br>PCCMCTRL | Bypass fixed code cache map<br>0b - The fixed code cache map is not bypassed<br>1b - The fixed code cache map is bypassed |
| 15-8 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 7-4<br>CBCS | Code Bus Cache Size<br>0000b - 0 KB<br>0001b - 1 KB<br>0010b - 2 KB<br>0011b - 4 KB<br>0100b - 8 KB<br>0101b - 16 KB<br>0110b - 32 KB |
| 3<br>DCBC | Disable code bus cache<br>0b - Enable code bus cache<br>1b - Disable code bus cache |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>CCBC | Clear code bus cache, this field always reads as 0.<br>0b - No effect<br>1b - Clear code bus cache |

## 6.2.1.7 Compute Operation Control Register (CPO)

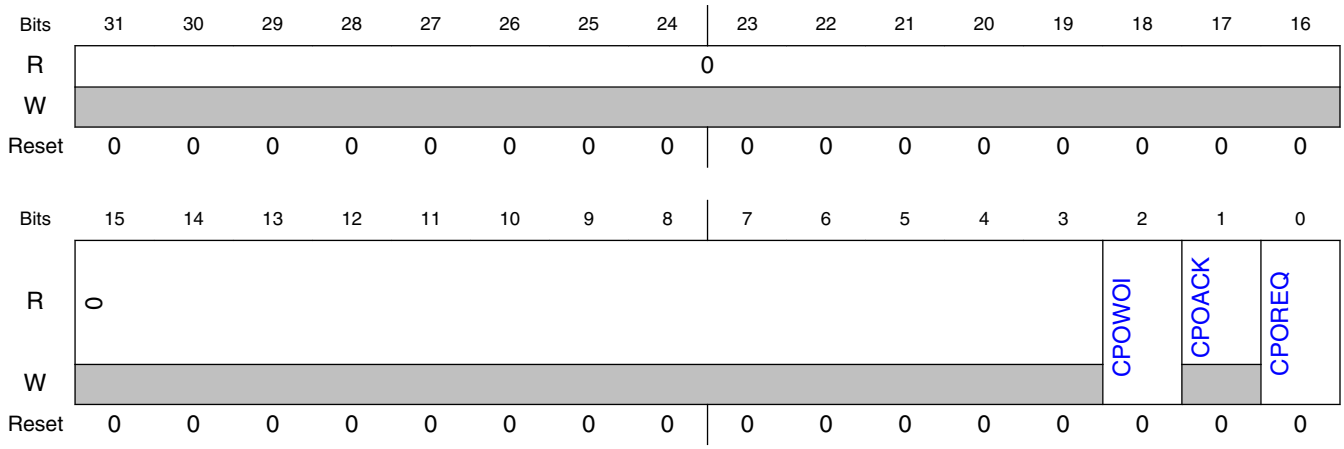
### 6.2.1.7.1 Offset

| Register | Offset |
|---|---|
| CPO | 40h |

### 6.2.1.7.2 Function

This register controls the Compute Operation.

#### 6.2.1.7.3 Diagram




#### 6.2.1.7.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>CPOWOI | Compute Operation wakeup on interrupt<br>0b - No effect.<br>1b - When set, the CPOREQ is cleared on any interrupt or exception vector fetch. |
| 1<br>CPOACK | Compute Operation acknowledge<br>0b - Compute operation entry has not completed or compute operation exit has completed.<br>1b - Compute operation entry has completed or compute operation exit has not completed. |
| 0<br>CPOREQ | Compute Operation request<br>This bit is auto-cleared by vector fetching if CPOWOI = 1.<br>0b - Request is cleared.<br>1b - Request Compute Operation. |

## 6.3 Functional description

This section describes the functional description of MCM module.

### 6.3.1 Interrupts

The MCM's interrupt is generated if any of the following is true:

- FPU input denormal interrupt is enabled (FIDCE) and an input is denormalized (FIDC)

- FPU inexact interrupt is enabled (FIXCE) and a number is inexact (FIXC)
- FPU underflow interrupt is enabled (FUFCE) and an underflow occurs (FUFC)
- FPU overflow interrupt is enabled (FOFCE) and an overflow occurs (FOFC)
- FPU divide-by-zero interrupt is enabled (FDZCE) and a divide-by-zero occurs (FDZC)
- FPU invalid operation interrupt is enabled (FIOCE) and an invalid occurs (FIOC)

### 6.3.2 Determining source of the interrupt

To determine the exact source of the interrupt qualify the interrupt status flags with the corresponding interrupt enable bits.

1. From MCM_ISCR[31:16] && MCM_ISCR[15:0]
2. Search the result for asserted flags, which indicate the exact interrupt sources

# Chapter 7
# Miscellaneous Control Module for CM0+ (MCM1)

## 7.1 Introduction

The Miscellaneous Control Module (MCM) provides miscellaneous control functions.

### 7.1.1 Features

The MCM includes the following features:

- Software-visible information on the platform configuration and revision

## 7.2 Memory map/register descriptions

The memory map and register descriptions below describe the registers using byte addresses.

### 7.2.1 MCM register descriptions

#### 7.2.1.1 MCM Memory map

MCM1 base address: F000_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | Crossbar Switch (AXBS) Slave Configuration (PLASC) | 16 | RO | 000Fh |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| Ah | Crossbar Switch (AXBS) Master Configuration (PLAMC) | 16 | RO | 000Fh |
| Ch | Core Platform Control Register (CPCR) | 32 | RW | 0000_0000h |
| 34h | Core Platform Control Register 2 (CPCR2) | 32 | RW | 0000_0030h |
| 40h | Compute Operation Control Register (CPO) | 32 | RW | 0000_0000h |

## 7.2.1.2 Crossbar Switch (AXBS) Slave Configuration (PLASC)

### 7.2.1.2.1 Offset

| Register | Offset |
|---|---|
| PLASC | 8h |

### 7.2.1.2.2 Function

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

### 7.2.1.2.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | ASC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

### 7.2.1.2.4 Fields

| Field | Function |
|---|---|
| 15-8<br>— | Reserved. |
| 7-0<br>ASC | Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port.<br>00000000b - A bus slave connection to AXBS input port n is absent<br>00000001b - A bus slave connection to AXBS input port n is present |

## 7.2.1.3 Crossbar Switch (AXBS) Master Configuration (PLAMC)

### 7.2.1.3.1 Offset

| Register | Offset |
|---|---|
| PLAMC | Ah |

### 7.2.1.3.2 Function

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

### 7.2.1.3.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | AMC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

### 7.2.1.3.4 Fields

| Field | Function |
|---|---|
| 15-8<br>— | Reserved. |
| 7-0<br>AMC | Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.<br>00000000b - A bus master connection to AXBS input port n is absent<br>00000001b - A bus master connection to AXBS input port n is present |

## 7.2.1.4 Core Platform Control Register (CPCR)

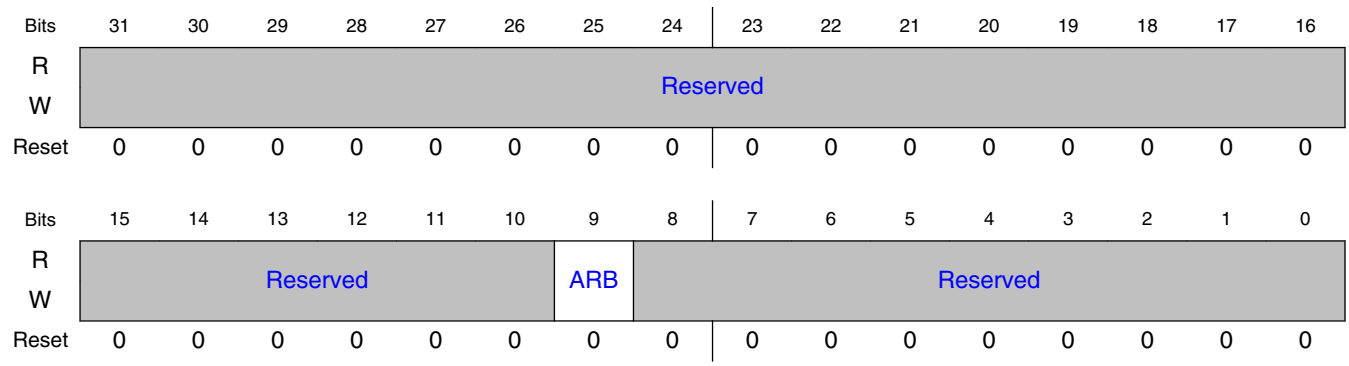
### 7.2.1.4.1 Offset

| Register | Offset |
|---|---|
| CPCR | Ch |

#### 7.2.1.4.2 Function

CPCR defines the arbitration scheme for the SRAM array.

#### 7.2.1.4.3 Diagram




#### 7.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9<br>ARB | Arbitration select<br>Defines the arbitration scheme.<br>0b - Fixed-priority arbitration for the crossbar masters<br>1b - Round-robin arbitration for the crossbar masters |
| 8-0<br>— | Reserved. |

### 7.2.1.5 Core Platform Control Register 2 (CPCR2)

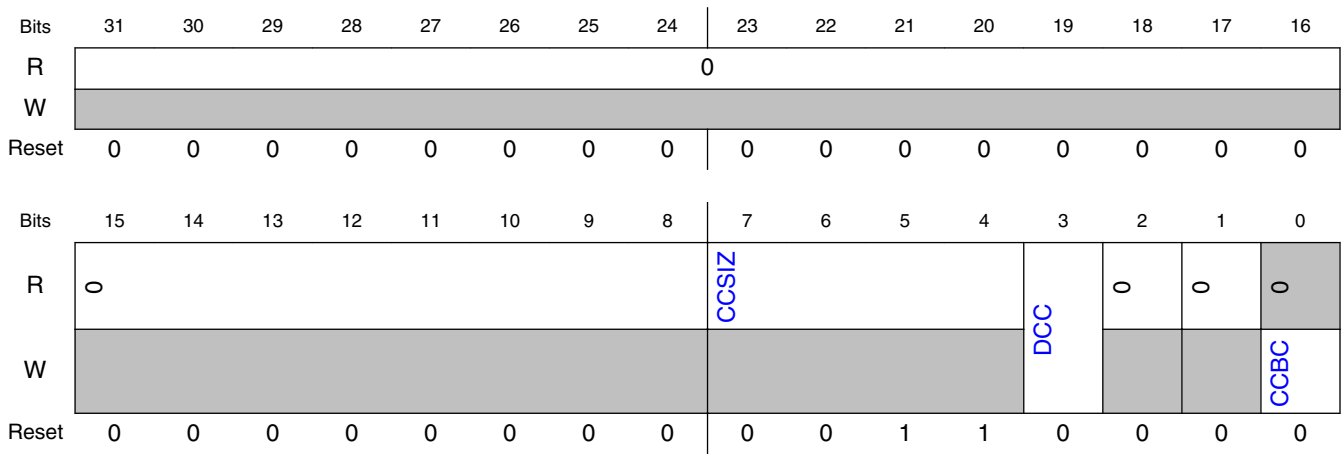
#### 7.2.1.5.1 Offset

| Register | Offset |
|---|---|
| CPCR2 | 34h |

### 7.2.1.5.2 Function

The CPCR2 register provides cache information and configuration options.

### 7.2.1.5.3 Diagram




### 7.2.1.5.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8<br>— | Reserved. |
| 7-4<br>CCSIZ | Code cache size<br>Defines the size of the code cache. Values not listed below are reserved.<br>0000b - No cache<br>0010b - 2KB<br>0011b - 4KB<br>0100b - 8KB |
| 3<br>DCC | Disable the code cache<br>Setting this bit disables the code cache. |
| 2<br>— | Reserved. |
| 1<br>— | Reserved. |
| 0<br>CCBC | Clear code bus cache<br>Writing a 1 to this field clears the cache. Writing a 0 to this field is ignored. This field always reads as 0. |

# 7.2.1.6 Compute Operation Control Register (CPO)

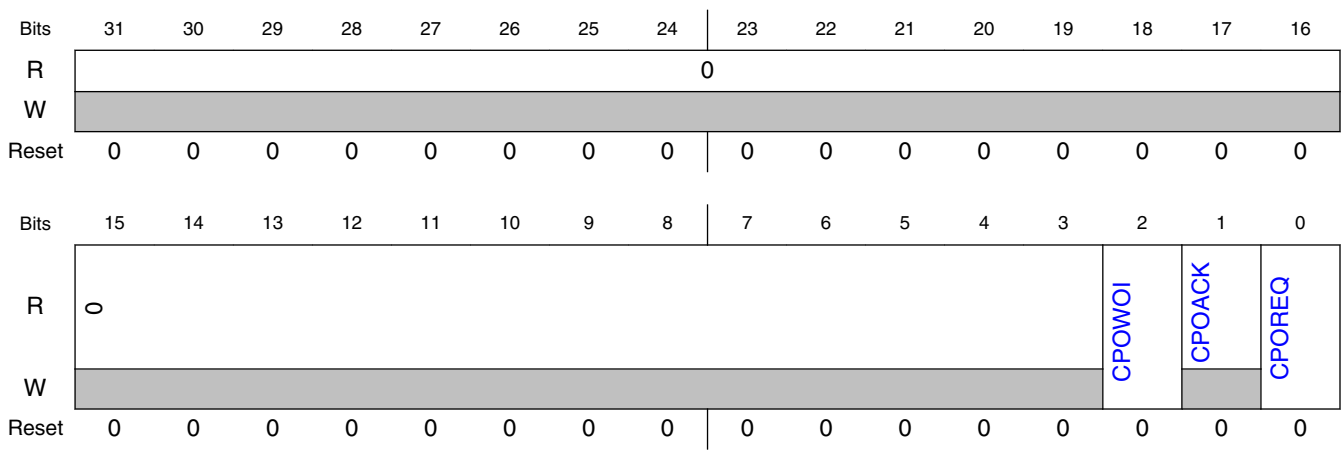
## 7.2.1.6.1 Offset

| Register | Offset |
|---|---|
| CPO | 40h |

## 7.2.1.6.2 Function

This register controls the Compute Operation.

## 7.2.1.6.3 Diagram




## 7.2.1.6.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2<br>CPOWOI | Compute Operation wakeup on interrupt<br>0b - No effect.<br>1b - When set, the CPOREQ is cleared on any interrupt or exception vector fetch. |
| 1<br>CPOACK | Compute Operation acknowledge<br>0b - Compute operation entry has not completed or compute operation exit has completed.<br>1b - Compute operation entry has completed or compute operation exit has not completed. |
| 0<br>CPOREQ | Compute Operation request<br>This bit is auto-cleared by vector fetching if CPOWOI = 1.<br>0b - Request is cleared.<br>1b - Request Compute Operation. |

# Chapter 8
# Miscellaneous System Control Module (MSCM)

## 8.1 Chip-specific MSCM information

The Core 0 and Core 1 in this chapter refer to the CPU0 (CM4F core) and CPU1 (CM0+ core) in other chapters.

## 8.2 Overview

The Miscellaneous System Control Module (MSCM) contains CPU configuration registers for Core 0 and Core 1, and on-chip memory controller registers.

## 8.3 Chip Configuration and Boot

The device's logical definition is controlled via chip-specific configuration bits, supported memory sizes and packing options. Collectively, these configuration bits define a reset configuration value (RCON).

Once the cores have fetched the needed reset vector(s), it is expected that core and system configuration information is read from a globally-accessible slave peripheral that properly converts the information into more appropriate values. More specifically, the cores access configuration information from a common set of peripheral addresses and the chip configuration logic properly evaluates based on the requesting processor and returns the appropriate value for the given processor, including core identification.

As an example, there is a single 32-bit read-only location for the core identification. A 32-bit read from this location returns a 4-character ASCII string: 0x434D3401("Cortex-M4").

The programming model associated with the core configuration information is included as part of the Miscellaneous System Control Module (MSCM). It specifically includes multiple views of the processor configuration; one view that is available generically to the cores, and other views that are available to any bus masters in the system.

## 8.4 MSCM Memory Map/Register Definition

### 8.4.1 CPU Configuration Memory Map and Registers

The CPU configuration portion of the MSCM module provides a set of memory-mapped read-only addresses defining the processor set-up. This portion of the MSCM programming model can only be accessed with privileged mode 32-bit read references; any other access type or size are terminated with an error. If the processor is logically not included in the chip configuration, then reads of its configuration registers return zeroes.

The CPU Configuration registers are organized based on the logical processor number (not any type of physical port number) and partitioned into the following equal sections:

**Table 8-1. CPU Configuration Register Sections**

| Offset addresses | Function |
|---|---|
| 0x000 - 0x01F | Defines the generic processor "x" configuration. This region is only accessible to the processor cores; reads by non-core bus masters are treated as read-as-zero (RAZ) accesses. |
| 0x020 - 0x03F | Defines the configuration information for processor 0 (CP0). This region is accessible to any bus master. |
| 0x040 - 0x05F | Defines the configuration information for processor 1 (CP1). This region is accessible to any bus master. For uniprocessor chip configurations, reads of this section are read as zero (RAZ). |

Attempted user mode or write accesses are terminated with an error.

### 8.4.2 MSCM register descriptions

#### 8.4.2.1 MSCM Memory map

MSCM base address: 4000_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Processor X Type Register (CPxTYPE) | 32 | RO | See description. |
| 4h | Processor X Number Register (CPxNUM) | 32 | RO | See description. |
| 8h | Processor X Master Register (CPxMASTER) | 32 | RO | See description. |
| Ch | Processor X Count Register (CPxCOUNT) | 32 | RO | 0000_0001h |
| 10h | Processor X Configuration Register 0 (CPxCFG0) | 32 | RO | See description. |
| 14h | Processor X Configuration Register 1 (CPxCFG1) | 32 | RO | See description. |
| 18h | Processor X Configuration Register 2 (CPxCFG2) | 32 | RO | See description. |
| 1Ch | Processor X Configuration Register 3 (CPxCFG3) | 32 | RO | See description. |
| 20h | Processor 0 Type Register (CP0TYPE) | 32 | RO | 434D_3401h |
| 24h | Processor 0 Number Register (CP0NUM) | 32 | RO | 0000_0000h |
| 28h | Processor 0 Master Register (CP0MASTER) | 32 | RO | 0000_0000h |
| 2Ch | Processor 0 Count Register (CP0COUNT) | 32 | RO | 0000_0001h |
| 30h | Processor 0 Configuration Register 0 (CP0CFG0) | 32 | RO | 0300_0000h |
| 34h | Processor 0 Configuration Register 1 (CP0CFG1) | 32 | RO | 0000_0000h |
| 38h | Processor 0 Configuration Register 2 (CP0CFG2) | 32 | RO | 0701_0901h |
| 3Ch | Processor 0 Configuration Register 3 (CP0CFG3) | 32 | RO | 0000_0121h |
| 40h | Processor 1 Type Register (CP1TYPE) | 32 | RO | 434D_3001h |
| 44h | Processor 1 Number Register (CP1NUM) | 32 | RO | 0000_0001h |
| 48h | Processor 1 Master Register (CP1MASTER) | 32 | RO | 0000_0020h |
| 4Ch | Processor 1 Count Register (CP1COUNT) | 32 | RO | 0000_0001h |
| 50h | Processor 1 Configuration Register 0 (CP1CFG0) | 32 | RO | 0300_0000h |
| 54h | Processor 1 Configuration Register 1 (CP1CFG1) | 32 | RO | 0000_0000h |
| 58h | Processor 1 Configuration Register 2 (CP1CFG2) | 32 | RO | 0801_0001h |
| 5Ch | Processor 1 Configuration Register 3 (CP1CFG3) | 32 | RO | 0000_0120h |
| 400h | On-Chip Memory Descriptor Register (OCMDR0) | 32 | RW | CB08_9000h |
| 404h | On-Chip Memory Descriptor Register (OCMDR1) | 32 | RW | C706_9000h |
| 408h | On-Chip Memory Descriptor Register (OCMDR2) | 32 | RW | C004_D000h |
| 40Ch | On-Chip Memory Descriptor Register (OCMDR3) | 32 | RW | D704_7000h |

## 8.4.2.2 Processor X Type Register (CPxTYPE)

#### 8.4.2.2.1 Offset

| Register | Offset |
|---|---|
| CPxTYPE | 0h |

#### 8.4.2.2.2 Function

The register provides a CPU-specific response indicating the personality of the core making the access. The 32-bit response includes 3 ASCII characters that define the CPU type, along with a byte that defines the logical revision number. The logical revision number follows ARM's rYpZ nomenclature.

**NOTE**

CPxTYPE value(s) for this device:

- If CPU0 is making the access = 0x434D3401
- If CPU1 is making the access = 0x434D3001
- If the read access is not from a CPU, then the value read is 0x00000000

#### 8.4.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | RYPZ | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 8.4.2.2.4 Fields

| Field | Function |
|---|---|
| 31-8<br>PERSONALITY | Processor x Personality<br>This read-only field defines the processor personality for CPUx<br>if CPUx = Cortex-M0+, then PERSONALITY = 0x43_4D_30 ("CM0+").<br>if CPUx = Cortex-M4, then PERSONALITY = 0x43_4D_34 ("CM4"). |
| 7-0 | Processor x Revision |

| Field | Function |
|---|---|
| RYPZ | This read-only field defines the processor revision for CPUx:<br>0x00 corresponds to the r0p0 core release.<br>0x01 corresponds to the r0p1 core release.<br>... |

## 8.4.2.3 Processor X Number Register (CPxNUM)

### 8.4.2.3.1 Offset

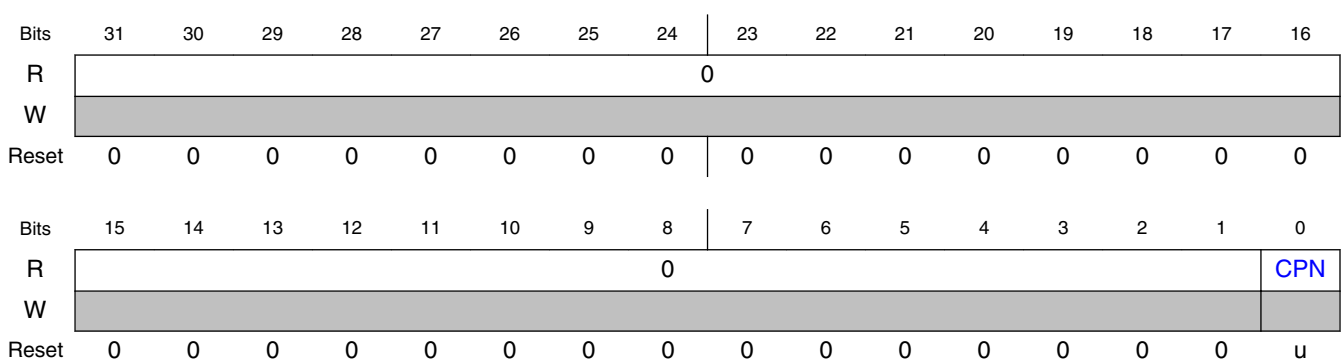
| Register | Offset |
|---|---|
| CPxNUM | 4h |

### 8.4.2.3.2 Function

The register provides a CPU-specific response indicating the logical processor number of the core making the access. The boot (or primary) core is assigned the number 0, while the secondary core is defined as number 1.

If the read access is not from a CPU, then the value read is 0x00000000.

### 8.4.2.3.3 Diagram




### 8.4.2.3.4 Fields

| Field | Function |
|---|---|
| 31-1 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 0<br>CPN | Processor x Number<br>This zero-filled word defines the logical processor number for CPUx<br>If boot (primary) core, then CPN = 0<br>For secondary core, CPN = 1 |

## 8.4.2.4 Processor X Master Register (CPxMASTER)

### 8.4.2.4.1 Offset

| Register | Offset |
|---|---|
| CPxMASTER | 8h |

### 8.4.2.4.2 Function

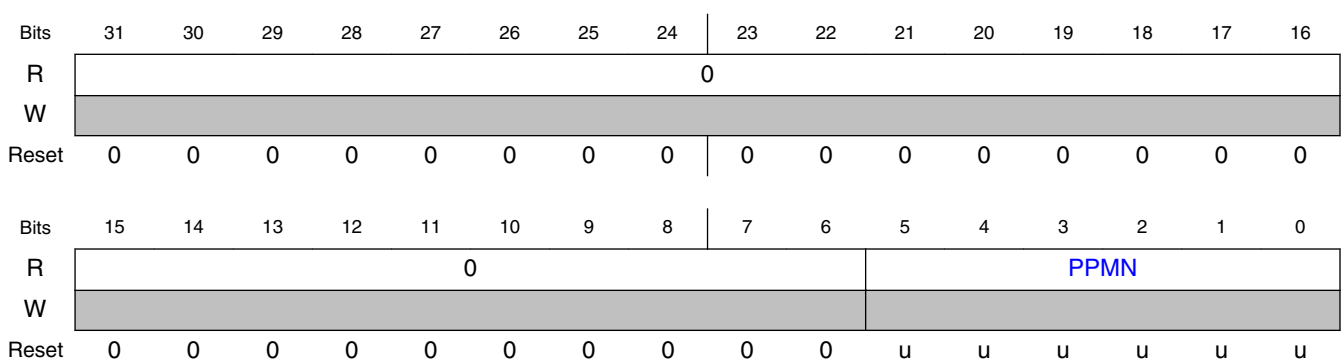
The register provides a CPU-specific response indicating the physical bus master number of the core that is making the access. The 32-bit response defines the physical master number for processor x.

- For CPU0, CPxMASTER = 0x00000000
- For CPU1, CPxMASTER = 0x00000020

**NOTE**

A privileged read from a CPU returns the appropriate processor information. Reads from a non-CPU bus master returns all zeroes. Attempted user mode or write accesses are terminated with an error.

### 8.4.2.4.3 Diagram



#### 8.4.2.4.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0<br>PPMN | Processor x Physical Master Number<br>This read-only field defines the physical bus master number for CPUx.<br>For CPU0, PPMN = 0x00<br>For CPU1, PPMN = 0x20 |

### 8.4.2.5 Processor X Count Register (CPxCOUNT)

#### 8.4.2.5.1 Offset

| Register | Offset |
|---|---|
| CPxCOUNT | Ch |

#### 8.4.2.5.2 Function

The register indicates the total number of processor cores in the chip configuration.

#### 8.4.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | PCNT | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### 8.4.2.5.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>PCNT | Processor Count<br>This read-only field defines the processor count for the chip configuration:<br>PCNT = 01 (Dual Core) |

### 8.4.2.6 Processor X Configuration Register 0 (CPxCFG0)

#### 8.4.2.6.1 Offset

| Register | Offset |
|---|---|
| CPxCFG0 | 10h |

#### 8.4.2.6.2 Function

The CPxCFG0 register provides a CPU-specific response detailing configuration information, in this case, information on the Level 1 caches (if present).

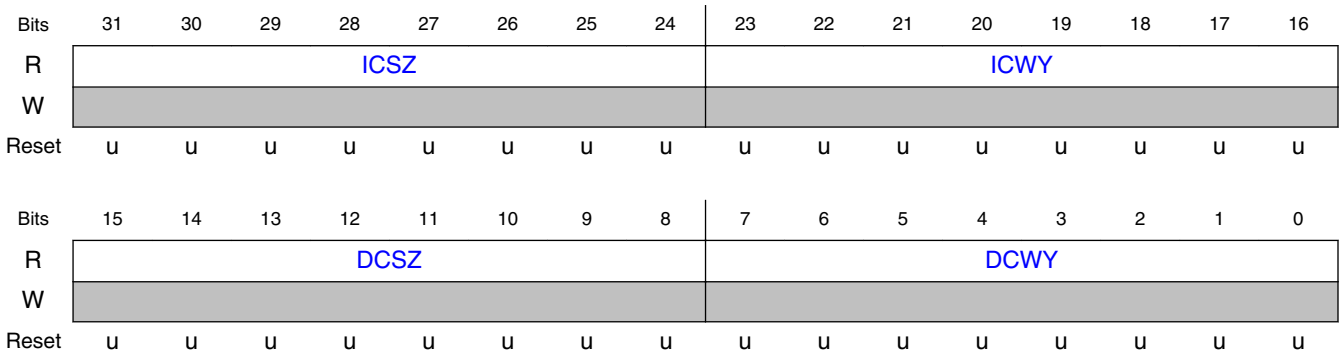
Access: Privileged read-only

**NOTE**

Reset values for the Processor X Configuration Register 0:

- For CPU0 - CPxCFG0 = 0x03000000
- For CPU1 - CPxCFG0 = 0x03000000
- If the read access is not from a CPU, then the value read is 0x00000000

### 8.4.2.6.3 Diagram




### 8.4.2.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>ICSZ | Level 1 Instruction Cache Size<br>This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+ICSZ)}$, where ICSZ is non-zero; a ICSZ = 0 indicates the memory is not present.<br>• if no Instruction Cache, then ICSZ = 0x00<br>• if a 1 Kbyte Instruction Cache, then ICSZ = 0x01<br>• if a 2 Kbyte Instruction Cache, then ICSZ = 0x02<br>• if a 4 Kbyte Instruction Cache, then ICSZ = 0x03<br>• if an 8 Kbyte Instruction Cache, then ICSZ = 0x04<br>• if a 16 Kbyte Instruction Cache, then ICSZ = 0x05<br>• if a 32 Kbyte Instruction Cache, then ICSZ = 0x06<br>• if a 64 Kbyte Instruction Cache, then ICSZ = 0x07<br>• if a 128 Kbyte Instruction Cache, then ICSZ = 0x08 |
| 23-16<br>ICWY | Level 1 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. ICWY=0x00 indicates not present. |
| 15-8<br>DCSZ | Level 1 Data Cache Size<br>This read-only field provides an encoded value of the Data Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+DCSZ)}$ , where DCSZ is non-zero; a DCSZ = 0 indicates the memory is not present.<br>• if no Data Cache, then DCSZ = 0x00<br>• if a 1 Kbyte Data Cache, then DCSZ = 0x01<br>• if a 2 Kbyte Data Cache, then DCSZ = 0x02<br>• if a 4 Kbyte Data Cache, then DCSZ = 0x03<br>• if an 8 Kbyte Data Cache, then DCSZ = 0x04<br>• if a 16 Kbyte Data Cache, then DCSZ = 0x05<br>• if a 32 Kbyte Data Cache, then DCSZ = 0x06<br>• if a 64 Kbyte Data Cache, then DCSZ = 0x07<br>• if a 128 Kbyte Data Cache, then DCSZ = 0x08 |
| 7-0<br>DCWY | Level 1 Data Cache Ways<br>This read-only field provides the number of cache ways for the Data Cache. DCWY=0x00 indicates not present. |

## 8.4.2.7 Processor X Configuration Register 1 (CPxCFG1)

### 8.4.2.7.1 Offset

| Register | Offset |
|---|---|
| CPxCFG1 | 14h |

### 8.4.2.7.2 Function

The CPxCFG1 register provides a CPU-specific response detailing configuration information, in this case, information on a Level 2 cache (if present).

Access: Privileged read-only

**NOTE**

Reset values for the Processor X Configuration Register 1:

- For CPU0 - CPxCFG1 = 0x00000000
- For CPU1 - CPxCFG1 = 0x00000000
- If the read access is not from a CPU, then the value read is 0x00000000

### 8.4.2.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L2SZ | | | | | | | | L2WY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 8.4.2.7.4 Fields

| Field | Function |
|---|---|
| 31-24<br>L2SZ | Level 2 Instruction Cache Size |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+L2SZ)}$, where L2SZ is non-zero; a L2SZ = 0 indicates the memory is not present.<br>• if no Level 2 Cache, then L2SZ = 0x00<br>• if a 1 Kbyte Level 2 Cache, then L2SZ = 0x01<br>• if a 2 Kbyte Level 2 Cache, then L2SZ = 0x02<br>• if a 4 Kbyte Level 2 Cache, then L2SZ = 0x03<br>• if an 8 Kbyte Level 2 Cache, then L2SZ = 0x04<br>• if a 16 Kbyte Level 2 Cache, then L2SZ = 0x05<br>• if a 32 Kbyte Level 2 Cache, then L2SZ = 0x06<br>• if a 64 Kbyte Level 2 Cache, then L2SZ = 0x07<br>• if a 128 Kbyte Level 2 Cache, then L2SZ = 0x08 |
| 23-16<br>L2WY | Level 2 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. L2WY=0x00 indicates not present. |
| 15-0<br>— | Reserved |

## 8.4.2.8 Processor X Configuration Register 2 (CPxCFG2)

### 8.4.2.8.1 Offset

| Register | Offset |
|---|---|
| CPxCFG2 | 18h |

### 8.4.2.8.2 Function

The CPxCFG2 register provides a CPU-specific response detailing configuration information, in this case, information on tightly-coupled local memories (if present).

**NOTE**

Reset values for the Processor X Configuration Register 2:
- For CPU0 - CPxCFG2 = 0x07010901
- For CPU1 - CPxCFG2 = 0x08010001
- If the read access is not from a CPU, then the value read is 0x00000000

## 8.4.2.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMLSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMUSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 8.4.2.8.4 Fields

| Field | Function |
|---|---|
| 31-24<br>TMLSZ | Tightly-coupled Memory Lower Size<br>This field provides an encoded value of the tightly-coupled local memory lower size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMLSZ)}$, where TMLSZ is non-zero; a TMLSZ = 0 indicates the memory is not present.<br>• if no TCML, then TMLSZ = 0x00<br>• if a 1 Kbyte TCML, then TMLSZ = 0x01<br>• if a 2 Kbyte TCML, then TMLSZ = 0x02<br>• if a 4 Kbyte TCML, then TMLSZ = 0x03<br>• if an 8 Kbyte TCML, then TMLSZ = 0x04<br>• if a 16 Kbyte TCML, then TMLSZ = 0x05<br>• if a 32 Kbyte TCML, then TMLSZ = 0x06<br>• if a 64 Kbyte TCML, then TMLSZ = 0x07<br>• if a 128 Kbyte TCML, then TMLSZ = 0x08<br>• if a 256 Kbyte TCML, then TMLSZ = 0x09 |
| 23-16<br>— | Reserved. |
| 15-8<br>TMUSZ | Tightly-coupled Memory Upper Size<br>This field provides an encoded value of the tightly-coupled local memory upper size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMUSZ)}$, where TMUSZ is non-zero; a TMUSZ = 0 indicates the memory is not present.<br>• if no TCMU, then TMUSZ = 0x00<br>• if a 1 Kbyte TCMU, then TMUSZ = 0x01<br>• if a 2 Kbyte TCMU, then TMUSZ = 0x02<br>• if a 4 Kbyte TCMU, then TMUSZ = 0x03<br>• if an 8 Kbyte TCMU, then TMUSZ = 0x04<br>• if a 16 Kbyte TCMU, then TMUSZ = 0x05<br>• if a 32 Kbyte TCMU, then TMUSZ = 0x06<br>• if a 64 Kbyte TCMU, then TMUSZ = 0x07<br>• if a 128 Kbyte TCMU, then TMUSZ = 0x08<br>• if a 256 Kbyte TCMU, then TMUSZ = 0x09 |
| 7-0<br>— | Reserved. |

## 8.4.2.9 Processor X Configuration Register 3 (CPxCFG3)

### 8.4.2.9.1 Offset

| Register | Offset |
|---|---|
| CPxCFG3 | 1Ch |

### 8.4.2.9.2 Function

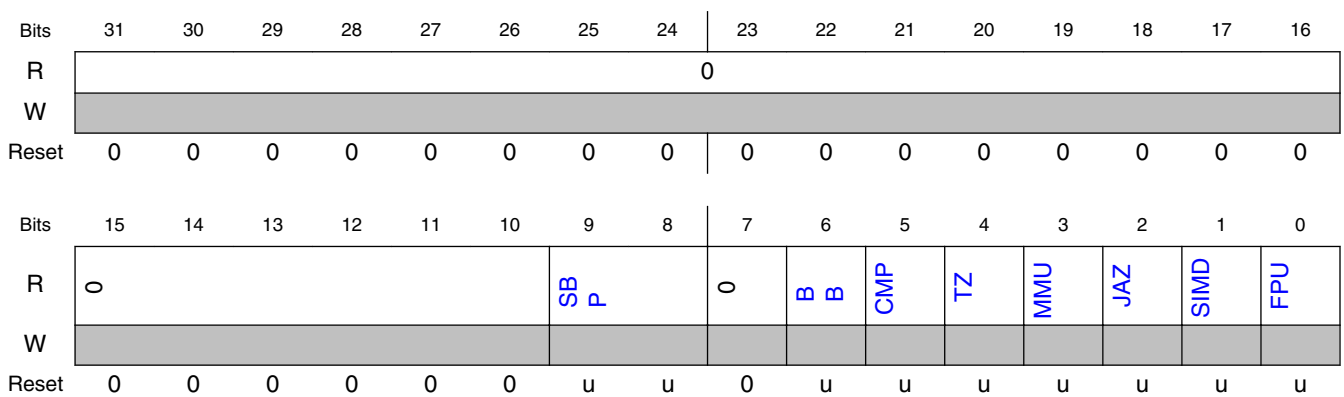
The CPxCFG3 register provides a CPU-specific response detailing configuration information, in this case, information on processor options.

**NOTE**

Reset values for the Processor X Configuration Register 3:

- For CPU0 - CPxCFG3 = 0x00000121
- For CPU1 - CPxCFG3 = 0x00000120
- If the read access is not from a CPU, then the value read is 0x00000000

### 8.4.2.9.3 Diagram




### 8.4.2.9.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 9-8<br>SBP | System Bus Ports<br>This field defines the number of physical connections to the system bus fabric for this processor. |
| 7<br>— | Reserved |
| 6<br>BB | Bit Banding<br>This field defines if the processor supports "bit banding".<br>0b - Bit Banding is not supported.<br>1b - Bit Banding is supported. |
| 5<br>CMP | Core Memory Protection unit<br>This field indicates if the core memory protection hardware is included in the processor.<br>0b - Core Memory Protection is not included.<br>1b - Core Memory Protection is included. |
| 4<br>TZ | Trust Zone<br>This field indicates if the Trust Zone capabilities are supported in the processor..<br>0b - Trust Zone support is not included.<br>1b - Trust Zone support is included. |
| 3<br>MMU | Memory Management Unit<br>Memory Management Unit. This field indicates if the virtual memory management capabilities are supported in the processor.<br>0b - MMU support is not included.<br>1b - MMU support is included. |
| 2<br>JAZ | Jazelle support<br>This field indicates if Jazelle hardware is supported in the processor.<br>0b - Jazelle support is not included.<br>1b - Jazelle support is included. |
| 1<br>SIMD | SIMD/NEON instruction support<br>This field indicates if the instruction set extensions supporting SIMD and/or NEON capabilities are supported in the processor.<br>0b - SIMD/NEON support is not included.<br>1b - SIMD/NEON support is included. |
| 0<br>FPU | Floating Point Unit<br>This field indicates if hardware support for floating point capabilities are supported in the processor.<br>0b - FPU support is not included.<br>1b - FPU support is included. |

## 8.4.2.10 Processor 0 Type Register (CP0TYPE)

### 8.4.2.10.1 Offset

| Register | Offset |
|---|---|
| CP0TYPE | 20h |

### 8.4.2.10.2 Function

The register provides the personality of Processor 0. The 32-bit response includes 3 ASCII characters defining the CPU type, along with a byte defining the logical revision number. The logical revision number follows ARM's rYpZ nomenclature.

### 8.4.2.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | RYPZ | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 8.4.2.10.4 Fields

| Field | Function |
|---|---|
| 31-8<br>PERSONALITY | Processor 0 Personality<br>This read-only field defines the processor personality for CP0<br>CP0 = Cortex-M4, then PERSONALITY = 0x43_4D_34 ("CM4"). |
| 7-0<br>RYPZ | Processor 0 Revision<br>This read-only field defines the processor revision for CPU0:<br>0x00 corresponds to the r0p0 core release.<br>0x01 corresponds to the r0p1 core release.<br>... |

## 8.4.2.11 Processor 0 Number Register (CP0NUM)

### 8.4.2.11.1 Offset

| Register | Offset |
|---|---|
| CP0NUM | 24h |

### 8.4.2.11.2 Function

The register provides the logical processor number of Processor 0. The boot (or primary) core is assigned the number 0, while the secondary core is defined as number 1.

### 8.4.2.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | CPN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 8.4.2.11.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>CPN | Processor 0 Number<br>This zero-filled word defines the logical processor number for CPU0<br>If boot (primary) core, then CPN = 0<br>For secondary core, CPN = 1 |

## 8.4.2.12 Processor 0 Master Register (CP0MASTER)

#### 8.4.2.12.1 Offset

| Register | Offset |
|---|---|
| CP0MASTER | 28h |

#### 8.4.2.12.2 Function

The register provides the physical bus master number of Processor 0.

#### 8.4.2.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | PPMN | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 8.4.2.12.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0<br>PPMN | Processor 0 Physical Master Number<br>This read-only field defines the physical bus master number for CPU0 |

### 8.4.2.13 Processor 0 Count Register (CP0COUNT)

#### 8.4.2.13.1 Offset

| Register | Offset |
|---|---|
| CP0COUNT | 2Ch |

### 8.4.2.13.2 Function

The register indicates the total number of processor cores in the chip configuration.

### 8.4.2.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | PCNT | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 8.4.2.13.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>PCNT | Processor Count<br>This read-only field defines the processor count for the chip configuration:<br>PCNT = 01 (Dual Core) |

## 8.4.2.14 Processor 0 Configuration Register 0 (CP0CFG0)

### 8.4.2.14.1 Offset

| Register | Offset |
|---|---|
| CP0CFG0 | 30h |

### 8.4.2.14.2 Function

The CP0CFG0 register provides information on the CPU0 Level 1 caches (if present).

Access: Privileged read-only

**NOTE**

Reset values for the Processor 0 Configuration Register 0:

- CP0CFG0 = 0x03000000

### 8.4.2.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ICSZ | | | | | | | | ICWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCSZ | | | | | | | | DCWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 8.4.2.14.4 Fields

| Field | Function |
|---|---|
| 31-24<br>ICSZ | Level 1 Instruction Cache Size<br>This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+ICSZ)}$, where ICSZ is non-zero; a ICSZ = 0 indicates the memory is not present.<br>• if no Instruction Cache, then ICSZ = 0x00<br>• if a 1 Kbyte Instruction Cache, then ICSZ = 0x01<br>• if a 2 Kbyte Instruction Cache, then ICSZ = 0x02<br>• if a 4 Kbyte Instruction Cache, then ICSZ = 0x03<br>• if an 8 Kbyte Instruction Cache, then ICSZ = 0x04<br>• if a 16 Kbyte Instruction Cache, then ICSZ = 0x05<br>• if a 32 Kbyte Instruction Cache, then ICSZ = 0x06<br>• if a 64 Kbyte Instruction Cache, then ICSZ = 0x07<br>• if a 128 Kbyte Instruction Cache, then ICSZ = 0x08 |
| 23-16<br>ICWY | Level 1 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. ICWY=0x00 indicates not present. |
| 15-8<br>DCSZ | Level 1 Data Cache Size<br>This read-only field provides an encoded value of the Data Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+DCSZ)}$ , where DCSZ is non-zero; a DCSZ = 0 indicates the memory is not present.<br>• if no Data Cache, then DCSZ = 0x00<br>• if a 1 Kbyte Data Cache, then DCSZ = 0x01<br>• if a 2 Kbyte Data Cache, then DCSZ = 0x02<br>• if a 4 Kbyte Data Cache, then DCSZ = 0x03<br>• if an 8 Kbyte Data Cache, then DCSZ = 0x04<br>• if a 16 Kbyte Data Cache, then DCSZ = 0x05 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • if a 32 Kbyte Data Cache, then DCSZ = 0x06<br>• if a 64 Kbyte Data Cache, then DCSZ = 0x07<br>• if a 128 Kbyte Data Cache, then DCSZ = 0x08 |
| 7-0<br>DCWY | Level 1 Data Cache Ways<br>This read-only field provides the number of cache ways for the Data Cache. DCWY=0x00 indicates not present. |

## 8.4.2.15 Processor 0 Configuration Register 1 (CP0CFG1)

### 8.4.2.15.1 Offset

| Register | Offset |
|---|---|
| CP0CFG1 | 34h |

### 8.4.2.15.2 Function

The CP0CFG1 register provides information on CPU0 Level 2 cache (if present).

Access: Privileged read-only

**NOTE**

Reset values for the Processor 0 Configuration Register 1:
- CP0CFG1 = 0x00000000

### 8.4.2.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L2SZ | | | | | | | | L2WY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 8.4.2.15.4 Fields

| Field | Function |
|---|---|
| 31-24<br>L2SZ | Level 2 Instruction Cache Size<br>This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+L2SZ)}$, where L2SZ is non-zero; a L2SZ = 0 indicates the memory is not present.<br>• if no Level 2 Cache, then L2SZ = 0x00<br>• if a 1 Kbyte Level 2 Cache, then L2SZ = 0x01<br>• if a 2 Kbyte Level 2 Cache, then L2SZ = 0x02<br>• if a 4 Kbyte Level 2 Cache, then L2SZ = 0x03<br>• if an 8 Kbyte Level 2 Cache, then L2SZ = 0x04<br>• if a 16 Kbyte Level 2 Cache, then L2SZ = 0x05<br>• if a 32 Kbyte Level 2 Cache, then L2SZ = 0x06<br>• if a 64 Kbyte Level 2 Cache, then L2SZ = 0x07<br>• if a 128 Kbyte Level 2 Cache, then L2SZ = 0x08 |
| 23-16<br>L2WY | Level 2 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. L2WY=0x00 indicates not present. |
| 15-0<br>— | Reserved |

### 8.4.2.16 Processor 0 Configuration Register 2 (CP0CFG2)

#### 8.4.2.16.1 Offset

| Register | Offset |
|---|---|
| CP0CFG2 | 38h |

#### 8.4.2.16.2 Function

The CP0CFG2 register provides information on CPU0 tightly-coupled local memories (if present).

**NOTE**

Reset values for the Processor 0 Configuration Register 2:

- CP0CFG2 = 0x07010901

## 8.4.2.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMLSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMUSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 8.4.2.16.4 Fields

| Field | Function |
|---|---|
| 31-24<br>TMLSZ | Tightly-coupled Memory Lower Size<br>This field provides an encoded value of the tightly-coupled local memory lower size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMLSZ)}$, where TMLSZ is non-zero; a TMLSZ = 0 indicates the memory is not present.<br>• if no TCML, then TMLSZ = 0x00<br>• if a 1 Kbyte TCML, then TMLSZ = 0x01<br>• if a 2 Kbyte TCML, then TMLSZ = 0x02<br>• if a 4 Kbyte TCML, then TMLSZ = 0x03<br>• if an 8 Kbyte TCML, then TMLSZ = 0x04<br>• if a 16 Kbyte TCML, then TMLSZ = 0x05<br>• if a 32 Kbyte TCML, then TMLSZ = 0x06<br>• if a 64 Kbyte TCML, then TMLSZ = 0x07<br>• if a 128 Kbyte TCML, then TMLSZ = 0x08<br>• if a 256 Kbyte TCML, then TMLSZ = 0x09 |
| 23-16<br>— | Reserved. |
| 15-8<br>TMUSZ | Tightly-coupled Memory Upper Size<br>This field provides an encoded value of the tightly-coupled local memory upper size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMUSZ)}$, where TMUSZ is non-zero; a TMUSZ = 0 indicates the memory is not present.<br>• if no TCMU, then TMUSZ = 0x00<br>• if a 1 Kbyte TCMU, then TMUSZ = 0x01<br>• if a 2 Kbyte TCMU, then TMUSZ = 0x02<br>• if a 4 Kbyte TCMU, then TMUSZ = 0x03<br>• if an 8 Kbyte TCMU, then TMUSZ = 0x04<br>• if a 16 Kbyte TCMU, then TMUSZ = 0x05<br>• if a 32 Kbyte TCMU, then TMUSZ = 0x06<br>• if a 64 Kbyte TCMU, then TMUSZ = 0x07<br>• if a 128 Kbyte TCMU, then TMUSZ = 0x08<br>• if a 256 Kbyte TCMU, then TMUSZ = 0x09 |
| 7-0<br>— | Reserved. |

## 8.4.2.17 Processor 0 Configuration Register 3 (CP0CFG3)

### 8.4.2.17.1 Offset

| Register | Offset |
|---|---|
| CP0CFG3 | 3Ch |

### 8.4.2.17.2 Function

The CP0CFG3 register provides information on Processor 0 options.

**NOTE**

Reset values for the Processor 0 Configuration Register 3:

- CP0CFG3 = 0x00000121

### 8.4.2.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | SBP | | 0 | BB | CMP | TZ | MMU | JAZ | SIMD | FPU |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

### 8.4.2.17.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>SBP | System Bus Ports<br>This field defines the number of physical connections to the system bus fabric for this processor. |
| 7<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>BB | Bit Banding<br>This field defines if the processor supports "bit banding".<br>0b - Bit Banding is not supported.<br>1b - Bit Banding is supported. |
| 5<br>CMP | Core Memory Protection unit<br>This field indicates if the core memory protection hardware is included in the processor.<br>0b - Core Memory Protection is not included.<br>1b - Core Memory Protection is included. |
| 4<br>TZ | Trust Zone<br>This field indicates if the Trust Zone capabilities are supported in the processor..<br>0b - Trust Zone support is not included.<br>1b - Trust Zone support is included. |
| 3<br>MMU | Memory Management Unit<br>Memory Management Unit. This field indicates if the virtual memory management capabilities are supported in the processor.<br>0b - MMU support is not included.<br>1b - MMU support is included. |
| 2<br>JAZ | Jazelle support<br>This field indicates if Jazelle hardware is supported in the processor.<br>0b - Jazelle support is not included.<br>1b - Jazelle support is included. |
| 1<br>SIMD | SIMD/NEON instruction support<br>This field indicates if the instruction set extensions supporting SIMD and/or NEON capabilities are supported in the processor.<br>0b - SIMD/NEON support is not included.<br>1b - SIMD/NEON support is included. |
| 0<br>FPU | Floating Point Unit<br>This field indicates if hardware support for floating point capabilities are supported in the processor.<br>0b - FPU support is not included.<br>1b - FPU support is included. |

## 8.4.2.18 Processor 1 Type Register (CP1TYPE)

### 8.4.2.18.1 Offset

| Register | Offset |
|---|---|
| CP1TYPE | 40h |

### 8.4.2.18.2 Function

The register provides the personality of Processor 1. The 32-bit response includes 3 ASCII characters defining the CPU type, along with a byte defining the logical revision number. The logical revision number follows ARM's rYpZ nomenclature.

### 8.4.2.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0[1] | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | RYPZ | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0[1] | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

1. See bit field descriptions

### 8.4.2.18.4 Fields

| Field | Function |
|---|---|
| 31-8<br>PERSONALITY | Processor 1 Personality<br>This read-only field defines the processor personality for CP1<br>CP1 = Cortex-M0+, then PERSONALITY = 0x43_4D_30 ("CM0+"). |
| 7-0<br>RYPZ | Processor 1 Revision<br>This read-only field defines the processor revision for CPU1:<br>0x00 corresponds to the r0p0 core release.<br>0x01 corresponds to the r0p1 core release.<br>... |

## 8.4.2.19 Processor 1 Number Register (CP1NUM)

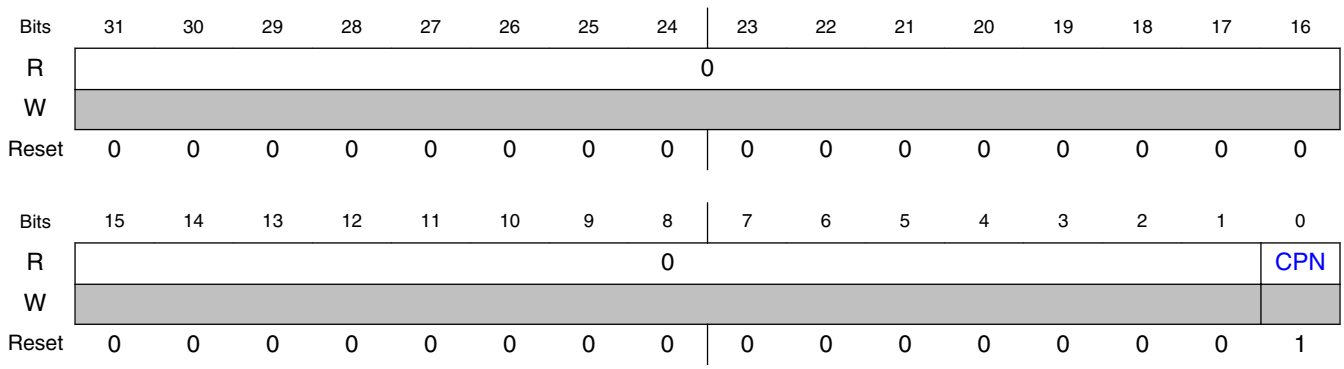
### 8.4.2.19.1 Offset

| Register | Offset |
|---|---|
| CP1NUM | 44h |

### 8.4.2.19.2 Function

The register provides the logical processor number of Processor 1. The boot (or primary) core is assigned the number 0, while the secondary core is defined as number 1.

### 8.4.2.19.3 Diagram




### 8.4.2.19.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>CPN | Processor 1 Number<br>This zero-filled word defines the logical processor number for CPU1<br>If boot (primary) core, then CPN = 0<br>For the secondary core, CPN = 1 |

## 8.4.2.20 Processor 1 Master Register (CP1MASTER)

### 8.4.2.20.1 Offset

| Register | Offset |
|---|---|
| CP1MASTER | 48h |

#### 8.4.2.20.2 Function

The register provides the physical bus master number of Processor 1.

#### 8.4.2.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | PPMN | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

#### 8.4.2.20.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0<br>PPMN | Processor 1 Physical Master Number<br>This read-only field defines the physical bus master number for CPU1. |

### 8.4.2.21 Processor 1 Count Register (CP1COUNT)

#### 8.4.2.21.1 Offset

| Register | Offset |
|---|---|
| CP1COUNT | 4Ch |

#### 8.4.2.21.2 Function

The register indicates the total number of processor cores in the chip configuration.

### 8.4.2.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | PCNT | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 8.4.2.21.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>PCNT | Processor Count<br>This read-only field defines the processor count for the chip configuration:<br>PCNT = 01 (Dual Core) |

## 8.4.2.22 Processor 1 Configuration Register 0 (CP1CFG0)

### 8.4.2.22.1 Offset

| Register | Offset |
|---|---|
| CP1CFG0 | 50h |

### 8.4.2.22.2 Function

The CP1CFG0 register provides information on the CPU1 Level 1 caches (if present).

Access: Privileged read-only

**NOTE**

Reset values for the Processor 1 Configuration Register 0:

- CP1CFG0 = 0x03000000

## 8.4.2.22.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ICSZ | | | | | | | | ICWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCSZ | | | | | | | | DCWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 8.4.2.22.4 Fields

| Field | Function |
|---|---|
| 31-24<br>ICSZ | Level 1 Instruction Cache Size<br>This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+ICSZ)}$, where ICSZ is non-zero; a ICSZ = 0 indicates the memory is not present.<br>• if no Instruction Cache, then ICSZ = 0x00<br>• if a 1 Kbyte Instruction Cache, then ICSZ = 0x01<br>• if a 2 Kbyte Instruction Cache, then ICSZ = 0x02<br>• if a 4 Kbyte Instruction Cache, then ICSZ = 0x03<br>• if an 8 Kbyte Instruction Cache, then ICSZ = 0x04<br>• if a 16 Kbyte Instruction Cache, then ICSZ = 0x05<br>• if a 32 Kbyte Instruction Cache, then ICSZ = 0x06<br>• if a 64 Kbyte Instruction Cache, then ICSZ = 0x07<br>• if a 128 Kbyte Instruction Cache, then ICSZ = 0x08 |
| 23-16<br>ICWY | Level 1 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. ICWY=0x00 indicates not present. |
| 15-8<br>DCSZ | Level 1 Data Cache Size<br>This read-only field provides an encoded value of the Data Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+DCSZ)}$ , where DCSZ is non-zero; a DCSZ = 0 indicates the memory is not present.<br>• if noData Cache, then DCSZ = 0x00<br>• if a 1 Kbyte Data Cache, then DCSZ = 0x01<br>• if a 2 Kbyte Data Cache, then DCSZ = 0x02<br>• if a 4 Kbyte Data Cache, then DCSZ = 0x03<br>• if an 8 Kbyte Data Cache, then DCSZ = 0x04<br>• if a 16 Kbyte Data Cache, then DCSZ = 0x05<br>• if a 32 Kbyte Data Cache, then DCSZ = 0x06<br>• if a 64 Kbyte Data Cache, then DCSZ = 0x07<br>• if a 128 Kbyte Data Cache, then DCSZ = 0x08 |
| 7-0<br>DCWY | Level 1 Data Cache Ways<br>This read-only field provides the number of cache ways for the Data Cache. DCWY=0x00 indicates not present. |

## 8.4.2.23 Processor 1 Configuration Register 1 (CP1CFG1)

### 8.4.2.23.1 Offset

| Register | Offset |
|---|---|
| CP1CFG1 | 54h |

### 8.4.2.23.2 Function

The CP1CFG1 register provides information on CPU1 Level 2 cache (if present).

Access: Privileged read-only

**NOTE**

Reset values for the Processor 1 Configuration Register 1:

- CP1CFG1 = 0x00000000

### 8.4.2.23.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L2SZ | | | | | | | | L2WY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 8.4.2.23.4 Fields

| Field | Function |
|---|---|
| 31-24<br>L2SZ | Level 2 Instruction Cache Size<br>This read-only field provides an encoded value of the Instruction Cache size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+L2SZ)}$, where L2SZ is non-zero; a L2SZ = 0 indicates the memory is not present.<br>• if no Level 2 Cache, then L2SZ = 0x00<br>• if a 1 Kbyte Level 2 Cache, then L2SZ = 0x01<br>• if a 2 Kbyte Level 2 Cache, then L2SZ = 0x02<br>• if a 4 Kbyte Level 2 Cache, then L2SZ = 0x03 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • if an 8 Kbyte Level 2 Cache, then L2SZ = 0x04<br>• if a 16 Kbyte Level 2 Cache, then L2SZ = 0x05<br>• if a 32 Kbyte Level 2 Cache, then L2SZ = 0x06<br>• if a 64 Kbyte Level 2 Cache, then L2SZ = 0x07<br>• if a 128 Kbyte Level 2 Cache, then L2SZ = 0x08 |
| 23-16<br>L2WY | Level 2 Instruction Cache Ways<br>This read-only field provides the number of cache ways for the Instruction Cache. L2WY=0x00 indicates not present. |
| 15-0<br>— | Reserved |

## 8.4.2.24 Processor 1 Configuration Register 2 (CP1CFG2)

### 8.4.2.24.1 Offset

| Register | Offset |
|---|---|
| CP1CFG2 | 58h |

### 8.4.2.24.2 Function

The CP1CFG2 register provides information on CPU1 tightly-coupled local memories (if present).

**NOTE**

Reset values for the Processor 1 Configuration Register 2:

- CP1CFG2 = 0x08010001

### 8.4.2.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMLSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMUSZ | | | | | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 8.4.2.24.4 Fields

| Field | Function |
|---|---|
| 31-24<br>TMLSZ | Tightly-coupled Memory Lower Size<br>This field provides an encoded value of the tightly-coupled local memory lower size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMLSZ)}$, where TMLSZ is non-zero; a TMLSZ = 0 indicates the memory is not present.<br>• if no TCML, then TMLSZ = 0x00<br>• if a 1 Kbyte TCML, then TMLSZ = 0x01<br>• if a 2 Kbyte TCML, then TMLSZ = 0x02<br>• if a 4 Kbyte TCML, then TMLSZ = 0x03<br>• if an 8 Kbyte TCML, then TMLSZ = 0x04<br>• if a 16 Kbyte TCML, then TMLSZ = 0x05<br>• if a 32 Kbyte TCML, then TMLSZ = 0x06<br>• if a 64 Kbyte TCML, then TMLSZ = 0x07<br>• if a 128 Kbyte TCML, then TMLSZ = 0x08<br>• if a 256 Kbyte TCML, then TMLSZ = 0x09 |
| 23-16<br>— | Reserved. |
| 15-8<br>TMUSZ | Tightly-coupled Memory Upper Size<br>This field provides an encoded value of the tightly-coupled local memory upper size. The capacity of the memory is expressed as Size [bytes] = $2^{(9+TMUSZ)}$, where TMUSZ is non-zero; a TMUSZ = 0 indicates the memory is not present.<br>• if no TCMU, then TMUSZ = 0x00<br>• if a 1 Kbyte TCMU, then TMUSZ = 0x01<br>• if a 2 Kbyte TCMU, then TMUSZ = 0x02<br>• if a 4 Kbyte TCMU, then TMUSZ = 0x03<br>• if an 8 Kbyte TCMU, then TMUSZ = 0x04<br>• if a 16 Kbyte TCMU, then TMUSZ = 0x05<br>• if a 32 Kbyte TCMU, then TMUSZ = 0x06<br>• if a 64 Kbyte TCMU, then TMUSZ = 0x07<br>• if a 128 Kbyte TCMU, then TMUSZ = 0x08<br>• if a 256 Kbyte TCMU, then TMUSZ = 0x09 |
| 7-0<br>— | Reserved. |

## 8.4.2.25 Processor 1 Configuration Register 3 (CP1CFG3)

### 8.4.2.25.1 Offset

| Register | Offset |
|---|---|
| CP1CFG3 | 5Ch |

### 8.4.2.25.2 Function

The CP1CFG3 register provides information on Processor 1 options.

**NOTE**

Reset values for the Processor 1 Configuration Register 3:

- CP1CFG3 = 0x00000120

### 8.4.2.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | SBP | | 0 | BB | CMP | TZ | MMU | JAZ | SIMD | FPU |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

### 8.4.2.25.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>SBP | System Bus Ports<br>This field defines the number of physical connections to the system bus fabric for this processor. |
| 7<br>— | Reserved |
| 6<br>BB | Bit Banding<br>This field defines if the processor supports "bit banding".<br>0b - Bit Banding is not supported.<br>1b - Bit Banding is supported. |
| 5<br>CMP | Core Memory Protection unit<br>This field indicates if the core memory protection hardware is included in the processor.<br>0b - Core Memory Protection is not included.<br>1b - Core Memory Protection is included. |
| 4<br>TZ | Trust Zone<br>This field indicates if the Trust Zone capabilities are supported in the processor..<br>0b - Trust Zone support is not included.<br>1b - Trust Zone support is included. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3<br>MMU | Memory Management Unit<br>Memory Management Unit. This field indicates if the virtual memory management capabilities are supported in the processor.<br>0b - MMU support is not included.<br>1b - MMU support is included. |
| 2<br>JAZ | Jazelle support<br>This field indicates if Jazelle hardware is supported in the processor.<br>0b - Jazelle support is not included.<br>1b - Jazelle support is included. |
| 1<br>SIMD | SIMD/NEON instruction support<br>This field indicates if the instruction set extensions supporting SIMD and/or NEON capabilities are supported in the processor.<br>0b - SIMD/NEON support is not included.<br>1b - SIMD/NEON support is included. |
| 0<br>FPU | Floating Point Unit<br>This field indicates if hardware support for floating point capabilities are supported in the processor.<br>0b - FPU support is not included.<br>1b - FPU support is included. |

## 8.4.2.26 On-Chip Memory Descriptor Register (OCMDR0)

### 8.4.2.26.1 Offset

| Register | Offset |
|---|---|
| OCMDR0 | 400h |

### 8.4.2.26.2 Function

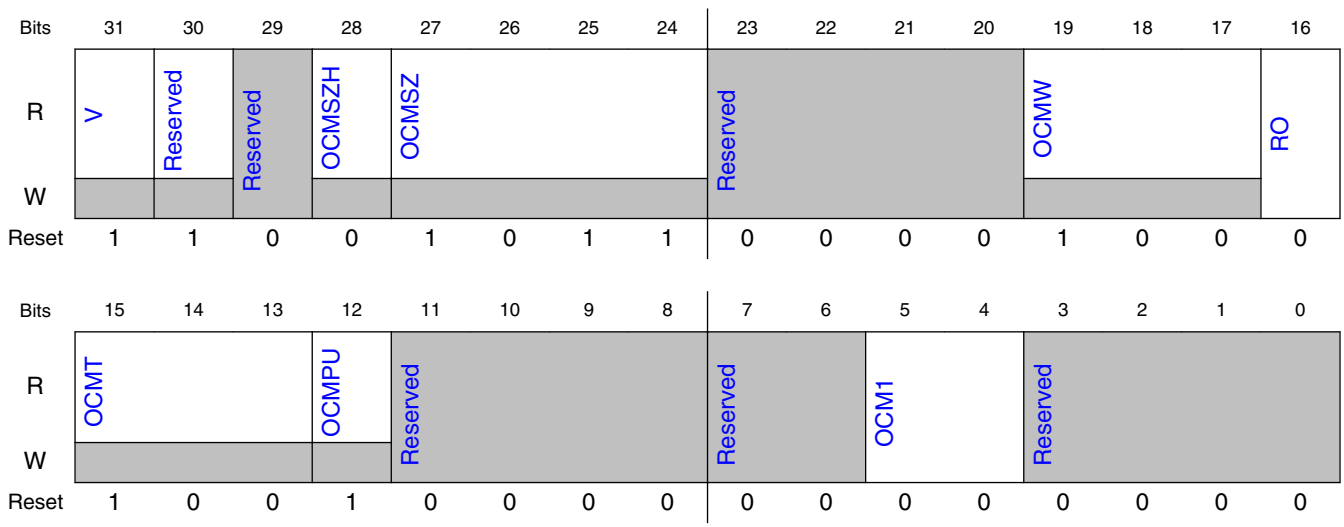
This section of the programming model is an array of 32-bit generic on-chip memory descriptor registers that provide static information about the attached memories, as well as configurable controls (where appropriate).

- Privileged 32-bit reads from a processor core or the debugger return the appropriate processor information.
- Reads from any other bus master return all zeroes.
- Privileged writes from a processor core or the debugger to writeable registers update the appropriate fields.
- Privileged writes from other bus masters are ignored.
- Attempted user mode accesses or any access with a size other than 32 bits are terminated with an error.

The following table describes the OCMDRn reset values and the associated memory type:

| OCMDRn | Reset Value | On-Chip Memory Type |
|---|---|---|
| OCMDR0 | 0xCB089000 | Program Flash |
| OCMDR1 | 0xC7069000 | Program Flash |
| OCMDR2 | 0xC004D000 | EEERAM |
| OCMDR3 | 0xD7047000 | ROM |

## 8.4.2.26.3 Diagram




## 8.4.2.26.4 Fields

| Field | Function |
|---|---|
| 31<br>V | V<br>OCMEM Valid bit. This read-only field defines the validity (presence) of the on-chip memory<br>0b - OCMEMn is not present.<br>1b - OCMEMn is present. |
| 30<br>— | Reserved<br>This Reserved field always has the value of 1. |
| 29<br>— | Reserved |
| 28<br>OCMSZH | OCMSZH<br>OCMEM Size “Hole”. For on-chip memories that are not fully populated, that is, include a memory “hole” in the upper 25% of the address range, this bit is used.<br>0b - OCMEMn is a power-of-2 capacity. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - OCMEMn is not a power-of-2, with a capacity is 0.75 * OCMSZ. |
| 27-24<br>OCMSZ | OCMSZ<br>OCMEM Size. This read-only field provides an encoded value of the on-chip memory size.<br>0000b - no OCMEMn<br>0001b - 1KB OCMEMn<br>0010b - 2KB OCMEMn<br>0011b - 4KB OCMEMn<br>0100b - 8KB OCMEMn<br>0101b - 16KB OCMEMn<br>0110b - 32KB OCMEMn<br>0111b - 64KB OCMEMn<br>1000b - 128KB OCMEMn<br>1001b - 256KB OCMEMn<br>1010b - 512KB OCMEMn<br>1011b - 1MB OCMEMn<br>1100b - 2MB OCMEMn<br>1101b - 4MB OCMEMn<br>1110b - 8MB OCMEMn<br>1111b - 16MB OCMEMn |
| 23-20<br>— | Reserved |
| 19-17<br>OCMW | OCMW<br>OCMEM datapath Width. This read-only field defines the width of the on-chip memory:<br>000-001b - Reserved<br>010b - OCMEMn 32-bits wide<br>011b - OCMEMn 64-bits wide<br>100b - OCMEMn 128-bits wide<br>101b - OCMEMn 256-bits wide<br>110-111b - Reserved |
| 16<br>RO | RO<br>Read-Only. This register bit provides a mechanism to “lock” the configuration state defined by OCMDRn[11:0]. Once asserted, attempted writes to the OCMDRn[11:0] register are ignored until the next reset clears the flag.<br>0b - Writes to the OCMDRn[11:0] are allowed<br>1b - Writes to the OCMDRn[11:0] are ignored |
| 15-13<br>OCMT | OCMT<br>OCMEM Type. This field defines the type of the on-chip memory:<br>000b - Reserved<br>001b - Reserved<br>010b - Reserved<br>011b - OCMEMn is a ROM.<br>100b - OCMEMn is a Program Flash.<br>101b - Reserved<br>110b - OCMEMn is an EEE.<br>111b - Reserved |
| 12<br>OCMPU | OCMPU<br>OCMEM Memory Protection Unit. This field is reserved for this device. |
| 11-8<br>— | Reserved |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>7-6<br>—</td><td>Reserved</td></tr>
<tr><td>5-4<br>OCM1</td><td>OCMEM Control Field 1<br>The flash controller needs to be idle when writing to an OCMDRn register associated with Flash memory. Changing controller configuration while active can cause undesired results.<br>OCMDRn[4] or OCMDRn[5] bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches or data references, see section FMC speculative reads. Value 0 means enable and value 1 means disable.
<table>
<tr><th>Flash Speculate<br>(OCMDRn bit 5)</th><th>Data Prefetch<br>(OCMDRn bit 4)</th><th>Result</th></tr>
<tr><td>Disabled</td><td>Disabled</td><td rowspan="2">All speculation disabled and speculation buffer is cleared</td></tr>
<tr><td>Disabled</td><td>Enabled</td></tr>
<tr><td>Enabled</td><td>Disabled</td><td>Speculation for Instruction enabled and Speculation for Data disabled</td></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Speculation for both Instruction and Data enabled</td></tr>
</table></td></tr>
<tr><td>3-0<br>—</td><td>Reserved</td></tr>
</table>

## 8.4.2.27 On-Chip Memory Descriptor Register (OCMDR1)

### 8.4.2.27.1 Offset

| Register | Offset |
|---|---|
| OCMDR1 | 404h |

### 8.4.2.27.2 Function

This section of the programming model is an array of 32-bit generic on-chip memory descriptor registers that provide static information about the attached memories, as well as configurable controls (where appropriate).

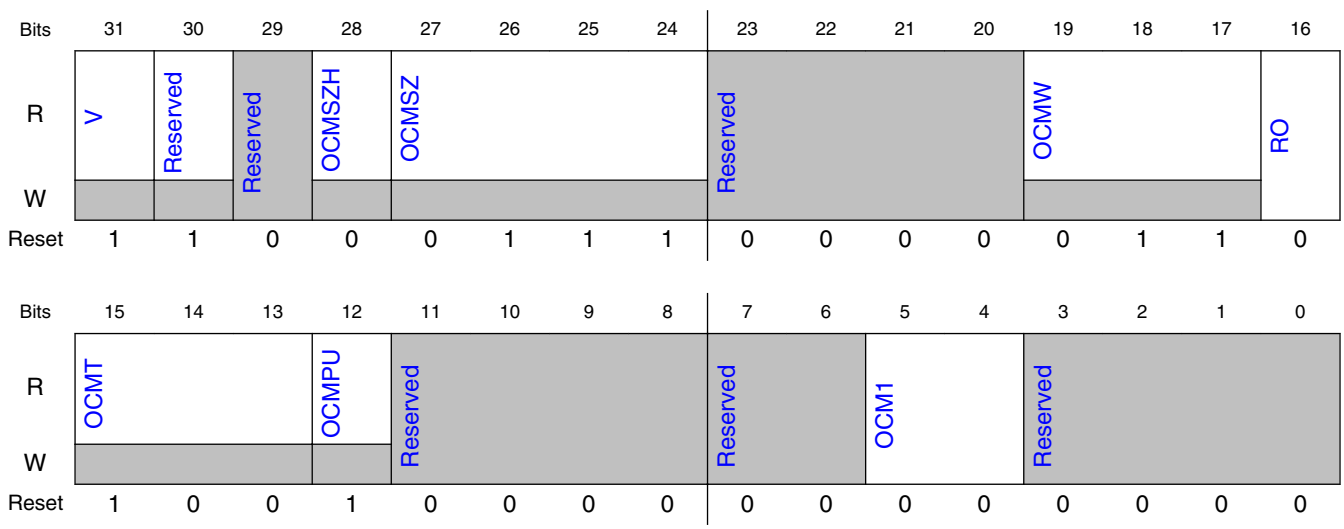
- Privileged 32-bit reads from a processor core or the debugger return the appropriate processor information.
- Reads from any other bus master return all zeroes.
- Privileged writes from a processor core or the debugger to writeable registers update the appropriate fields.

- Privileged writes from other bus masters are ignored.
- Attempted user mode accesses or any access with a size other than 32 bits are terminated with an error.

The following table describes the OCMDRn reset values and the associated memory type:

| OCMDRn | Reset Value | On-Chip Memory Type |
|---|---|---|
| OCMDR0 | 0xCB089000 | Program Flash |
| OCMDR1 | 0xC7069000 | Program Flash |
| OCMDR2 | 0xC004D000 | EEERAM |
| OCMDR3 | 0xD7047000 | ROM |

## 8.4.2.27.3 Diagram




## 8.4.2.27.4 Fields

| Field | Function |
|---|---|
| 31<br>V | V<br>OCMEM Valid bit. This read-only field defines the validity (presence) of the on-chip memory<br>0b - OCMEMn is not present.<br>1b - OCMEMn is present. |
| 30<br>— | Reserved<br>This Reserved field always has the value of 1. |
| 29<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 28<br>OCMSZH | OCMSZH<br>OCMEM Size “Hole”. For on-chip memories that are not fully populated, that is, include a memory “hole” in the upper 25% of the address range, this bit is used.<br>0b - OCMEMn is a power-of-2 capacity.<br>1b - OCMEMn is not a power-of-2, with a capacity is 0.75 * OCMSZ. |
| 27-24<br>OCMSZ | OCMSZ<br>OCMEM Size. This read-only field provides an encoded value of the on-chip memory size.<br>0000b - no OCMEMn<br>0001b - 1KB OCMEMn<br>0010b - 2KB OCMEMn<br>0011b - 4KB OCMEMn<br>0100b - 8KB OCMEMn<br>0101b - 16KB OCMEMn<br>0110b - 32KB OCMEMn<br>0111b - 64KB OCMEMn<br>1000b - 128KB OCMEMn<br>1001b - 256KB OCMEMn<br>1010b - 512KB OCMEMn<br>1011b - 1MB OCMEMn<br>1100b - 2MB OCMEMn<br>1101b - 4MB OCMEMn<br>1110b - 8MB OCMEMn<br>1111b - 16MB OCMEMn |
| 23-20<br>— | Reserved |
| 19-17<br>OCMW | OCMW<br>OCMEM datapath Width. This read-only field defines the width of the on-chip memory:<br>000-001b - Reserved<br>010b - OCMEMn 32-bits wide<br>011b - OCMEMn 64-bits wide<br>100b - OCMEMn 128-bits wide<br>101b - OCMEMn 256-bits wide<br>110-111b - Reserved |
| 16<br>RO | RO<br>Read-Only. This register bit provides a mechanism to “lock” the configuration state defined by OCMDRn[11:0]. Once asserted, attempted writes to the OCMDRn[11:0] register are ignored until the next reset clears the flag.<br>0b - Writes to the OCMDRn[11:0] are allowed<br>1b - Writes to the OCMDRn[11:0] are ignored |
| 15-13<br>OCMT | OCMT<br>OCMEM Type. This field defines the type of the on-chip memory:<br>000b - Reserved<br>001b - Reserved<br>010b - Reserved<br>011b - OCMEMn is a ROM.<br>100b - OCMEMn is a Program Flash.<br>101b - Reserved<br>110b - OCMEMn is an EEE.<br>111b - Reserved |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>12<br>OCMPU</td><td>OCMPU<br>OCMEM Memory Protection Unit. This field is reserved for this device.</td></tr>
<tr><td>11-8<br>—</td><td>Reserved</td></tr>
<tr><td>7-6<br>—</td><td>Reserved</td></tr>
<tr><td>5-4<br>OCM1</td><td>OCMEM Control Field 1<br>The flash controller needs to be idle when writing to an OCMDRn register associated with Flash memory. Changing controller configuration while active can cause undesired results.<br>OCMDRn[4] or OCMDRn[5] bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches or data references, see section FMC speculative reads. Value 0 means enable and value 1 means disable.
<table>
<tr><th>Flash Speculate<br>(OCMDRn bit 5)</th><th>Data Prefetch<br>(OCMDRn bit 4)</th><th>Result</th></tr>
<tr><td>Disabled</td><td>Disabled</td><td rowspan="2">All speculation disabled and speculation buffer is cleared</td></tr>
<tr><td>Disabled</td><td>Enabled</td></tr>
<tr><td>Enabled</td><td>Disabled</td><td>Speculation for Instruction enabled and Speculation for Data disabled</td></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Speculation for both Instruction and Data enabled</td></tr>
</table></td></tr>
<tr><td>3-0<br>—</td><td>Reserved</td></tr>
</table>

## 8.4.2.28 On-Chip Memory Descriptor Register (OCMDR2)

### 8.4.2.28.1 Offset

| Register | Offset |
|---|---|
| OCMDR2 | 408h |

### 8.4.2.28.2 Function

This section of the programming model is an array of 32-bit generic on-chip memory descriptor registers that provide static information about the attached memories, as well as configurable controls (where appropriate).

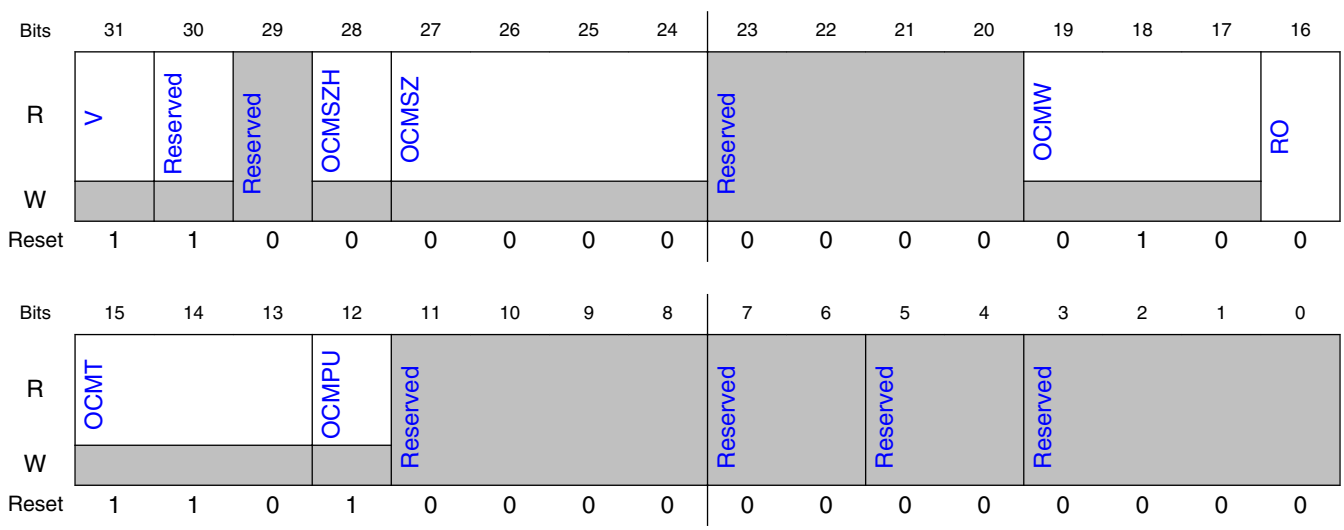
- Privileged 32-bit reads from a processor core or the debugger return the appropriate processor information.

- Reads from any other bus master return all zeroes.
- Privileged writes from a processor core or the debugger to writeable registers update the appropriate fields.
- Privileged writes from other bus masters are ignored.
- Attempted user mode accesses or any access with a size other than 32 bits are terminated with an error.

The following table describes the OCMDRn reset values and the associated memory type:

| OCMDRn | Reset Value | On-Chip Memory Type |
|---|---|---|
| OCMDR0 | 0xCB089000 | Program Flash |
| OCMDR1 | 0xC7069000 | Program Flash |
| OCMDR2 | 0xC004D000 | EEERAM |
| OCMDR3 | 0xD7047000 | ROM |

### 8.4.2.28.3 Diagram




### 8.4.2.28.4 Fields

| Field | Function |
|---|---|
| 31<br>V | V<br>OCMEM Valid bit. This read-only field defines the validity (presence) of the on-chip memory<br>0b - OCMEMn is not present.<br>1b - OCMEMn is present. |
| 30 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | This Reserved field always has the value of 1. |
| 29<br>— | Reserved |
| 28<br>OCMSZH | OCMSZH<br>OCMEM Size “Hole”. For on-chip memories that are not fully populated, that is, include a memory “hole” in the upper 25% of the address range, this bit is used.<br>0b - OCMEMn is a power-of-2 capacity.<br>1b - OCMEMn is not a power-of-2, with a capacity is 0.75 * OCMSZ. |
| 27-24<br>OCMSZ | OCMSZ<br>OCMEM Size. This read-only field provides an encoded value of the on-chip memory size.<br>0000b - no OCMEMn<br>0001b - 1KB OCMEMn<br>0010b - 2KB OCMEMn<br>0011b - 4KB OCMEMn<br>0100b - 8KB OCMEMn<br>0101b - 16KB OCMEMn<br>0110b - 32KB OCMEMn<br>0111b - 64KB OCMEMn<br>1000b - 128KB OCMEMn<br>1001b - 256KB OCMEMn<br>1010b - 512KB OCMEMn<br>1011b - 1MB OCMEMn<br>1100b - 2MB OCMEMn<br>1101b - 4MB OCMEMn<br>1110b - 8MB OCMEMn<br>1111b - 16MB OCMEMn |
| 23-20<br>— | Reserved |
| 19-17<br>OCMW | OCMW<br>OCMEM datapath Width. This read-only field defines the width of the on-chip memory:<br>000-001b - Reserved<br>010b - OCMEMn 32-bits wide<br>011b - OCMEMn 64-bits wide<br>100b - OCMEMn 128-bits wide<br>101b - OCMEMn 256-bits wide<br>110-111b - Reserved |
| 16<br>RO | RO<br>Read-Only. This register bit provides a mechanism to “lock” the configuration state defined by OCMDRn[11:0]. Once asserted, attempted writes to the OCMDRn[11:0] register are ignored until the next reset clears the flag.<br>0b - Writes to the OCMDRn[11:0] are allowed<br>1b - Writes to the OCMDRn[11:0] are ignored |
| 15-13<br>OCMT | OCMT<br>OCMEM Type. This field defines the type of the on-chip memory:<br>000b - Reserved<br>001b - Reserved<br>010b - Reserved<br>011b - OCMEMn is a ROM. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - OCMEMn is a Program Flash.<br>101b - Reserved<br>110b - OCMEMn is an EEE.<br>111b - Reserved |
| 12<br>OCMPU | OCMPU<br>OCMEM Memory Protection Unit. This field is reserved for this device. |
| 11-8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-4<br>— | Reserved |
| 3-0<br>— | Reserved |

## 8.4.2.29 On-Chip Memory Descriptor Register (OCMDR3)

### 8.4.2.29.1 Offset

| Register | Offset |
|---|---|
| OCMDR3 | 40Ch |

### 8.4.2.29.2 Function

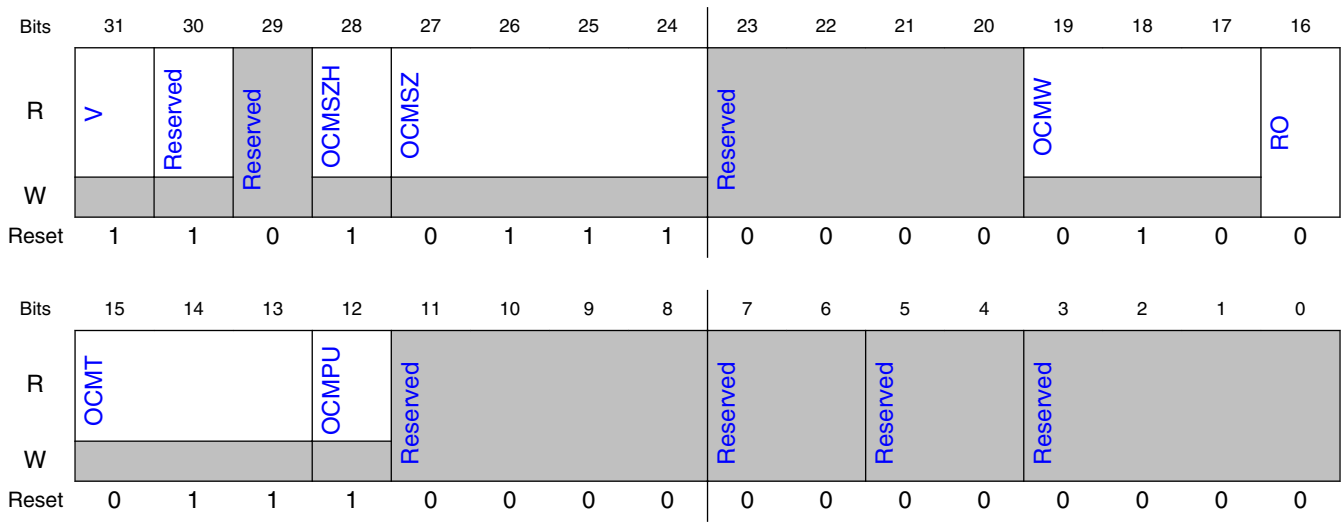
This section of the programming model is an array of 32-bit generic on-chip memory descriptor registers that provide static information about the attached memories, as well as configurable controls (where appropriate).

- Privileged 32-bit reads from a processor core or the debugger return the appropriate processor information.
- Reads from any other bus master return all zeroes.
- Privileged writes from a processor core or the debugger to writeable registers update the appropriate fields.
- Privileged writes from other bus masters are ignored.
- Attempted user mode accesses or any access with a size other than 32 bits are terminated with an error.

The following table describes the OCMDRn reset values and the associated memory type:

| OCMDRn | Reset Value | On-Chip Memory Type |
|---|---|---|
| OCMDR0 | 0xCB089000 | Program Flash |
| OCMDR1 | 0xC7069000 | Program Flash |
| OCMDR2 | 0xC004D000 | EEERAM |
| OCMDR3 | 0xD7047000 | ROM |

## 8.4.2.29.3 Diagram




## 8.4.2.29.4 Fields

| Field | Function |
|---|---|
| 31<br>V | V<br>OCMEM Valid bit. This read-only field defines the validity (presence) of the on-chip memory<br>0b - OCMEMn is not present.<br>1b - OCMEMn is present. |
| 30<br>— | Reserved<br>This Reserved field always has the value of 1. |
| 29<br>— | Reserved |
| 28<br>OCMSZH | OCMSZH<br>OCMEM Size “Hole”. For on-chip memories that are not fully populated, that is, include a memory “hole” in the upper 25% of the address range, this bit is used.<br>0b - OCMEMn is a power-of-2 capacity.<br>1b - OCMEMn is not a power-of-2, with a capacity is 0.75 * OCMSZ. |
| 27-24<br>OCMSZ | OCMSZ<br>OCMEM Size. This read-only field provides an encoded value of the on-chip memory size. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0000b - no OCMEMn<br>0001b - 1KB OCMEMn<br>0010b - 2KB OCMEMn<br>0011b - 4KB OCMEMn<br>0100b - 8KB OCMEMn<br>0101b - 16KB OCMEMn<br>0110b - 32KB OCMEMn<br>0111b - 64KB OCMEMn<br>1000b - 128KB OCMEMn<br>1001b - 256KB OCMEMn<br>1010b - 512KB OCMEMn<br>1011b - 1MB OCMEMn<br>1100b - 2MB OCMEMn<br>1101b - 4MB OCMEMn<br>1110b - 8MB OCMEMn<br>1111b - 16MB OCMEMn |
| 23-20<br>— | Reserved |
| 19-17<br>OCMW | OCMW<br>OCMEM datapath Width. This read-only field defines the width of the on-chip memory:<br>000-001b - Reserved<br>010b - OCMEMn 32-bits wide<br>011b - OCMEMn 64-bits wide<br>100b - OCMEMn 128-bits wide<br>101b - OCMEMn 256-bits wide<br>110-111b - Reserved |
| 16<br>RO | RO<br>Read-Only. This register bit provides a mechanism to “lock” the configuration state defined by OCMDRn[11:0]. Once asserted, attempted writes to the OCMDRn[11:0] register are ignored until the next reset clears the flag.<br>0b - Writes to the OCMDRn[11:0] are allowed<br>1b - Writes to the OCMDRn[11:0] are ignored |
| 15-13<br>OCMT | OCMT<br>OCMEM Type. This field defines the type of the on-chip memory:<br>000b - Reserved<br>001b - Reserved<br>010b - Reserved<br>011b - OCMEMn is a ROM.<br>100b - OCMEMn is a Program Flash.<br>101b - Reserved<br>110b - OCMEMn is an EEE.<br>111b - Reserved |
| 12<br>OCMPU | OCMPU<br>OCMEM Memory Protection Unit. This field is reserved for this device. |
| 11-8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-4 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 3-0<br>— | Reserved |

# Chapter 9
# Extended Resource Domain Controller (XRDC)

## 9.1 Chip-specific XRDC information

### 9.1.1 XRDC configuration

The XRDC implementation is distributed across multiple submodules instantiated throughout the device.

The following table shows the XRDC configurations for this device.

**Table 9-1. XRDC Configuration**

| Parameter | Number |
|---|---|
| DOMAIN_NUM | 2 |
| MDAC_NUM | 12 |
| MRC_NUM | 1 |
| PAC_NUM | 2 |

The following table shows the MDC and PID registers mapping for this device.

**Table 9-2. MDC and PID registers mapping**

| Master | MDAC | PID |
|---|---|---|
| CM4F C-BUS | MDAC0 | PID0 |
| CM4F S-BUS | MDAC1 | PID1 |
| CM0+ | MDAC32 | PID2 |

## 9.1.2 XRDC register configurations

### Table 9-3. MDAC configurations

| MDAC # | Master | xRDC MDAC register[1] | MDAC_REG_NUM | DEFAULT_DID | MDAC_CPU |
|---|---|---|---|---|---|
| MDAC0 | CM4F C-BUS[2] | MDA_W*_DFMT0 | 2 | 0 | 1 |
| MDAC1 | CM4F S-BUS[3] | MDA_W*_DFMT0 | 2 | 0 | 1 |
| MDAC2 | DMA0 | MDA_W*_DFMT1 | 1 | 0 | 0 |
| MDAC3 | uSDHC | MDA_W*_DFMT1 | 1 | 2 | 0 |
| MDAC4 | USB | MDA_W*_DFMT1 | 1 | 2 | 0 |
| MDAC16 | RI5CY C-BUS | MDA_W*_DFMT0 | 2 | 0 | 1 |
| MDAC17 | RI5CY S-BUS | MDA_W*_DFMT0 | 2 | 0 | 1 |
| MDAC32 | CM0+ | MDA_W*_DFMT0 | 2 | 1 | 1 |
| MDAC33 | DMA1 | MDA_W*_DFMT1 | 1 | 1 | 0 |
| MDAC34 | CAU3 | MDA_W*_DFMT1 | 1 | 1 | 0 |
| MDAC35 | ZERORISCY C-BUS | MDA_W*_DFMT0 | 2 | 1 | 1 |
| MDAC36 | ZERORISCY D-BUS | MDA_W*_DFMT0 | 2 | 1 | 1 |

1. The MDA_W*_DFMTn here refers to MDAC_W0_0_DFMT0 - MDA_W1_0_DFMT0.
2. Access address range: 0x0000_0000-0x1FFF_FFFF.
3. Access address range: 0x2000_0000-0xFFFF_FFFF.

### Table 9-4. MRC configurations

| MRC # | xRDC MRC register[1] | Slave Memory | REGION_NUM |
|---|---|---|---|
| MRC0 | MRGD_W*_0_0 | • CM4F flash<br>• CM4F ITCM SRAM<br>• Boot ROM<br>• CM4F DTCM SRAM<br>• FlexRAM<br>• FlexBus | 8 |
| MRC0 | MRGD_W*_0_1 | | |
| MRC0 | MRGD_W*_0_2 | | |
| MRC0 | MRGD_W*_0_3 | | |
| MRC0 | MRGD_W*_0_4 | | |
| MRC0 | MRGD_W*_0_5 | | |
| MRC0 | MRGD_W*_0_6 | | |
| MRC0 | MRGD_W*_0_7 | | |
| MRC1 | MRGD_W*_1_0 | • CM0+ flash<br>• CM0+ SRAM<br>• CM0+ PPB | 8 |
| MRC1 | MRGD_W*_1_1 | | |
| MRC1 | MRGD_W*_1_2 | | |
| MRC1 | MRGD_W*_1_3 | | |
| MRC1 | MRGD_W*_1_4 | | |
| MRC1 | MRGD_W*_1_5 | | |
| MRC1 | MRGD_W*_1_6 | | |
| MRC1 | MRGD_W*_1_7 | | |

1. The MRGD_W*_m_n here refers to MRGD_W0_m_n,MRGD_W1_m_n, MRGD_W2_m_n, MRGD_W3_m_n, MRGD_W4_m_n.

## Table 9-5. PAC configurations

| PAC # | xRDC PAC register | Peripheral | Slot | IP Bus |
|---|---|---|---|---|
| PAC0 | PDAC_W*_0_1 | Miscellaneous System Control Module (MSCM) | 1 | AIPS0 |
| PAC0 | PDAC_W*_0_4 | MAX CORE | 4 | AIPS0 |
| PAC0 | PDAC_W*_0_8 | Direct Memory Access 0 (DMA0) controller | 8 | AIPS0 |
| PAC0 | PDAC_W*_0_9 | Direct Memory Access 0 (DMA0) controller transfer control descriptors | 9 | AIPS0 |
| PAC0 | PDAC_W*_0_12 | External Bus Interface (FlexBus) | 12 | AIPS0 |
| PAC0 | PDAC_W*_0_20 | Resource Domain Controller (RDC) MGR | 20 | AIPS0 |
| PAC0 | PDAC_W*_0_21 | Resource Domain Controller (RDC) MDAC | 21 | AIPS0 |
| PAC0 | PDAC_W*_0_22 | Resource Domain Controller (RDC) PAC | 22 | AIPS0 |
| PAC0 | PDAC_W*_0_23 | Resource Domain Controller (RDC) MRC | 23 | AIPS0 |
| PAC0 | PDAC_W*_0_27 | Semaphore Unit (SEMA42) | 27 | AIPS0 |
| PAC0 | PDAC_W*_0_32 | System Mode Controller 0 (SMC0) | 32 | AIPS0 |
| PAC0 | PDAC_W*_0_33 | Direct memory Access Mutliplexer 0 (DMAMUX0) | 33 | AIPS0 |
| PAC0 | PDAC_W*_0_34 | External Watchdog Monitor (EWM) | 34 | AIPS0 |
| PAC0 | PDAC_W*_0_35 | Flash Memory Module (FTFE) | 35 | AIPS0 |
| PAC0 | PDAC_W*_0_36 | Low Leakage Wake-up Unit 0 (LLWU0) | 36 | AIPS0 |
| PAC0 | PDAC_W*_0_37 | Messaging Unit - Side A (MU-A) | 37 | AIPS0 |
| PAC0 | PDAC_W*_0_38 | System Integration Module (SIM) | 38 | AIPS0 |
| PAC0 | PDAC_W*_0_39 | USB Voltage Regulator (USBVREG) | 39 | AIPS0 |

*Table continues on the next page...*

## Table 9-5. PAC configurations (continued)

| PAC # | xRDC PAC register | Peripheral | Slot | IP Bus |
|---|---|---|---|---|
| PAC0 | PDAC_W*_0_40 | System Power Manager (SPM) | 40 | AIPS0 |
| PAC0 | PDAC_W*_0_41 | Tirgger Multiplexer 0 (TRGMUX0) | 41 | AIPS0 |
| PAC0 | PDAC_W*_0_42 | Watchdog 0 (WDOG0) | 42 | AIPS0 |
| PAC0 | PDAC_W*_0_43 | Peripheral Clock Controller 0 (PCC0) | 43 | AIPS0 |
| PAC0 | PDAC_W*_0_44 | System Clock Generator (SCG) | 44 | AIPS0 |
| PAC0 | PDAC_W*_0_45 | System Register File | 45 | AIPS0 |
| PAC0 | PDAC_W*_0_46 | VBAT Register File | 46 | AIPS0 |
| PAC0 | PDAC_W*_0_47 | Cyclic Redundancy Check 0 (CRC0) | 47 | AIPS0 |
| PAC0 | PDAC_W*_0_48 | Low-Power Periodic Interrupt Timer 0 (LPIT0) | 48 | AIPS0 |
| PAC0 | PDAC_W*_0_49 | Real Time Clock (RTC) | 49 | AIPS0 |
| PAC0 | PDAC_W*_0_50 | Low-Power Timer 0 (LPTMR0) | 50 | AIPS0 |
| PAC0 | PDAC_W*_0_51 | Low-Power Timer 1 (LPTMR1) | 51 | AIPS0 |
| PAC0 | PDAC_W*_0_52 | Time Stamp Timer A (TSTMRA) | 52 | AIPS0 |
| PAC0 | PDAC_W*_0_53 | Timer/PWM 0 (TPM0) - 6 channel | 53 | AIPS0 |
| PAC0 | PDAC_W*_0_54 | Timer/PWM Module 1 (TPM1) - 2 channel | 54 | AIPS0 |
| PAC0 | PDAC_W*_0_55 | Timer/PWM Module 2 (TPM2) - 6 channel | 55 | AIPS0 |
| PAC0 | PDAC_W*_0_56 | Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) | 56 | AIPS0 |
| PAC0 | PDAC_W*_0_57 | Flexible Input / Output 0 (FlexIO0) | 57 | AIPS0 |
| PAC0 | PDAC_W*_0_58 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) | 58 | AIPS0 |
| PAC0 | PDAC_W*_0_59 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) | 59 | AIPS0 |
| PAC0 | PDAC_W*_0_60 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) | 60 | AIPS0 |

*Table continues on the next page...*

## Table 9-5. PAC configurations (continued)

| PAC # | xRDC PAC register | Peripheral | Slot | IP Bus |
|---|---|---|---|---|
| PAC0 | PDAC_W*_0_61 | Inter-IC Sound 0 (I2S0) | 61 | AIPS0 |
| PAC0 | PDAC_W*_0_62 | Ultra Secured Digital Host Controller 0 (uSDHC) | 62 | AIPS0 |
| PAC0 | PDAC_W*_0_63 | Low-Power Serial Peripheral Interface 0 (LPSPI0) | 63 | AIPS0 |
| PAC0 | PDAC_W*_0_64 | Low-Power Serial Peripheral Interface 1 (LPSPI1) | 64 | AIPS0 |
| PAC0 | PDAC_W*_0_65 | Low-Power Serial Peripheral Interface 2 (LPSPI2) | 65 | AIPS0 |
| PAC0 | PDAC_W*_0_66 | Low-Power Universal Asynchronous Receiver/Transmitter 0 (LPUART0) | 66 | AIPS0 |
| PAC0 | PDAC_W*_0_67 | Low-Power Universal Asynchronous Receiver/Transmitter 1 (LPUART1) | 67 | AIPS0 |
| PAC0 | PDAC_W*_0_68 | Low-Power Universal Asynchronous Receiver/Transmitter 2 (LPUART2) | 68 | AIPS0 |
| PAC0 | PDAC_W*_0_69 | Universal Serial Bus 0 (USB0) - Full Speed, Device Only | 69 | AIPS0 |
| PAC0 | PDAC_W*_0_70 | PORTA | 70 | AIPS0 |
| PAC0 | PDAC_W*_0_71 | PORTB | 71 | AIPS0 |
| PAC0 | PDAC_W*_0_72 | PORTC | 72 | AIPS0 |
| PAC0 | PDAC_W*_0_73 | PORTD | 73 | AIPS0 |
| PAC0 | PDAC_W*_0_74 | Low Power Analog-to-Digital Converter 0 (LPADC0) | 74 | AIPS0 |
| PAC0 | PDAC_W*_0_75 | Low-Power Comparator 0 (LPCMP0) | 75 | AIPS0 |
| PAC0 | PDAC_W*_0_76 | Low Power Digital-to-Analog Converter 0 (LPDAC0) | 76 | AIPS0 |
| PAC0 | PDAC_W*_0_77 | Voltage Reference (VREF) | 77 | AIPS0 |
| PAC0 | PDAC_W*_0_79 | RI5CY INTMUX | 79 | AIPS0 |
| PAC1 | PDAC_W*_1_8 | Direct Memory Access 1 (DMA1) controller | 8 | AIPS1 |

*Table continues on the next page...*

## Table 9-5. PAC configurations (continued)

| PAC # | xRDC PAC register | Peripheral | Slot | IP Bus |
|---|---|---|---|---|
| PAC1 | PDAC_W*_1_9 | Direct Memory Access 1 (DMA1) controller transfer control descriptors | 9 | AIPS1 |
| PAC1 | PDAC_W*_1_15 | IO Port Alias | 15 | AIPS1 |
| PAC1 | PDAC_W*_1_27 | Semaphore Unit (SEMA42) | 27 | AIPS1 |
| PAC1 | PDAC_W*_1_31 | ZERORISCY Event Unit | 31 | AIPS1 |
| PAC1 | PDAC_W*_1_32 | System Mode Controller 1 (SMC1) | 32 | AIPS1 |
| PAC1 | PDAC_W*_1_33 | Direct Memory Access Mutiplexer 1 (DMAMUX1) | 33 | AIPS1 |
| PAC1 | PDAC_W*_1_34 | Interrupt Multiplexer 0 (INTMUX0) | 34 | AIPS1 |
| PAC1 | PDAC_W*_1_35 | Low Leakage Wake-up Unit 1 (LLWU1) | 35 | AIPS1 |
| PAC1 | PDAC_W*_1_36 | Messaging Unit - Side B (MU-B) | 36 | AIPS1 |
| PAC1 | PDAC_W*_1_37 | Trigger Multiplexer 1 (TRGMUX1) | 37 | AIPS1 |
| PAC1 | PDAC_W*_1_38 | Watchdog 1 (WDOG1) | 38 | AIPS1 |
| PAC1 | PDAC_W*_1_39 | Peripheral Clock Controller 1 (PCC1) | 39 | AIPS1 |
| PAC1 | PDAC_W*_1_40 | Cryptographic Acceleration Unit (CAU3) | 40 | AIPS1 |
| PAC1 | PDAC_W*_1_41 | True Random Number Generator (TRNG) | 41 | AIPS1 |
| PAC1 | PDAC_W*_1_42 | Low-Power Periodic Interrupt Timer 1 (LPIT1) | 42 | AIPS1 |
| PAC1 | PDAC_W*_1_43 | Low-Power Timer 2 (LPTMR2) | 43 | AIPS1 |
| PAC1 | PDAC_W*_1_44 | Time Stamp Timer B (TSTMRB) | 44 | AIPS1 |
| PAC1 | PDAC_W*_1_45 | Timer/PWM Module 3 (TPM3) - 2 channel | 45 | AIPS1 |
| PAC1 | PDAC_W*_1_46 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) | 46 | AIPS1 |
| PAC1 | PDAC_W*_1_47 | 2.4GHz Radio (RF2.4G) - RSIM | 47 | AIPS1 |

*Table continues on the next page...*

**Table 9-5. PAC configurations (continued)**

| PAC # | xRDC PAC register | Peripheral | Slot | IP Bus |
|---|---|---|---|---|
| PAC1 | PDAC_W*_1_48 | 2.4GHz Radio (RF2.4G) - XCVR | 48 | AIPS1 |
| PAC1 | PDAC_W*_1_50 | 2.4GHz Radio (RF2.4G) - Bluetooth Link layer | 50 | AIPS1 |
| PAC1 | PDAC_W*_1_51 | 2.4GHz Radio (RF2.4G) - Generic Link layer | 51 | AIPS1 |
| PAC1 | PDAC_W*_1_52 | 2.4GHz Radio (RF2.4G) - IEEE 802.15.4 Link Layer | 52 | AIPS1 |
| PAC1 | PDAC_W*_1_53 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | 53 | AIPS1 |
| PAC1 | PDAC_W*_1_54 | Low-Power Universal Asynchronous Receiver/Transmitter 3 (LPUART3) | 54 | AIPS1 |
| PAC1 | PDAC_W*_1_55 | PORTE | 55 | AIPS1 |
| PAC1 | PDAC_W*_1_56 | Low-Power Comparator 1 (LPCMP1) | 56 | AIPS1 |
| PAC2(MSC) | PDAC_W*_2_16 | USB SRAM (slot 16) | 16 | - |
| PAC2(MSC) | PDAC_W*_2_32 | Rapid GPIO (slot 32) | 32 | - |

## 9.2 Introduction

The Extended Resource Domain Controller (XRDC) provides an integrated, scalable architectural framework for access control, system memory protection and peripheral isolation. It allows software to assign chip resources including processor cores, non-core bus masters, memory regions and slave peripherals to processing *domains* to support enforcement of robust operational environments. First, each bus mastering resource is assigned to a domain identifier (domainID, DID). For processors, there are additional fields that can optionally be used to assign it to multiple domains. Next, the access control policies for the individual domains are programmed into any number of slave memory region descriptors and slave peripheral domain access control registers. Finally, all accesses throughout the device are monitored concurrently to determine the validity of each and every access. If a reference from a given domain has sufficient access rights, it is allowed to continue, else the access is aborted and error information captured.

The access control scheme defined by the XRDC supports a 4-level model, combining the traditional privileged (also known as supervisor) and user modes with an additional signal defining the secure, nonsecure attributes of each memory reference. The result is a 4-level hierarchical access control mechanism, where:

PrivSecure > PrivNonsecure > UserSecure > UserNonsecure

with different access control policies based on read and write references. Combined with the user/privileged and secure/nonsecure attributes, a domainID is associated with every system bus transaction and forms the hardware basis for implementation of the XRDC's access control mechanisms.

Access to shared memory regions and slave peripherals can be dynamically controlled with the optional inclusion of a hardware semaphore. If a hardware semaphore is enabled for a given address space or peripheral, then writes to the targeted address space are only allowed if the requesting domain owns the semaphore. This capability allows the access control policy for a given resource to be dynamically revised based on hardware semaphore ownership.

### 9.2.1 Features

The key features of the XRDC include:

- Assignment of chip resources to processing "domains"
  - Processor cores, non-core bus masters, slave memories and slave peripherals
  - Each processing domain is assigned a unique domain identifier (domainID, DID)
  - DomainID is a new attribute associated with every system bus transaction
  - Used in conjunction with user/privileged, secure/nonsecure attributes
- Access rights to slave targets defined in region descriptor registers for memories and access control registers for peripherals
- Supports sharing of memory and peripherals with optional inclusion of hardware semaphores to dynamically determine access rights
- Built upon a 4-level hierarchical access control model
  - PrivSecure > PrivNonsecure > UserSecure > UserNonsecure

    - Encoded into a 3-bit per-domain access control policy (ACP) used throughout the XRDC
    - Certain processors do not support the PrivNonsecure state. For these cores, the model simplifies to a 3-state definition: PrivSecure > UserSecure > UserNonsecure
- Programming model and hardware implementation is distributed across multiple submodules
    - Supports a broad, highly-configurable architecture definition
    - Memory region descriptors support a format leveraged from earlier System Memory Protection Units (SMPU)

## 9.2.2 Block diagram

As previously noted, the XRDC implementation is distributed across multiple submodules instantiated throughout the device. The XRDC submodules include:

- XRDC_MGR
    - The Manager (MGR) submodule coordinates all programming model reads and writes
- XRDC_MDAC
    - The Master Domain Assignment Controller (MDAC) handles resource assignments and generation of the domain identifiers
- XRDC_MRC
    - The Memory Region Controller (MRC) implements the access controls for slave memories based on the pre-programmed region descriptor registers
- XRDC_PAC
    - The Peripheral Access Controller (PAC) implements the access controls for slave peripherals based on the pre-programmed domain access control registers
- XRDC_MSC
    - The Memory Slot Controller (MSC) implements the access controls for memory slots based on the pre-programmed domain access control registers

For chip-specific implementation details of this module's instances, see the chip specific information.

See Figure 9-1 for a simplified XRDC block diagram focusing on topology and connections.

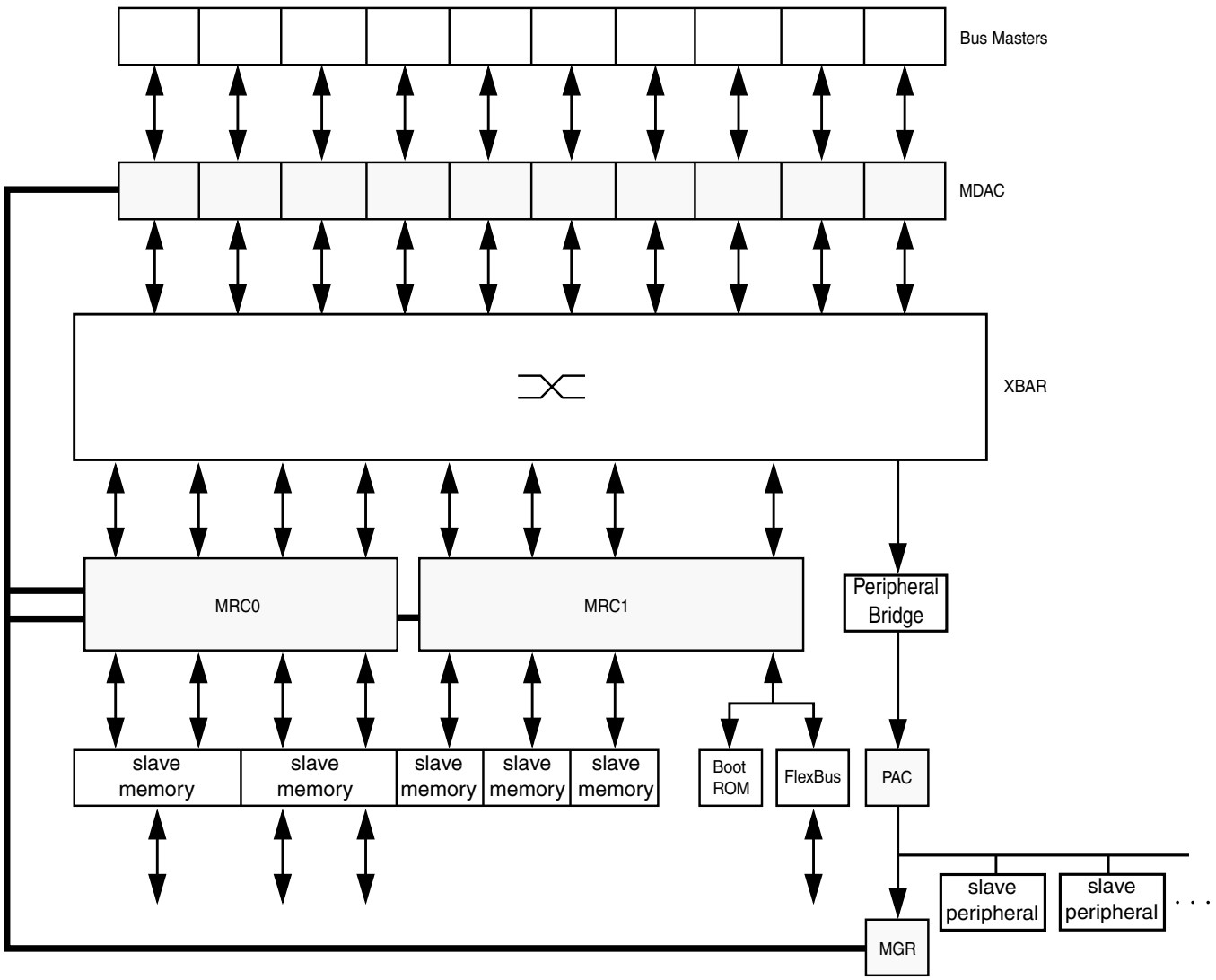



**Figure 9-1. Simplified XRDC Block Diagram**

## 9.2.3 Modes of operation

The XRDC module does not support any special modes of operation. As a memory-mapped device located in the core platform's clock domain, it responds based strictly on the memory addresses of the connected system buses. The slave peripheral bus is used to access the XRDC's distributed programming model; these references are only allowed if the requesting bus master is operating in the Privilege mode. Domain assignment and

access control functions are evaluated on a reference-by-reference basis using the addresses and attributes connected to the system bus port(s). The implementation of the XRDC_MRC submodule supports either AMBA-AHB or AMBA-AXI system bus protocols.

## 9.3 External signal description

The XRDC module does not directly support any external interfaces.

The *internal* interfaces include a standard 32-bit slave bus for all programming model accesses, connections to the address phase signals associated with AHB and/or AXI system buses and connections to the slave peripheral buses as shown in XRDC block diagram.

## 9.4 Memory map/register definition

The XRDC module supports a large number of architecturally-defined, program-visible registers distributed across four 4 Kbyte address slots. The actual registers present in any given device are application (and SoC) specific.

The register resources of the XRDC programming model can be accessed in privileged secure and privileged nonsecure modes. Unless noted otherwise, the programming model registers can be accessed via 8-, 16- or 32-bit reads and 32-bit write references. Attempted accesses in a different operating mode, using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error unless noted otherwise. The XRDC programming model is partitioned into 5 groups of registers:

- Basic hardware control and configuration
- Domain errors: location and details
- Master domain assignments
- Peripheral domain access controls
- Memory region descriptors

It should be noted that many of the programming model registers in the XRDC are organized as 2-dimensional data structures. These generic arrays contain "m" words representing the "columns" and "n" instances of the structure representing the "rows". These may be described as `structure[n][m]`. They appear in the address space of the programming model memory map in the standard C language row-major layout, that is,

the "m" words representing the "column" appear sequentially with the entire row replicated "n" times. The XRDC register structure uses the naming convention of *XRDC_<regname>_Wm_n*, where the column number "m" appears as a numeric suffix on the W (32-bit word) identifier, and the row identifier "n" appears as the final numerical suffix.

The architectural definition of the complete XRDC programming model is presented in this section. Each implementation is SoC-specific and highly dependent on the device requirements. For the reset values in this section, a "*" indicates an initial value determined by the hardware configuration.

## 9.4.1 XRDC register descriptions

### 9.4.1.1 XRDC Memory map

XRDC: 4001_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control Register (CR) | 32 | RW | 0000_000Ch |
| F0h | Hardware Configuration Register 0 (HWCFG0) | 32 | RO | 1201_0B02h |
| F4h | Hardware Configuration Register 1 (HWCFG1) | 32 | RO | See description. |
| F8h | Hardware Configuration Register 2 (HWCFG2) | 32 | RO | 0000_0000h |
| FCh | Hardware Configuration Register 3 (HWCFG3) | 32 | RO | 0000_0000h |
| 100h - 124h | Master Domain Assignment Configuration Register (MDACFG0 - MDACFG36) | 8 | RO | 02h |
| 140h - 141h | Memory Region Configuration Register (MRCFG0 - MRCFG1) | 8 | RO | 08h |
| 1FCh | Fault Domain ID (FDID) | 32 | RW | 0000_0000h |
| 200h - 208h | Domain Error Location Register (DERRLOC0 - DERRLOC2) | 32 | RO | 0000_0000h |
| 400h | Domain Error Word0 Register (DERR_W0_0) | 32 | RO | 0000_0000h |
| 404h | Domain Error Word1 Register (DERR_W1_0) | 32 | RO | 0000_0000h |
| 40Ch | Domain Error Word3 Register (DERR_W3_0) | 32 | WORZ | 0000_0000h |
| 410h | Domain Error Word0 Register (DERR_W0_1) | 32 | RO | 0000_0000h |
| 414h | Domain Error Word1 Register (DERR_W1_1) | 32 | RO | 0000_0000h |
| 41Ch | Domain Error Word3 Register (DERR_W3_1) | 32 | WORZ | 0000_0000h |
| 500h | Domain Error Word0 Register (DERR_W0_16) | 32 | RO | 0000_0000h |
| 504h | Domain Error Word1 Register (DERR_W1_16) | 32 | RO | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 50Ch | Domain Error Word3 Register (DERR_W3_16) | 32 | WORZ | 0000_0000h |
| 510h | Domain Error Word0 Register (DERR_W0_17) | 32 | RO | 0000_0000h |
| 514h | Domain Error Word1 Register (DERR_W1_17) | 32 | RO | 0000_0000h |
| 51Ch | Domain Error Word3 Register (DERR_W3_17) | 32 | WORZ | 0000_0000h |
| 520h | Domain Error Word0 Register (DERR_W0_18) | 32 | RO | 0000_0000h |
| 524h | Domain Error Word1 Register (DERR_W1_18) | 32 | RO | 0000_0000h |
| 52Ch | Domain Error Word3 Register (DERR_W3_18) | 32 | WORZ | 0000_0000h |
| 700h | Process Identifier (PID0) | 32 | RW | 0000_0000h |
| 704h | Process Identifier (PID1) | 32 | RW | 0000_0000h |
| 740h | Process Identifier (PID16) | 32 | RW | 0000_0000h |
| 744h | Process Identifier (PID17) | 32 | RW | 0000_0000h |
| 780h | Process Identifier (PID32) | 32 | RW | 0000_0000h |
| 78Ch | Process Identifier (PID35) | 32 | RW | 0000_0000h |
| 790h | Process Identifier (PID36) | 32 | RW | 0000_0000h |
| 800h | Master Domain Assignment (MDA_W0_0_DFMT0) | 32 | RW | 0000_0000h |
| 804h | Master Domain Assignment (MDA_W1_0_DFMT0) | 32 | RW | 0000_0000h |
| 820h | Master Domain Assignment (MDA_W0_1_DFMT0) | 32 | RW | 0000_0000h |
| 824h | Master Domain Assignment (MDA_W1_1_DFMT0) | 32 | RW | 0000_0000h |
| 840h | Master Domain Assignment (MDA_W0_2_DFMT1) | 32 | RW | 2000_0000h |
| 860h | Master Domain Assignment (MDA_W0_3_DFMT1) | 32 | RW | 2000_0000h |
| 880h | Master Domain Assignment (MDA_W0_4_DFMT1) | 32 | RW | 2000_0000h |
| A00h | Master Domain Assignment (MDA_W0_16_DFMT0) | 32 | RW | 0000_0000h |
| A04h | Master Domain Assignment (MDA_W1_16_DFMT0) | 32 | RW | 0000_0000h |
| A20h | Master Domain Assignment (MDA_W0_17_DFMT0) | 32 | RW | 0000_0000h |
| A24h | Master Domain Assignment (MDA_W1_17_DFMT0) | 32 | RW | 0000_0000h |
| C00h | Master Domain Assignment (MDA_W0_32_DFMT0) | 32 | RW | 0000_0000h |
| C04h | Master Domain Assignment (MDA_W1_32_DFMT0) | 32 | RW | 0000_0000h |
| C20h | Master Domain Assignment (MDA_W0_33_DFMT1) | 32 | RW | 2000_0000h |
| C40h | Master Domain Assignment (MDA_W0_34_DFMT1) | 32 | RW | 2000_0000h |
| C60h | Master Domain Assignment (MDA_W0_35_DFMT0) | 32 | RW | 0000_0000h |
| C64h | Master Domain Assignment (MDA_W1_35_DFMT0) | 32 | RW | 0000_0000h |
| C80h | Master Domain Assignment (MDA_W0_36_DFMT0) | 32 | RW | 0000_0000h |
| C84h | Master Domain Assignment (MDA_W1_36_DFMT0) | 32 | RW | 0000_0000h |
| 1008h | Peripheral Domain Access Control (PDAC_W0_0_1) | 32 | RW | 0000_0000h |
| 100Ch | Peripheral Domain Access Control (PDAC_W1_0_1) | 32 | RW | 0000_0000h |
| 1018h | Peripheral Domain Access Control (PDAC_W0_0_3) | 32 | RW | 0000_0000h |
| 101Ch | Peripheral Domain Access Control (PDAC_W1_0_3) | 32 | RW | 0000_0000h |
| 1020h | Peripheral Domain Access Control (PDAC_W0_0_4) | 32 | RW | 0000_0000h |
| 1024h | Peripheral Domain Access Control (PDAC_W1_0_4) | 32 | RW | 0000_0000h |
| 1040h | Peripheral Domain Access Control (PDAC_W0_0_8) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1044h | Peripheral Domain Access Control (PDAC_W1_0_8) | 32 | RW | 0000_0000h |
| 1048h | Peripheral Domain Access Control (PDAC_W0_0_9) | 32 | RW | 0000_0000h |
| 104Ch | Peripheral Domain Access Control (PDAC_W1_0_9) | 32 | RW | 0000_0000h |
| 1060h | Peripheral Domain Access Control (PDAC_W0_0_12) | 32 | RW | 0000_0000h |
| 1064h | Peripheral Domain Access Control (PDAC_W1_0_12) | 32 | RW | 0000_0000h |
| 1078h | Peripheral Domain Access Control (PDAC_W0_0_15) | 32 | RW | 0000_0000h |
| 107Ch | Peripheral Domain Access Control (PDAC_W1_0_15) | 32 | RW | 0000_0000h |
| 10A0h | Peripheral Domain Access Control (PDAC_W0_0_20) | 32 | RW | 0000_0000h |
| 10A4h | Peripheral Domain Access Control (PDAC_W1_0_20) | 32 | RW | 0000_0000h |
| 10A8h | Peripheral Domain Access Control (PDAC_W0_0_21) | 32 | RW | 0000_0000h |
| 10ACh | Peripheral Domain Access Control (PDAC_W1_0_21) | 32 | RW | 0000_0000h |
| 10B0h | Peripheral Domain Access Control (PDAC_W0_0_22) | 32 | RW | 0000_0000h |
| 10B4h | Peripheral Domain Access Control (PDAC_W1_0_22) | 32 | RW | 0000_0000h |
| 10B8h | Peripheral Domain Access Control (PDAC_W0_0_23) | 32 | RW | 0000_0000h |
| 10BCh | Peripheral Domain Access Control (PDAC_W1_0_23) | 32 | RW | 0000_0000h |
| 10D8h | Peripheral Domain Access Control (PDAC_W0_0_27) | 32 | RW | 0000_0000h |
| 10DCh | Peripheral Domain Access Control (PDAC_W1_0_27) | 32 | RW | 0000_0000h |
| 1100h | Peripheral Domain Access Control (PDAC_W0_0_32) | 32 | RW | 0000_0000h |
| 1104h | Peripheral Domain Access Control (PDAC_W1_0_32) | 32 | RW | 0000_0000h |
| 1108h | Peripheral Domain Access Control (PDAC_W0_0_33) | 32 | RW | 0000_0000h |
| 110Ch | Peripheral Domain Access Control (PDAC_W1_0_33) | 32 | RW | 0000_0000h |
| 1110h | Peripheral Domain Access Control (PDAC_W0_0_34) | 32 | RW | 0000_0000h |
| 1114h | Peripheral Domain Access Control (PDAC_W1_0_34) | 32 | RW | 0000_0000h |
| 1118h | Peripheral Domain Access Control (PDAC_W0_0_35) | 32 | RW | 0000_0000h |
| 111Ch | Peripheral Domain Access Control (PDAC_W1_0_35) | 32 | RW | 0000_0000h |
| 1120h | Peripheral Domain Access Control (PDAC_W0_0_36) | 32 | RW | 0000_0000h |
| 1124h | Peripheral Domain Access Control (PDAC_W1_0_36) | 32 | RW | 0000_0000h |
| 1128h | Peripheral Domain Access Control (PDAC_W0_0_37) | 32 | RW | 0000_0000h |
| 112Ch | Peripheral Domain Access Control (PDAC_W1_0_37) | 32 | RW | 0000_0000h |
| 1130h | Peripheral Domain Access Control (PDAC_W0_0_38) | 32 | RW | 0000_0000h |
| 1134h | Peripheral Domain Access Control (PDAC_W1_0_38) | 32 | RW | 0000_0000h |
| 1138h | Peripheral Domain Access Control (PDAC_W0_0_39) | 32 | RW | 0000_0000h |
| 113Ch | Peripheral Domain Access Control (PDAC_W1_0_39) | 32 | RW | 0000_0000h |
| 1140h | Peripheral Domain Access Control (PDAC_W0_0_40) | 32 | RW | 0000_0000h |
| 1144h | Peripheral Domain Access Control (PDAC_W1_0_40) | 32 | RW | 0000_0000h |
| 1148h | Peripheral Domain Access Control (PDAC_W0_0_41) | 32 | RW | 0000_0000h |
| 114Ch | Peripheral Domain Access Control (PDAC_W1_0_41) | 32 | RW | 0000_0000h |
| 1150h | Peripheral Domain Access Control (PDAC_W0_0_42) | 32 | RW | 0000_0000h |
| 1154h | Peripheral Domain Access Control (PDAC_W1_0_42) | 32 | RW | 0000_0000h |
| 1158h | Peripheral Domain Access Control (PDAC_W0_0_43) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 115Ch | Peripheral Domain Access Control (PDAC_W1_0_43) | 32 | RW | 0000_0000h |
| 1160h | Peripheral Domain Access Control (PDAC_W0_0_44) | 32 | RW | 0000_0000h |
| 1164h | Peripheral Domain Access Control (PDAC_W1_0_44) | 32 | RW | 0000_0000h |
| 1168h | Peripheral Domain Access Control (PDAC_W0_0_45) | 32 | RW | 0000_0000h |
| 116Ch | Peripheral Domain Access Control (PDAC_W1_0_45) | 32 | RW | 0000_0000h |
| 1170h | Peripheral Domain Access Control (PDAC_W0_0_46) | 32 | RW | 0000_0000h |
| 1174h | Peripheral Domain Access Control (PDAC_W1_0_46) | 32 | RW | 0000_0000h |
| 1178h | Peripheral Domain Access Control (PDAC_W0_0_47) | 32 | RW | 0000_0000h |
| 117Ch | Peripheral Domain Access Control (PDAC_W1_0_47) | 32 | RW | 0000_0000h |
| 1180h | Peripheral Domain Access Control (PDAC_W0_0_48) | 32 | RW | 0000_0000h |
| 1184h | Peripheral Domain Access Control (PDAC_W1_0_48) | 32 | RW | 0000_0000h |
| 1188h | Peripheral Domain Access Control (PDAC_W0_0_49) | 32 | RW | 0000_0000h |
| 118Ch | Peripheral Domain Access Control (PDAC_W1_0_49) | 32 | RW | 0000_0000h |
| 1190h | Peripheral Domain Access Control (PDAC_W0_0_50) | 32 | RW | 0000_0000h |
| 1194h | Peripheral Domain Access Control (PDAC_W1_0_50) | 32 | RW | 0000_0000h |
| 1198h | Peripheral Domain Access Control (PDAC_W0_0_51) | 32 | RW | 0000_0000h |
| 119Ch | Peripheral Domain Access Control (PDAC_W1_0_51) | 32 | RW | 0000_0000h |
| 11A0h | Peripheral Domain Access Control (PDAC_W0_0_52) | 32 | RW | 0000_0000h |
| 11A4h | Peripheral Domain Access Control (PDAC_W1_0_52) | 32 | RW | 0000_0000h |
| 11A8h | Peripheral Domain Access Control (PDAC_W0_0_53) | 32 | RW | 0000_0000h |
| 11ACh | Peripheral Domain Access Control (PDAC_W1_0_53) | 32 | RW | 0000_0000h |
| 11B0h | Peripheral Domain Access Control (PDAC_W0_0_54) | 32 | RW | 0000_0000h |
| 11B4h | Peripheral Domain Access Control (PDAC_W1_0_54) | 32 | RW | 0000_0000h |
| 11B8h | Peripheral Domain Access Control (PDAC_W0_0_55) | 32 | RW | 0000_0000h |
| 11BCh | Peripheral Domain Access Control (PDAC_W1_0_55) | 32 | RW | 0000_0000h |
| 11C0h | Peripheral Domain Access Control (PDAC_W0_0_56) | 32 | RW | 0000_0000h |
| 11C4h | Peripheral Domain Access Control (PDAC_W1_0_56) | 32 | RW | 0000_0000h |
| 11C8h | Peripheral Domain Access Control (PDAC_W0_0_57) | 32 | RW | 0000_0000h |
| 11CCh | Peripheral Domain Access Control (PDAC_W1_0_57) | 32 | RW | 0000_0000h |
| 11D0h | Peripheral Domain Access Control (PDAC_W0_0_58) | 32 | RW | 0000_0000h |
| 11D4h | Peripheral Domain Access Control (PDAC_W1_0_58) | 32 | RW | 0000_0000h |
| 11D8h | Peripheral Domain Access Control (PDAC_W0_0_59) | 32 | RW | 0000_0000h |
| 11DCh | Peripheral Domain Access Control (PDAC_W1_0_59) | 32 | RW | 0000_0000h |
| 11E0h | Peripheral Domain Access Control (PDAC_W0_0_60) | 32 | RW | 0000_0000h |
| 11E4h | Peripheral Domain Access Control (PDAC_W1_0_60) | 32 | RW | 0000_0000h |
| 11E8h | Peripheral Domain Access Control (PDAC_W0_0_61) | 32 | RW | 0000_0000h |
| 11ECh | Peripheral Domain Access Control (PDAC_W1_0_61) | 32 | RW | 0000_0000h |
| 11F0h | Peripheral Domain Access Control (PDAC_W0_0_62) | 32 | RW | 0000_0000h |
| 11F4h | Peripheral Domain Access Control (PDAC_W1_0_62) | 32 | RW | 0000_0000h |
| 11F8h | Peripheral Domain Access Control (PDAC_W0_0_63) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 11FCh | Peripheral Domain Access Control (PDAC_W1_0_63) | 32 | RW | 0000_0000h |
| 1200h | Peripheral Domain Access Control (PDAC_W0_0_64) | 32 | RW | 0000_0000h |
| 1204h | Peripheral Domain Access Control (PDAC_W1_0_64) | 32 | RW | 0000_0000h |
| 1208h | Peripheral Domain Access Control (PDAC_W0_0_65) | 32 | RW | 0000_0000h |
| 120Ch | Peripheral Domain Access Control (PDAC_W1_0_65) | 32 | RW | 0000_0000h |
| 1210h | Peripheral Domain Access Control (PDAC_W0_0_66) | 32 | RW | 0000_0000h |
| 1214h | Peripheral Domain Access Control (PDAC_W1_0_66) | 32 | RW | 0000_0000h |
| 1218h | Peripheral Domain Access Control (PDAC_W0_0_67) | 32 | RW | 0000_0000h |
| 121Ch | Peripheral Domain Access Control (PDAC_W1_0_67) | 32 | RW | 0000_0000h |
| 1220h | Peripheral Domain Access Control (PDAC_W0_0_68) | 32 | RW | 0000_0000h |
| 1224h | Peripheral Domain Access Control (PDAC_W1_0_68) | 32 | RW | 0000_0000h |
| 1228h | Peripheral Domain Access Control (PDAC_W0_0_69) | 32 | RW | 0000_0000h |
| 122Ch | Peripheral Domain Access Control (PDAC_W1_0_69) | 32 | RW | 0000_0000h |
| 1230h | Peripheral Domain Access Control (PDAC_W0_0_70) | 32 | RW | 0000_0000h |
| 1234h | Peripheral Domain Access Control (PDAC_W1_0_70) | 32 | RW | 0000_0000h |
| 1238h | Peripheral Domain Access Control (PDAC_W0_0_71) | 32 | RW | 0000_0000h |
| 123Ch | Peripheral Domain Access Control (PDAC_W1_0_71) | 32 | RW | 0000_0000h |
| 1240h | Peripheral Domain Access Control (PDAC_W0_0_72) | 32 | RW | 0000_0000h |
| 1244h | Peripheral Domain Access Control (PDAC_W1_0_72) | 32 | RW | 0000_0000h |
| 1248h | Peripheral Domain Access Control (PDAC_W0_0_73) | 32 | RW | 0000_0000h |
| 124Ch | Peripheral Domain Access Control (PDAC_W1_0_73) | 32 | RW | 0000_0000h |
| 1250h | Peripheral Domain Access Control (PDAC_W0_0_74) | 32 | RW | 0000_0000h |
| 1254h | Peripheral Domain Access Control (PDAC_W1_0_74) | 32 | RW | 0000_0000h |
| 1258h | Peripheral Domain Access Control (PDAC_W0_0_75) | 32 | RW | 0000_0000h |
| 125Ch | Peripheral Domain Access Control (PDAC_W1_0_75) | 32 | RW | 0000_0000h |
| 1260h | Peripheral Domain Access Control (PDAC_W0_0_76) | 32 | RW | 0000_0000h |
| 1264h | Peripheral Domain Access Control (PDAC_W1_0_76) | 32 | RW | 0000_0000h |
| 1268h | Peripheral Domain Access Control (PDAC_W0_0_77) | 32 | RW | 0000_0000h |
| 126Ch | Peripheral Domain Access Control (PDAC_W1_0_77) | 32 | RW | 0000_0000h |
| 1270h | Peripheral Domain Access Control (PDAC_W0_0_78) | 32 | RW | 0000_0000h |
| 1274h | Peripheral Domain Access Control (PDAC_W1_0_78) | 32 | RW | 0000_0000h |
| 1278h | Peripheral Domain Access Control (PDAC_W0_0_79) | 32 | RW | 0000_0000h |
| 127Ch | Peripheral Domain Access Control (PDAC_W1_0_79) | 32 | RW | 0000_0000h |
| 13E8h | Peripheral Domain Access Control (PDAC_W0_0_125) | 32 | RW | 0000_0000h |
| 13ECh | Peripheral Domain Access Control (PDAC_W1_0_125) | 32 | RW | 0000_0000h |
| 13F0h | Peripheral Domain Access Control (PDAC_W0_0_126) | 32 | RW | 0000_0000h |
| 13F4h | Peripheral Domain Access Control (PDAC_W1_0_126) | 32 | RW | 0000_0000h |
| 13F8h | Peripheral Domain Access Control (PDAC_W0_0_127) | 32 | RW | 0000_0000h |
| 13FCh | Peripheral Domain Access Control (PDAC_W1_0_127) | 32 | RW | 0000_0000h |
| 1440h | Peripheral Domain Access Control (PDAC_W0_1_8) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1444h | Peripheral Domain Access Control (PDAC_W1_1_8) | 32 | RW | 0000_0000h |
| 1448h | Peripheral Domain Access Control (PDAC_W0_1_9) | 32 | RW | 0000_0000h |
| 144Ch | Peripheral Domain Access Control (PDAC_W1_1_9) | 32 | RW | 0000_0000h |
| 1478h | Peripheral Domain Access Control (PDAC_W0_1_15) | 32 | RW | 0000_0000h |
| 147Ch | Peripheral Domain Access Control (PDAC_W1_1_15) | 32 | RW | 0000_0000h |
| 14D8h | Peripheral Domain Access Control (PDAC_W0_1_27) | 32 | RW | 0000_0000h |
| 14DCh | Peripheral Domain Access Control (PDAC_W1_1_27) | 32 | RW | 0000_0000h |
| 14F8h | Peripheral Domain Access Control (PDAC_W0_1_31) | 32 | RW | 0000_0000h |
| 14FCh | Peripheral Domain Access Control (PDAC_W1_1_31) | 32 | RW | 0000_0000h |
| 1500h | Peripheral Domain Access Control (PDAC_W0_1_32) | 32 | RW | 0000_0000h |
| 1504h | Peripheral Domain Access Control (PDAC_W1_1_32) | 32 | RW | 0000_0000h |
| 1508h | Peripheral Domain Access Control (PDAC_W0_1_33) | 32 | RW | 0000_0000h |
| 150Ch | Peripheral Domain Access Control (PDAC_W1_1_33) | 32 | RW | 0000_0000h |
| 1510h | Peripheral Domain Access Control (PDAC_W0_1_34) | 32 | RW | 0000_0000h |
| 1514h | Peripheral Domain Access Control (PDAC_W1_1_34) | 32 | RW | 0000_0000h |
| 1518h | Peripheral Domain Access Control (PDAC_W0_1_35) | 32 | RW | 0000_0000h |
| 151Ch | Peripheral Domain Access Control (PDAC_W1_1_35) | 32 | RW | 0000_0000h |
| 1520h | Peripheral Domain Access Control (PDAC_W0_1_36) | 32 | RW | 0000_0000h |
| 1524h | Peripheral Domain Access Control (PDAC_W1_1_36) | 32 | RW | 0000_0000h |
| 1528h | Peripheral Domain Access Control (PDAC_W0_1_37) | 32 | RW | 0000_0000h |
| 152Ch | Peripheral Domain Access Control (PDAC_W1_1_37) | 32 | RW | 0000_0000h |
| 1530h | Peripheral Domain Access Control (PDAC_W0_1_38) | 32 | RW | 0000_0000h |
| 1534h | Peripheral Domain Access Control (PDAC_W1_1_38) | 32 | RW | 0000_0000h |
| 1538h | Peripheral Domain Access Control (PDAC_W0_1_39) | 32 | RW | 0000_0000h |
| 153Ch | Peripheral Domain Access Control (PDAC_W1_1_39) | 32 | RW | 0000_0000h |
| 1540h | Peripheral Domain Access Control (PDAC_W0_1_40) | 32 | RW | 0000_0000h |
| 1544h | Peripheral Domain Access Control (PDAC_W1_1_40) | 32 | RW | 0000_0000h |
| 1548h | Peripheral Domain Access Control (PDAC_W0_1_41) | 32 | RW | 0000_0000h |
| 154Ch | Peripheral Domain Access Control (PDAC_W1_1_41) | 32 | RW | 0000_0000h |
| 1550h | Peripheral Domain Access Control (PDAC_W0_1_42) | 32 | RW | 0000_0000h |
| 1554h | Peripheral Domain Access Control (PDAC_W1_1_42) | 32 | RW | 0000_0000h |
| 1558h | Peripheral Domain Access Control (PDAC_W0_1_43) | 32 | RW | 0000_0000h |
| 155Ch | Peripheral Domain Access Control (PDAC_W1_1_43) | 32 | RW | 0000_0000h |
| 1560h | Peripheral Domain Access Control (PDAC_W0_1_44) | 32 | RW | 0000_0000h |
| 1564h | Peripheral Domain Access Control (PDAC_W1_1_44) | 32 | RW | 0000_0000h |
| 1568h | Peripheral Domain Access Control (PDAC_W0_1_45) | 32 | RW | 0000_0000h |
| 156Ch | Peripheral Domain Access Control (PDAC_W1_1_45) | 32 | RW | 0000_0000h |
| 1570h | Peripheral Domain Access Control (PDAC_W0_1_46) | 32 | RW | 0000_0000h |
| 1574h | Peripheral Domain Access Control (PDAC_W1_1_46) | 32 | RW | 0000_0000h |
| 1578h | Peripheral Domain Access Control (PDAC_W0_1_47) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 157Ch | Peripheral Domain Access Control (PDAC_W1_1_47) | 32 | RW | 0000_0000h |
| 1580h | Peripheral Domain Access Control (PDAC_W0_1_48) | 32 | RW | 0000_0000h |
| 1584h | Peripheral Domain Access Control (PDAC_W1_1_48) | 32 | RW | 0000_0000h |
| 1588h | Peripheral Domain Access Control (PDAC_W0_1_49) | 32 | RW | 0000_0000h |
| 158Ch | Peripheral Domain Access Control (PDAC_W1_1_49) | 32 | RW | 0000_0000h |
| 1590h | Peripheral Domain Access Control (PDAC_W0_1_50) | 32 | RW | 0000_0000h |
| 1594h | Peripheral Domain Access Control (PDAC_W1_1_50) | 32 | RW | 0000_0000h |
| 1598h | Peripheral Domain Access Control (PDAC_W0_1_51) | 32 | RW | 0000_0000h |
| 159Ch | Peripheral Domain Access Control (PDAC_W1_1_51) | 32 | RW | 0000_0000h |
| 15A0h | Peripheral Domain Access Control (PDAC_W0_1_52) | 32 | RW | 0000_0000h |
| 15A4h | Peripheral Domain Access Control (PDAC_W1_1_52) | 32 | RW | 0000_0000h |
| 15A8h | Peripheral Domain Access Control (PDAC_W0_1_53) | 32 | RW | 0000_0000h |
| 15ACh | Peripheral Domain Access Control (PDAC_W1_1_53) | 32 | RW | 0000_0000h |
| 15B0h | Peripheral Domain Access Control (PDAC_W0_1_54) | 32 | RW | 0000_0000h |
| 15B4h | Peripheral Domain Access Control (PDAC_W1_1_54) | 32 | RW | 0000_0000h |
| 15B8h | Peripheral Domain Access Control (PDAC_W0_1_55) | 32 | RW | 0000_0000h |
| 15BCh | Peripheral Domain Access Control (PDAC_W1_1_55) | 32 | RW | 0000_0000h |
| 15C0h | Peripheral Domain Access Control (PDAC_W0_1_56) | 32 | RW | 0000_0000h |
| 15C4h | Peripheral Domain Access Control (PDAC_W1_1_56) | 32 | RW | 0000_0000h |
| 1880h | Peripheral Domain Access Control (PDAC_W0_2_16) | 32 | RW | 0000_0000h |
| 1884h | Peripheral Domain Access Control (PDAC_W1_2_16) | 32 | RW | 0000_0000h |
| 1900h | Peripheral Domain Access Control (PDAC_W0_2_32) | 32 | RW | 0000_0000h |
| 1904h | Peripheral Domain Access Control (PDAC_W1_2_32) | 32 | RW | 0000_0000h |
| 2000h | Memory Region Descriptor (MRGD_W0_0_0) | 32 | RW | 0000_0000h |
| 2004h | Memory Region Descriptor (MRGD_W1_0_0) | 32 | RW | 0000_001Fh |
| 2008h | Memory Region Descriptor (MRGD_W2_0_0) | 32 | RW | 0000_0000h |
| 200Ch | Memory Region Descriptor (MRGD_W3_0_0) | 32 | RW | 0000_0000h |
| 2010h | Memory Region Descriptor (MRGD_W4_0_0) | 32 | RW | 0000_0000h |
| 2020h | Memory Region Descriptor (MRGD_W0_0_1) | 32 | RW | 0000_0000h |
| 2024h | Memory Region Descriptor (MRGD_W1_0_1) | 32 | RW | 0000_001Fh |
| 2028h | Memory Region Descriptor (MRGD_W2_0_1) | 32 | RW | 0000_0000h |
| 202Ch | Memory Region Descriptor (MRGD_W3_0_1) | 32 | RW | 0000_0000h |
| 2030h | Memory Region Descriptor (MRGD_W4_0_1) | 32 | RW | 0000_0000h |
| 2040h | Memory Region Descriptor (MRGD_W0_0_2) | 32 | RW | 0000_0000h |
| 2044h | Memory Region Descriptor (MRGD_W1_0_2) | 32 | RW | 0000_001Fh |
| 2048h | Memory Region Descriptor (MRGD_W2_0_2) | 32 | RW | 0000_0000h |
| 204Ch | Memory Region Descriptor (MRGD_W3_0_2) | 32 | RW | 0000_0000h |
| 2050h | Memory Region Descriptor (MRGD_W4_0_2) | 32 | RW | 0000_0000h |
| 2060h | Memory Region Descriptor (MRGD_W0_0_3) | 32 | RW | 0000_0000h |
| 2064h | Memory Region Descriptor (MRGD_W1_0_3) | 32 | RW | 0000_001Fh |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2068h | Memory Region Descriptor (MRGD_W2_0_3) | 32 | RW | 0000_0000h |
| 206Ch | Memory Region Descriptor (MRGD_W3_0_3) | 32 | RW | 0000_0000h |
| 2070h | Memory Region Descriptor (MRGD_W4_0_3) | 32 | RW | 0000_0000h |
| 2080h | Memory Region Descriptor (MRGD_W0_0_4) | 32 | RW | 0000_0000h |
| 2084h | Memory Region Descriptor (MRGD_W1_0_4) | 32 | RW | 0000_001Fh |
| 2088h | Memory Region Descriptor (MRGD_W2_0_4) | 32 | RW | 0000_0000h |
| 208Ch | Memory Region Descriptor (MRGD_W3_0_4) | 32 | RW | 0000_0000h |
| 2090h | Memory Region Descriptor (MRGD_W4_0_4) | 32 | RW | 0000_0000h |
| 20A0h | Memory Region Descriptor (MRGD_W0_0_5) | 32 | RW | 0000_0000h |
| 20A4h | Memory Region Descriptor (MRGD_W1_0_5) | 32 | RW | 0000_001Fh |
| 20A8h | Memory Region Descriptor (MRGD_W2_0_5) | 32 | RW | 0000_0000h |
| 20ACh | Memory Region Descriptor (MRGD_W3_0_5) | 32 | RW | 0000_0000h |
| 20B0h | Memory Region Descriptor (MRGD_W4_0_5) | 32 | RW | 0000_0000h |
| 20C0h | Memory Region Descriptor (MRGD_W0_0_6) | 32 | RW | 0000_0000h |
| 20C4h | Memory Region Descriptor (MRGD_W1_0_6) | 32 | RW | 0000_001Fh |
| 20C8h | Memory Region Descriptor (MRGD_W2_0_6) | 32 | RW | 0000_0000h |
| 20CCh | Memory Region Descriptor (MRGD_W3_0_6) | 32 | RW | 0000_0000h |
| 20D0h | Memory Region Descriptor (MRGD_W4_0_6) | 32 | RW | 0000_0000h |
| 20E0h | Memory Region Descriptor (MRGD_W0_0_7) | 32 | RW | 0000_0000h |
| 20E4h | Memory Region Descriptor (MRGD_W1_0_7) | 32 | RW | 0000_001Fh |
| 20E8h | Memory Region Descriptor (MRGD_W2_0_7) | 32 | RW | 0000_0000h |
| 20ECh | Memory Region Descriptor (MRGD_W3_0_7) | 32 | RW | 0000_0000h |
| 20F0h | Memory Region Descriptor (MRGD_W4_0_7) | 32 | RW | 0000_0000h |
| 2200h | Memory Region Descriptor (MRGD_W0_1_0) | 32 | RW | 0000_0000h |
| 2204h | Memory Region Descriptor (MRGD_W1_1_0) | 32 | RW | 0000_001Fh |
| 2208h | Memory Region Descriptor (MRGD_W2_1_0) | 32 | RW | 0000_0000h |
| 220Ch | Memory Region Descriptor (MRGD_W3_1_0) | 32 | RW | 0000_0000h |
| 2210h | Memory Region Descriptor (MRGD_W4_1_0) | 32 | RW | 0000_0000h |
| 2220h | Memory Region Descriptor (MRGD_W0_1_1) | 32 | RW | 0000_0000h |
| 2224h | Memory Region Descriptor (MRGD_W1_1_1) | 32 | RW | 0000_001Fh |
| 2228h | Memory Region Descriptor (MRGD_W2_1_1) | 32 | RW | 0000_0000h |
| 222Ch | Memory Region Descriptor (MRGD_W3_1_1) | 32 | RW | 0000_0000h |
| 2230h | Memory Region Descriptor (MRGD_W4_1_1) | 32 | RW | 0000_0000h |
| 2240h | Memory Region Descriptor (MRGD_W0_1_2) | 32 | RW | 0000_0000h |
| 2244h | Memory Region Descriptor (MRGD_W1_1_2) | 32 | RW | 0000_001Fh |
| 2248h | Memory Region Descriptor (MRGD_W2_1_2) | 32 | RW | 0000_0000h |
| 224Ch | Memory Region Descriptor (MRGD_W3_1_2) | 32 | RW | 0000_0000h |
| 2250h | Memory Region Descriptor (MRGD_W4_1_2) | 32 | RW | 0000_0000h |
| 2260h | Memory Region Descriptor (MRGD_W0_1_3) | 32 | RW | 0000_0000h |
| 2264h | Memory Region Descriptor (MRGD_W1_1_3) | 32 | RW | 0000_001Fh |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2268h | Memory Region Descriptor (MRGD_W2_1_3) | 32 | RW | 0000_0000h |
| 226Ch | Memory Region Descriptor (MRGD_W3_1_3) | 32 | RW | 0000_0000h |
| 2270h | Memory Region Descriptor (MRGD_W4_1_3) | 32 | RW | 0000_0000h |
| 2280h | Memory Region Descriptor (MRGD_W0_1_4) | 32 | RW | 0000_0000h |
| 2284h | Memory Region Descriptor (MRGD_W1_1_4) | 32 | RW | 0000_001Fh |
| 2288h | Memory Region Descriptor (MRGD_W2_1_4) | 32 | RW | 0000_0000h |
| 228Ch | Memory Region Descriptor (MRGD_W3_1_4) | 32 | RW | 0000_0000h |
| 2290h | Memory Region Descriptor (MRGD_W4_1_4) | 32 | RW | 0000_0000h |
| 22A0h | Memory Region Descriptor (MRGD_W0_1_5) | 32 | RW | 0000_0000h |
| 22A4h | Memory Region Descriptor (MRGD_W1_1_5) | 32 | RW | 0000_001Fh |
| 22A8h | Memory Region Descriptor (MRGD_W2_1_5) | 32 | RW | 0000_0000h |
| 22ACh | Memory Region Descriptor (MRGD_W3_1_5) | 32 | RW | 0000_0000h |
| 22B0h | Memory Region Descriptor (MRGD_W4_1_5) | 32 | RW | 0000_0000h |
| 22C0h | Memory Region Descriptor (MRGD_W0_1_6) | 32 | RW | 0000_0000h |
| 22C4h | Memory Region Descriptor (MRGD_W1_1_6) | 32 | RW | 0000_001Fh |
| 22C8h | Memory Region Descriptor (MRGD_W2_1_6) | 32 | RW | 0000_0000h |
| 22CCh | Memory Region Descriptor (MRGD_W3_1_6) | 32 | RW | 0000_0000h |
| 22D0h | Memory Region Descriptor (MRGD_W4_1_6) | 32 | RW | 0000_0000h |
| 22E0h | Memory Region Descriptor (MRGD_W0_1_7) | 32 | RW | 0000_0000h |
| 22E4h | Memory Region Descriptor (MRGD_W1_1_7) | 32 | RW | 0000_001Fh |
| 22E8h | Memory Region Descriptor (MRGD_W2_1_7) | 32 | RW | 0000_0000h |
| 22ECh | Memory Region Descriptor (MRGD_W3_1_7) | 32 | RW | 0000_0000h |
| 22F0h | Memory Region Descriptor (MRGD_W4_1_7) | 32 | RW | 0000_0000h |

## 9.4.1.2 Control Register (CR)

### 9.4.1.2.1 Offset

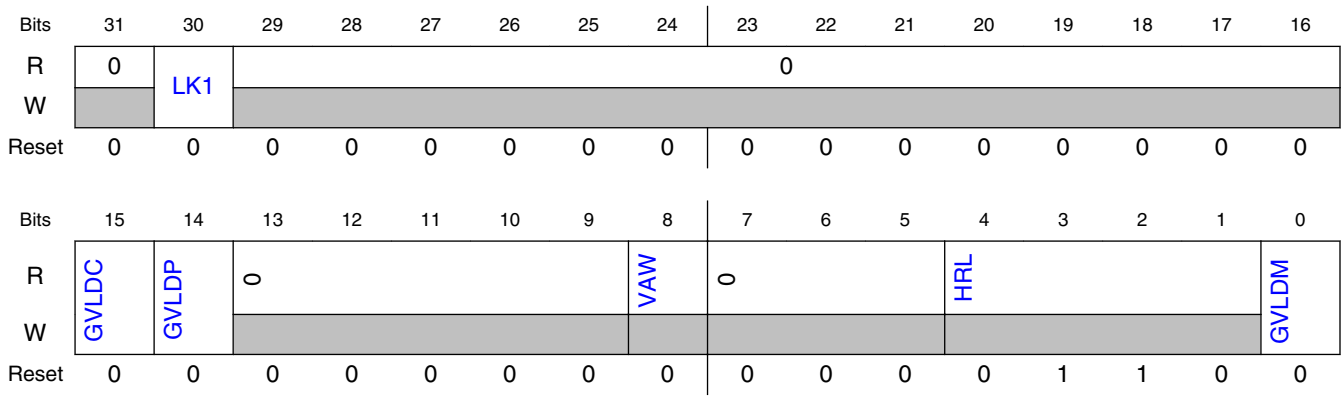
| Register | Offset |
|---|---|
| CR | 0h |

### 9.4.1.2.2 Function

This register provides read-only status about the XRDC and a global enable bit for the entire module's operation.

Access: Privileged Read/write

### 9.4.1.2.3 Diagram




### 9.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>LK1 | 1-bit Lock<br>This field provides a locking mechanism that can be used to limit the ability to write the register. Once set, this bit remains asserted until the next reset.<br>0b - Register can be written by any secure privileged write.<br>1b - Register is locked (read-only) until the next reset. |
| 29-16<br>— | Reserved |
| 15<br>GVLDC | Global Valid for MRCs<br>XRDC global MRC enable/disable<br>0b - XRDC MRCs are disabled.<br>1b - XRDC MRCs are enabled. |
| 14<br>GVLDP | Global Valid for PACs/MSCs<br>XRDC global PAC/MSC enable/disable<br>0b - XRDC PACs/MSCs are disabled.<br>1b - XRDC PACs/MSCs are enabled. |
| 13-9<br>— | Reserved |
| 8<br>VAW | Virtualization aware<br>This field signals that the domain assignments support the optional inclusion of a logical partition identifier (also known as an operating system number).<br>0b - Implementation is not virtualization aware.<br>1b - Implementation is virtualization aware. |
| 7-5<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 4-1<br>HRL | Hardware Revision Level<br>This read-only field specifies the XRDC's hardware and definition revision level. It can be read by software to determine the functional definition of the module. |
| 0<br>GVLDM | Global Valid MDACs(XRDC global enable/disable).<br>0b - XRDC MDACs are disabled.<br>1b - XRDC MDACs are enabled. |

### 9.4.1.3 Hardware Configuration Register 0 (HWCFG0)

#### 9.4.1.3.1 Offset

| Register | Offset |
|---|---|
| HWCFG0 | F0h |

#### 9.4.1.3.2 Function

This read-only register contains information on the XRDC's hardware configuration. Specifically, it defines the number of implemented domains and bus masters along with the number of instances of memory region controllers (MRCs) and peripheral access controllers (PACs). The register value at reset is device-specific. Attempted writes are error terminated.

Access: Privileged Read

#### 9.4.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MID | | | | NPAC | | | | NMRC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NMSTR | | | | | | | | NDID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 9.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31-28<br>MID | Module ID |
| 27-24<br>NPAC | Number of PACs<br>This field defines the number of PACs minus 1 in the device. Add one to the field value to get the actual number of peripheral access controllers [1-4]. |
| 23-16<br>NMRC | Number of MRCs<br>This field defines the number of MRCs minus 1 in the device. Add one to the field value to get the actual number of memory region controllers [1-16]. |
| 15-8<br>NMSTR | Number of bus masters<br>This read-only field defines the number of bus masters minus 1 in the device. Add one to the field value to get the actual number of bus masters [1-64]. |
| 7-0<br>NDID | Number of domains<br>This read-only field defines the number of domains minus 1 in the device. Add one to the field value to get the actual number of domains [1-16]. |

### 9.4.1.4 Hardware Configuration Register 1 (HWCFG1)

#### 9.4.1.4.1 Offset

| Register | Offset |
|---|---|
| HWCFG1 | F4h |

#### 9.4.1.4.2 Function

This register contains information on the XRDC's hardware configuration. It provides a mechanism for software to determine its domain number by simply reading the register. See Domain error capture management for more details on typical usage. Attempted writes are error terminated.

Access: Privileged Read

#### 9.4.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DID | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u |

1. Reset value is determined by the hardware configuration.

#### 9.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>DID | Domain identifier number<br>This field provides the domain number [0-15] of the requesting bus master. |

### 9.4.1.5 Hardware Configuration Register 2 (HWCFG2)

#### 9.4.1.5.1 Offset

| Register | Offset |
|---|---|
| HWCFG2 | F8h |

#### 9.4.1.5.2 Function

This register contains information on the XRDC's hardware configuration. It provides a bitmap signaling the presence of a process identifier (PID) register sourced from the given bus master. The HWCFG2 register is associated with bus masters [31-0]. Attempted writes are error terminated.

Access: Privileged Read

### 9.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDP31 | PIDP30 | PIDP29 | PIDP28 | PIDP27 | PIDP26 | PIDP25 | PIDP24 | PIDP23 | PIDP22 | PIDP21 | PIDP20 | PIDP19 | PIDP18 | PIDP17 | PIDP16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDP15 | PIDP14 | PIDP13 | PIDP12 | PIDP11 | PIDP10 | PIDP9 | PIDP8 | PIDP7 | PIDP6 | PIDP5 | PIDP4 | PIDP3 | PIDP2 | PIDP1 | PIDP0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>PIDP31 | Process identifier<br>Process identifier present from bus master 31. This field provides a bitmap to signal that bus master 31 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 31 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 31 sources a process identifier register to the XRDC_MDAC logic. |
| 30<br>PIDP30 | Process identifier<br>Process identifier present from bus master 30. This field provides a bitmap to signal that bus master 30 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 30 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 30 sources a process identifier register to the XRDC_MDAC logic. |
| 29<br>PIDP29 | Process identifier<br>Process identifier present from bus master 29. This field provides a bitmap to signal that bus master 29 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 29 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 29 sources a process identifier register to the XRDC_MDAC logic. |
| 28<br>PIDP28 | Process identifier<br>Process identifier present from bus master 28. This field provides a bitmap to signal that bus master 28 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 28 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 28 sources a process identifier register to the XRDC_MDAC logic. |
| 27<br>PIDP27 | Process identifier |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Process identifier present from bus master 27. This field provides a bitmap to signal that bus master 27 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 27 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 27 sources a process identifier register to the XRDC_MDAC logic. |
| 26<br>PIDP26 | Process identifier<br>Process identifier present from bus master 26. This field provides a bitmap to signal that bus master 26 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 26 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 26 sources a process identifier register to the XRDC_MDAC logic. |
| 25<br>PIDP25 | Process identifier<br>Process identifier present from bus master 25. This field provides a bitmap to signal that bus master 25 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 25 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 25 sources a process identifier register to the XRDC_MDAC logic. |
| 24<br>PIDP24 | Process identifier<br>Process identifier present from bus master 24. This field provides a bitmap to signal that bus master 24 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 24 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 24 sources a process identifier register to the XRDC_MDAC logic. |
| 23<br>PIDP23 | Process identifier<br>Process identifier present from bus master 23. This field provides a bitmap to signal that bus master 23 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 23 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 23 sources a process identifier register to the XRDC_MDAC logic. |
| 22<br>PIDP22 | Process identifier<br>Process identifier present from bus master 22. This field provides a bitmap to signal that bus master 22 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 22 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 22 sources a process identifier register to the XRDC_MDAC logic. |
| 21<br>PIDP21 | Process identifier<br>Process identifier present from bus master 21. This field provides a bitmap to signal that bus master 21 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 21 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 21 sources a process identifier register to the XRDC_MDAC logic. |
| 20<br>PIDP20 | Process identifier<br>Process identifier present from bus master 20. This field provides a bitmap to signal that bus master 20 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 20 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 20 sources a process identifier register to the XRDC_MDAC logic. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19 | Process identifier |
| PIDP19 | Process identifier present from bus master 19. This field provides a bitmap to signal that bus master 19 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 19 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 19 sources a process identifier register to the XRDC_MDAC logic. |
| 18 | Process identifier |
| PIDP18 | Process identifier present from bus master 18. This field provides a bitmap to signal that bus master 18 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 18 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 18 sources a process identifier register to the XRDC_MDAC logic. |
| 17 | Process identifier |
| PIDP17 | Process identifier present from bus master 17. This field provides a bitmap to signal that bus master 17 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 17 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 17 sources a process identifier register to the XRDC_MDAC logic. |
| 16 | Process identifier |
| PIDP16 | Process identifier present from bus master 16. This field provides a bitmap to signal that bus master 16 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 16 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 16 sources a process identifier register to the XRDC_MDAC logic. |
| 15 | Process identifier |
| PIDP15 | Process identifier present from bus master 15. This field provides a bitmap to signal that bus master 15 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 15 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 15 sources a process identifier register to the XRDC_MDAC logic. |
| 14 | Process identifier |
| PIDP14 | Process identifier present from bus master 14. This field provides a bitmap to signal that bus master 14 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 14 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 14 sources a process identifier register to the XRDC_MDAC logic. |
| 13 | Process identifier |
| PIDP13 | Process identifier present from bus master 13. This field provides a bitmap to signal that bus master 13 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 13 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 13 sources a process identifier register to the XRDC_MDAC logic. |
| 12 | Process identifier |
| PIDP12 | Process identifier present from bus master 12. This field provides a bitmap to signal that bus master 12 sources a process identifier register to the XRDC_MDAC logic. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Bus master 12 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 12 sources a process identifier register to the XRDC_MDAC logic. |
| 11<br>PIDP11 | Process identifier<br>Process identifier present from bus master 11. This field provides a bitmap to signal that bus master 11 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 11 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 11 sources a process identifier register to the XRDC_MDAC logic. |
| 10<br>PIDP10 | Process identifier<br>Process identifier present from bus master 10. This field provides a bitmap to signal that bus master 10 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 10 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 10 sources a process identifier register to the XRDC_MDAC logic. |
| 9<br>PIDP9 | Process identifier<br>Process identifier present from bus master 9. This field provides a bitmap to signal that bus master 9 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 9 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 9 sources a process identifier register to the XRDC_MDAC logic. |
| 8<br>PIDP8 | Process identifier<br>Process identifier present from bus master 8. This field provides a bitmap to signal that bus master 8 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 8 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 8 sources a process identifier register to the XRDC_MDAC logic. |
| 7<br>PIDP7 | Process identifier<br>Process identifier present from bus master 7. This field provides a bitmap to signal that bus master 7 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 7 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 7 sources a process identifier register to the XRDC_MDAC logic. |
| 6<br>PIDP6 | Process identifier<br>Process identifier present from bus master 6. This field provides a bitmap to signal that bus master 6 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 6 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 6 sources a process identifier register to the XRDC_MDAC logic. |
| 5<br>PIDP5 | Process identifier<br>Process identifier present from bus master 5. This field provides a bitmap to signal that bus master 5 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 5 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 5 sources a process identifier register to the XRDC_MDAC logic. |
| 4<br>PIDP4 | Process identifier |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Process identifier present from bus master 4. This field provides a bitmap to signal that bus master 4 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 4 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 4 sources a process identifier register to the XRDC_MDAC logic. |
| 3<br>PIDP3 | Process identifier<br>Process identifier present from bus master 3. This field provides a bitmap to signal that bus master 3 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 3 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 3 sources a process identifier register to the XRDC_MDAC logic. |
| 2<br>PIDP2 | Process identifier<br>Process identifier present from bus master 2. This field provides a bitmap to signal that bus master 2 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 2 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 2 sources a process identifier register to the XRDC_MDAC logic. |
| 1<br>PIDP1 | Process identifier<br>Process identifier present from bus master 1. This field provides a bitmap to signal that bus master 1 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 1 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 1 sources a process identifier register to the XRDC_MDAC logic. |
| 0<br>PIDP0 | Process identifier<br>Process identifier present from bus master 0. This field provides a bitmap to signal that bus master 0 sources a process identifier register to the XRDC_MDAC logic.<br>0b - Bus master 0 does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1b - Bus master 0 sources a process identifier register to the XRDC_MDAC logic. |

## 9.4.1.6 Hardware Configuration Register 3 (HWCFG3)

### 9.4.1.6.1 Offset

| Register | Offset |
|---|---|
| HWCFG3 | FCh |

#### 9.4.1.6.2 Function

This register contains information on the XRDC's hardware configuration. Specifically, it provides a bitmap signaling the presence of a process identifier (PID) register sourced from the given bus master. The HWCFG3 register is associated with bus masters [63-32], while the HWCFG2 register covers bus masters [31-0]. Attempted writes are error terminated.

Access: Privileged Read

#### 9.4.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDPn | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDPn | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 9.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PIDPn | Process identifier<br>Process identifier present from bus master n, where n = [63-32]. This field provides a bitmap to signal that bus master n (where n is defined as 32 + bit number) sources a process identifier register to the XRDC_MDAC logic.<br>0 Bus master "n" does not source a process identifier register. The XRDC_MDAC logic provides the needed PID for processor cores.<br>1 Bus master "n" sources a process identifier register to the XRDC_MDAC logic. |

### 9.4.1.7 Master Domain Assignment Configuration Register (MDAC FG0 - MDACFG36)

#### 9.4.1.7.1 Offset

| Register | Offset |
|---|---|
| MDACFG0 | 100h |
| MDACFG1 | 101h |
| MDACFG2 | 102h |
| MDACFG3 | 103h |
| MDACFG4 | 104h |
| MDACFG16 | 110h |
| MDACFG17 | 111h |
| MDACFG32 | 120h |
| MDACFG33 | 121h |
| MDACFG34 | 122h |
| MDACFG35 | 123h |
| MDACFG36 | 124h |

#### 9.4.1.7.2 Function

This register defines the number of implemented domain assignment registers for bus master m, where m+1 can specify from 1 to 64 bus masters. These registers are organized as a byte-sized data array and can be read using 8-, 16- or 32-bit accesses. An all-zero value (NCM = 0, NMDAR = 0) indicates a non-existent bus master. Attempted writes are error terminated.

Typically, processor bus masters have one or more domain assignment registers, while non-processor masters have a single domain assignment register.

Access: Privileged Read

#### 9.4.1.7.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NCM | 0 | | | NMDAR | | | |
| W | | | | | | | | |
| Reset | See Register reset values. | | | | | | | |

#### 9.4.1.7.4 Register reset values

| Register | Reset value |
|---|---|
| MDACFG0–MDACFG1 | 02h |
| MDACFG2–MDACFG4 | 81h |
| MDACFG5–MDACFG15 | Register not supported |
| MDACFG16–MDACFG32 | 02h |
| MDACFG18–MDACFG31 | Register not supported |
| MDACFG33–MDACFG34 | 81h |
| MDACFG35–MDACFG36 | 02h |
| MDACFG37–MDACFG62 | Register not supported |

#### 9.4.1.7.5 Fields

| Field | Function |
|---|---|
| 7<br>NCM | Non-CPU Master<br>This read-only field signals that bus master m is a non-CPU master. It specifies that the format of the associated MDA_Wr_m register defines a non-processor domain assignment. This field is zero for a non-existent bus master.<br>0b - Bus master is a processor.<br>1b - Bus master is a non-processor. |
| 6-4<br>— | Reserved |
| 3-0<br>NMDAR | Number of master domain assignment registers for bus master m<br>This read-only field specifies the number of registers associated with the master domain assignment register for a given bus master. The value is limited to the range [0-8], where zero indicates a non-existent bus master and non-zero values indicate the number of implemented registers associated with this MDAm. |

### 9.4.1.8 Memory Region Configuration Register (MRCFG0 - MRCFG1)

#### 9.4.1.8.1 Offset

For r = 0 to 1:

| Register | Offset |
|---|---|
| MRCFGr | 140h + (r × 1h) |

#### 9.4.1.8.2 Function

This read-only register defines the number of implemented memory region descriptors for each MRCr, where r+1 can specify up to 16 instances. These registers are organized as a byte-sized data array and can be read using 8-, 16- or 32-bit accesses. A zero value indicates a non-existent memory region controller instance. Attempted writes are error terminated.

Access: Privileged Read

#### 9.4.1.8.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | NMRGD | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

#### 9.4.1.8.4 Fields

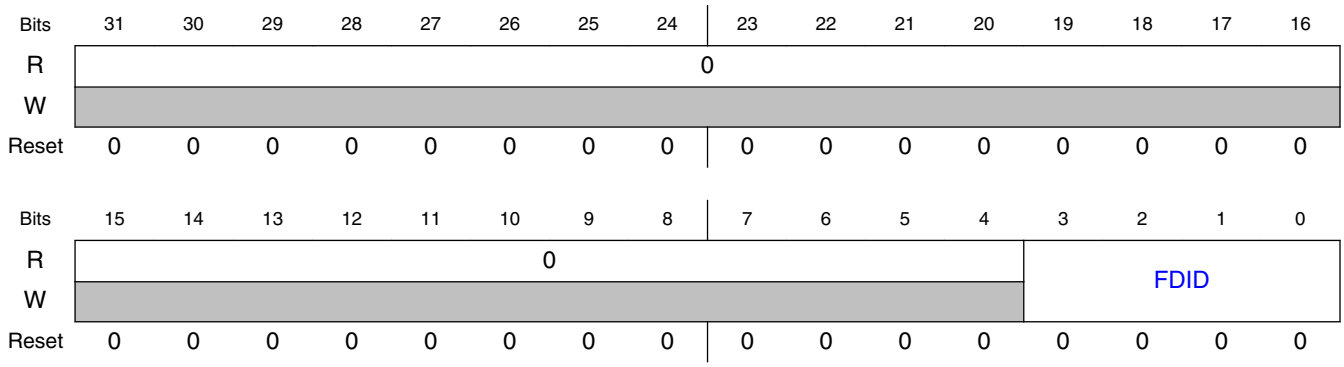
| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4-0<br>NMRGD | Number of memory region descriptors for memory region controller n<br>Number of memory region descriptors for MRCr. This field specifies the number of memory region descriptors associated with a given memory region controller instance. The value is limited to the range [0-16], where zero indicates a non-existent MRC instance and non-zero values indicate the number of implemented memory region descriptors [4, 8,12,16] associated with the XRDC_MRCr submodule. |

### 9.4.1.9 Fault Domain ID (FDID)

#### 9.4.1.9.1 Offset

| Register | Offset |
|---|---|
| FDID | 1FCh |

### 9.4.1.9.2 Diagram




### 9.4.1.9.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>FDID | Domain ID of Faulted Access<br>This field indicates the domainID of the fault that will be used to index the array of domain error information. The user queries the DERRLOCx registers to find the DomainID of the faulting access and must write this register with the domain ID before reading the Domain Error registers. |

## 9.4.1.10 Domain Error Location Register (DERRLOC0 - DERRLOC2)

### 9.4.1.10.1 Offset

For d = 0 to 2:

| Register | Offset |
|---|---|
| DERRLOCd | 200h + (d × 4h) |

### 9.4.1.10.2 Function

This array of read-only registers provide the instance number of the submodule where (an) access violation(s) occurred. These registers are organized as a word array, which is indexed by the faulting domain number, d. The two fields of this register provide a bitmap of instances associated with all submodules containing captured error information

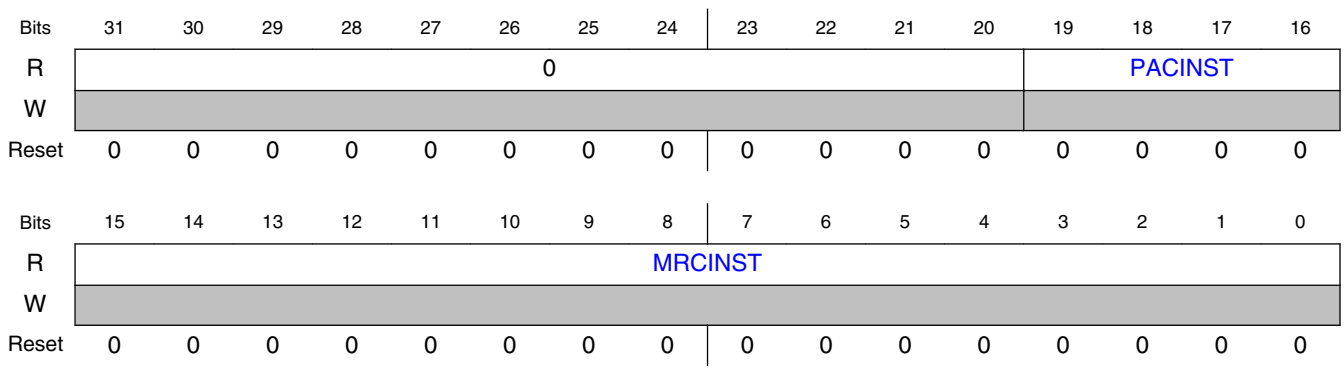
for that domain. These instance numbers are then used as indices into the DERR_W0_*i*, DERR_W1_*i*, and DERR_W3_*i* register arrays. See Domain error capture management for more details.

When an access violation is detected by either a memory region controller (MRC) or a peripheral access controller (PAC), address and attribute information of the offending access is captured. Using the faulting domainID number as the index, d, this array of read-only registers provide additional information signaling the instance number of the submodule where the access violation(s) occurred. Since the resulting exception handler needs the submodule instance to retrieve the captured address and attribute information from DERR_W0_*i* and DERR_W1_*i*, these registers provide the instance number details.

Attempted writes are error terminated.

Access: Privileged Read

### 9.4.1.10.3 Diagram




### 9.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>PACINST | PAC instance<br>This field is a bitmap indicating the presence of a detected access violation for domain d in the implemented instances of the PAC. The least-significant bit of this field (bit 16) corresponds to PAC instance 0. The most-significant bit of this field (bit 19) corresponds to PAC instance 3, and so on. Multiple bits can be set at any time indicating access violations for this domain have been detected across multiple instances of the PACs.<br>**NOTE:** When a memory slot controller (MSC) is present, it will appear here in the same way as a PAC instance.<br>For each bit in this field: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • 0 - The peripheral access controller has not detected an access violation or is not physically present.<br>• 1 - The peripheral access controller has detected one or more access violations for this domain. |
| 15-0<br>MRCINST | MRC instance<br>This field is a bitmap indicating the presence of a detected access violation for domain d in the implemented instances of the MRC. The least-significant bit of this field (bit 0) corresponds to MRC instance 0. The most-significant bit of this field (bit 3) corresponds to MRC instance 3, and so on. Multiple bits can be set at any time indicating access violations for this domain have been detected across multiple instances of the MRCs.<br>For each bit in this field:<br>• 0 - The memory region controller has not detected an access violation or is not physically present.<br>• 1 - The memory region controller has detected one or more access violations for this domain. |

## 9.4.1.11 Domain Error Word0 Register (DERR_W0_0 - DERR_W0_18)

### 9.4.1.11.1 Offset

| Register | Offset |
|---|---|
| DERR_W0_0 | 400h |
| DERR_W0_1 | 410h |
| DERR_W0_16 | 500h |
| DERR_W0_17 | 510h |
| DERR_W0_18 | 520h |

### 9.4.1.11.2 Function

This read-only register array provides the address of an access violation detected by either a memory region controller (MRC) or a peripheral access controller (PAC). These registers are organized as a word array, which is indexed by the violating submodule instance number. That is, the index, i, of this array is the instance number of the submodule that detected the access violation. The submodule instance numbers are provided by the DERRLOC registers. The memory-mapped error capture detail registers are organized as 20 sequential 16-byte entries. The first 16 register sets associated with the MRCs. The following register sets are associated with the PACs. Each 16-byte structure contains DERR_W0_i, DERR_W1_i, and DERR_W3_i.

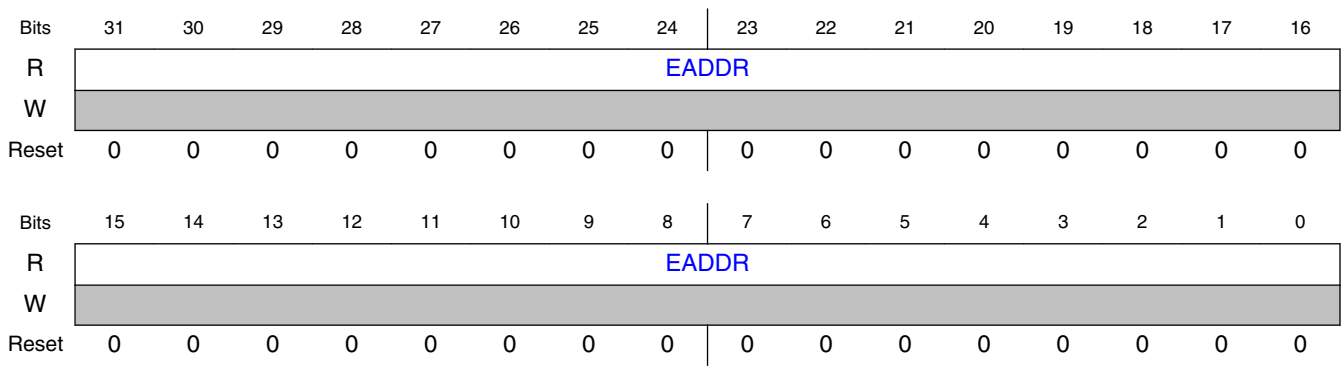
The error capture registers in the MRC and PAC submodules contain physical registers for each domain, but are organized in the DERR_W0_i and DERR_W1_i registers to provide the information for the requesting domain only. Thus, the access violation exception handler for each domain only has visibility into the captured error information for its own domain. See Domain error capture management for more details.

When an access violation is detected and the offending information captured, subsequent updates to this register are disabled until the required data pattern is written to the DERR_W3_i register. At that time, this register is cleared and re-enabled to capture the next access violation.

Attempted writes are error terminated as are attempted reads of an MRC or PAC instance that is not physically present.

Access: Privileged Read

### 9.4.1.11.3 Diagram




### 9.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31-0<br>EADDR | Error address<br>This is the access address that generated an access violation. |

## 9.4.1.12 Domain Error Word1 Register (DERR_W1_0 - DERR_W1_18)

### 9.4.1.12.1 Offset

| Register | Offset |
|---|---|
| DERR_W1_0 | 404h |
| DERR_W1_1 | 414h |
| DERR_W1_16 | 504h |
| DERR_W1_17 | 514h |
| DERR_W1_18 | 524h |

### 9.4.1.12.2 Function

This read-only register array provides the attributes of an access violation detected by either a memory region controller (MRC) or a peripheral access controller (PAC). These registers are organized as a word array, which is indexed by the violating submodule instance number. Refer to register Domain Error Location(DERRLOCd), Domain Error Word0 Register(DERR_W0_i), and Domain error capture management for more information.

Access: Privileged Read

### 9.4.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EST | | 0 | | | EPORT | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | ERW | EATR | | | 0 | | | | EDID | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.12.4 Fields

| Field | Function |
|---|---|
| 31-30<br>EST | Error state<br>This field signals the state of access violations for this domain in this instance of the memory region controller or peripheral access controller. Once an access violation has been detected and the faulting address and attribute information stored, subsequent errors are simply recorded as an overrun condition without any data captured. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | After retrieving the faulting address and attributes, the error capture mechanism must be rearmed by performing a write to DERR_W3_i.<br>00b - No access violation has been detected.<br>01b - No access violation has been detected.<br>10b - A single access violation has been detected.<br>11b - Multiple access violations for this domain have been detected by this submodule instance. Only the address and attribute information for the first error have been captured in DERR_W0_i and DERR_W1_i. |
| 29-27<br>— | Reserved |
| 26-24<br>EPORT | Error port<br>This field identifies the encoded port number of the MRC that detected the access violation. The MRC port number connection is device-specific. See the chip configuration details for more information. For access violations detected by the PAC, this field is cleared. |
| 23-12<br>— | Reserved |
| 11<br>ERW | Error read/write<br>This field signals whether the captured access violation occurred on a read or write reference.<br>0b - Read access<br>1b - Write access |
| 10-8<br>EATR | Error attributes<br>This field captures certain attributes of the access violation.<br>000b - Secure user mode, instruction fetch access.<br>001b - Secure user mode, data access.<br>010b - Secure privileged mode, instruction fetch access.<br>011b - Secure privileged mode, data access.<br>100b - Nonsecure user mode, instruction fetch access.<br>101b - Nonsecure user mode, data access.<br>110b - Nonsecure privileged mode, instruction fetch access.<br>111b - Nonsecure privileged mode, data access. |
| 7-4<br>— | Reserved |
| 3-0<br>EDID | Error domain identifier<br>This field captures the domain identifier of the access violation. |

## 9.4.1.13 Domain Error Word3 Register (DERR_W3_0 - DERR_W3_18)

### 9.4.1.13.1 Offset

| Register | Offset |
|---|---|
| DERR_W3_0 | 40Ch |
| DERR_W3_1 | 41Ch |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| DERR_W3_16 | 50Ch |
| DERR_W3_17 | 51Ch |
| DERR_W3_18 | 52Ch |

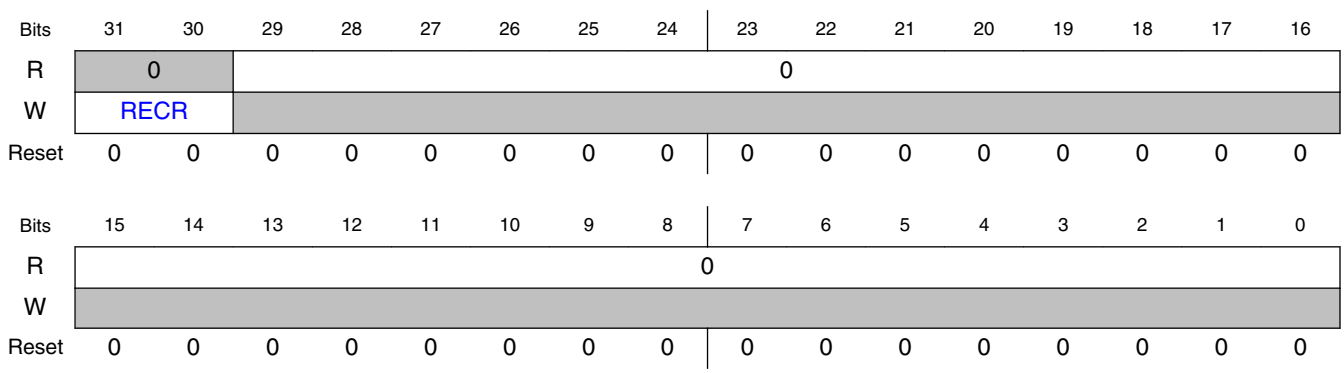
### 9.4.1.13.2 Function

This register is used to rearm the error capture logic and clear the DERR_W0_i and DERR_W1_i registers. After the domain access violation error details have been read, typically in an exception service routine, a 32-bit word write to this register is required to rearm the error capture logic.

A read of this location returns zeroes. Attempted reads of a memory region controller (MRC) or peripheral access controller (PAC) instance that is not physically present are error terminated.

See Domain error capture management for more details.

### 9.4.1.13.3 Diagram




### 9.4.1.13.4 Fields

| Field | Function |
|---|---|
| 31-30<br>RECR | Rearm Error Capture Registers<br>This 2-bit, write-only field controls the rearming of the domain error capture registers. Once an access violation has been detected with the faulting address and attribute information stored, subsequent errors are simply recorded as an overrun condition without any data captured.<br>Writing 01b to this field rearms the error capture mechanism and clears the DERR_W0_i and DERR_W1_i registers. A write of any value other than 01b has no effect. |
| 29-0<br>— | Reserved |

## 9.4.1.14 Process Identifier (PID0 - PID36)

### 9.4.1.14.1 Offset

| Register | Offset |
|---|---|
| PID0 | 700h |
| PID1 | 704h |
| PID16 | 740h |
| PID17 | 744h |
| PID32 | 780h |
| PID35 | 78Ch |
| PID36 | 790h |

### 9.4.1.14.2 Function

In the XRDC's access control definition, each processor has a corresponding process identifier (PID) which performs two important functions: the PID[5] bit defines the secure/nonsecure attribute associated with an executing task, and the entire value can be used to group tasks into different domains.

While certain processors include this register in their programming model definitions, others do not. This register is provided for those processors that do *not* include a PID register in their programming models. Secure privileged software saves and restores the PID as part of any context switch.

This data structure defines an array of 32-bit values, one per MDA module, that define the PID. Since this register resource is only applicable to processor cores, the data structure is typically sparsely populated. The HWCFG[2-3] registers provide a bitmap of the implemented PID*n* registers. This data structure is indexed using the corresponding MDA instance number.

For processors only supporting the 3-state access control model (PrivSecure, UserSecure, UserNonsecure), the nonsecure[n] output signal from the MDAC submodule is forced to zero while in privileged mode to enable precise state transitions between user and privileged modes. Specifically, the MDAC logic for bus master n forms the nonsecure attribute output signal as a function of 2 configuration bits: PIDm[TSM] and HWCFG{2,3}[n]. See Table 9-6.

When TSM is enabled, the output DID is a function of the mode:

TSM=1, mode = PrivSecure, DID_out= {2'b00, did_in[1:0]}

TSM=1, mode = UserSecure, if did_in=0,1, DID_out=2 else DID_out=did_in

TSM=1, mode = UserNonsecure, if did_in=0,1, DID_out=2 else DID_out=did_in

PrivNonsecure is an invalid mode when TSM=1.

Furthermore, when SP4SM is enabled, the MDAC doesn't use the MDA[DIDS,DID] fields. The MDAC tracks the access control mode and generates specific domainIDs. In PrivSec mode, the DID is forced to 0. In UserSecure and UserNonsecure modes, the DID is forced to 0 if DIDin is 0 or 1, and passes DIDin = DIDout for DIDs >=2.

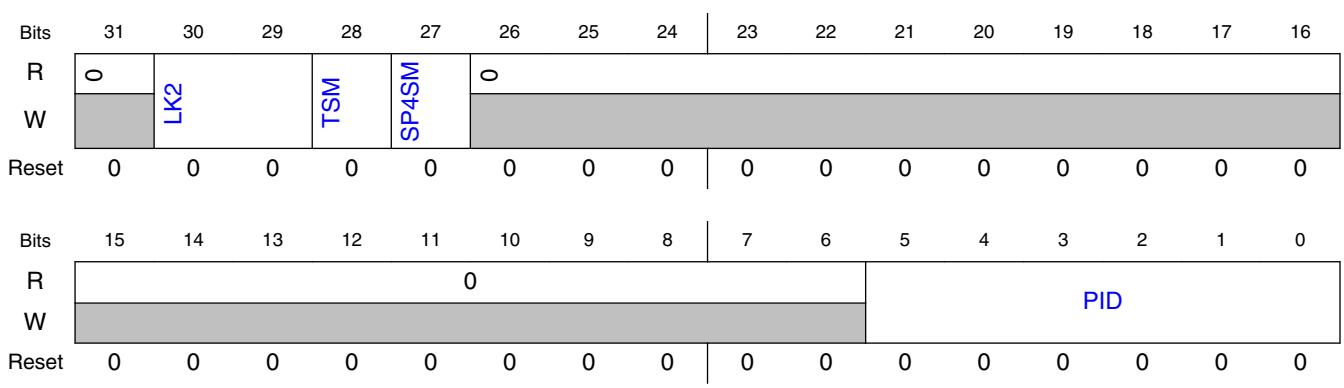
Depending on the operating clock domain of each MDAC instance, there may be optional information stored in the corresponding PIDm register to properly implement the LK2 = 2 functionality.

Reads of the PID*n* register return the contents of this register, or the PID*n* value directly sourced from a processor. For non-processor bus masters, this register does not exist and any attempted read is error terminated.

**Table 9-6. Generation of the nonsecure_out[n] Attribute**

| Configuration {PID*n*[TSM], HWCFG{2,3}[n]} | local_nonsecure[n] | nonsecure_out[n] |
|---|---|---|
| 00b | nonsecure_in[n] | local_nonsecure[n] |
| 01b | nonsecure_in[n] | local_nonsecure[n] & ~priv_in[n] |
| 10b | PID*n*[5] | local_nonsecure[n] & ~priv_in[n] |
| 11b | pid_in[n][5] | local_nonsecure[n] & ~priv_in[n] |

### 9.4.1.14.3 Diagram



### 9.4.1.14.4 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>—</td><td>Reserved</td></tr>
<tr><td>30-29<br>LK2</td><td>Lock<br>This field provides a locking mechanism that can be used to limit the ability to write the register. Once set, these bits individually remain asserted until the next reset.<br>If the processor sources its PIDm directly, that is, HWCFG{2-3}[PIDPm = 1], then privileged reads of this memory location return zero for this field.<br>00b - Register can be written by any secure privileged write.<br>01b - Register can be written by any secure privileged write.<br>10b - Register can only be written by a secure privileged write from bus master m.<br>11b - Register is locked (read-only) until the next reset.</td></tr>
<tr><td>28<br>TSM</td><td>Three-state model<br>If asserted, this bit indicates the associated processor core only supports the 3-state access control model. This indicator is “sticky”, that is, once set, it remains set until the next reset. It can only be set in secure mode.<br>For processors only supporting the 3-state access control model, this field must be set before loading any nonsecure value into the PID.<br>See Table 9-6 for the specification of the privileged and nonsecure attributes generated by MDAC.</td></tr>
<tr><td>27<br>SP4SM</td><td>Special 4-state model<br>This indicator is “sticky”, that is, once set, it remains set until the next reset. It can only be set in secure mode.<br>If asserted, a Mealy finite state machine in each processor MDAC tracks the current 2-bit access mode and generates specific domainIDs for specific modes.<br>Domain ID is set as following:<br><table><tr><th>State</th><th>Domain ID</th><th>Description</th></tr><tr><td>PrivSecure</td><td>0x0</td><td>Hardware-enforced association between PrivSecure mode & DID=0x0</td></tr><tr><td>PrivNonsecure</td><td>0x1</td><td>Hardware-enforced association between PrivNonSecure mode & DID=0x1</td></tr><tr><td>UserSecure</td><td>>= 0x2</td><td>Software-defined DID configured via the core MDAn registers (MDAC hardware gaurantees the DID is greater than or equal to 2</td></tr><tr><td>UserNonsecure</td><td>>= 0x2</td><td>Software-defined DID configured via the core MDAn registers (MDAC hardware gaurantees the DID is greater than or equal to 2</td></tr></table><br>Refer to Special 4-State Model for more details.</td></tr>
<tr><td>26-6<br>—</td><td>Reserved</td></tr>
</table>

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5-0<br>PID | Process identifier<br>The PIDm[5] bit determines the secure (0) or nonsecure (1) attribute for transactions associated with the corresponding processor.<br>If the processor sources its PID*n* directly, then secure privileged reads of this memory location return the processor register for this field. |

## 9.4.1.15 Master Domain Assignment (MDA_W0_0_DFMT0 - MDA_W1_36_DFMT0)

### 9.4.1.15.1 Offset

| Register | Offset |
|---|---|
| MDA_W0_0_DFMT0 | 800h |
| MDA_W1_0_DFMT0 | 804h |
| MDA_W0_1_DFMT0 | 820h |
| MDA_W1_1_DFMT0 | 824h |
| MDA_W0_16_DFMT0 | A00h |
| MDA_W1_16_DFMT0 | A04h |
| MDA_W0_17_DFMT0 | A20h |
| MDA_W1_17_DFMT0 | A24h |
| MDA_W0_32_DFMT0 | C00h |
| MDA_W1_32_DFMT0 | C04h |
| MDA_W0_35_DFMT0 | C60h |
| MDA_W1_35_DFMT0 | C64h |
| MDA_W0_36_DFMT0 | C80h |
| MDA_W1_36_DFMT0 | C84h |

### 9.4.1.15.2 Function

The MDA_Wr_m registers provide a 2-dimensional data structure for assigning bus masters to domains. The number of implemented registers is defined by MDACFGm[NMDAR]. This per-master domain assignment is then repeated for each bus master (MDAm). Thus, m specifies the master number and r refers to the specific MDA register for a given bus master.

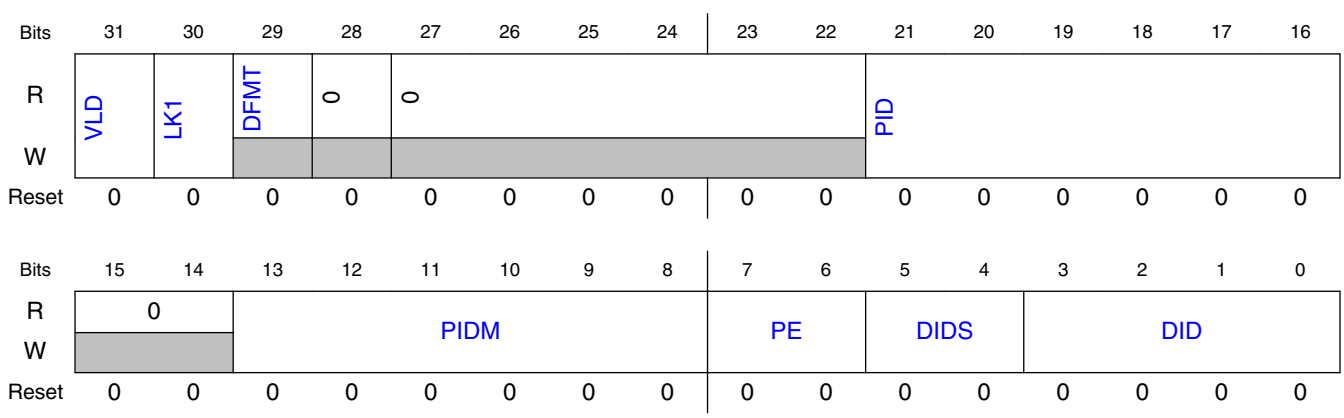
Each Wr within the MDAm structure is a word-sized definition; there are two formats supported, one for processor cores and another for non-processors. Processor masters typically support one or more Wr domain definitions, while non-processor masters support a single Wr.

The MDAC submodule is responsible for the generation of domain identifiers for every transaction from every bus master. If there is a single Wr for a given master, then the specified domain identifier is used directly. If there are multiple Wr values for a given master, then the MDAC evaluates the conditional terms to determine a “hit”. For all Wr hits, their corresponding domain identifiers are simply logically summed together (boolean OR). Use cases are typically expected to *hit in a single Wr* for a processor master. Special care is needed if *none* of the conditional terms hit in any Wr evaluation; for this case, the generated DID = 0 and software needs to be aware of any potential access rights granted for this DID.

Each MDA_Wr_m register has one of two programming models depending on the state of the domain format field, DFMT. The model described in this section is for DFMT = 0. This definition allows three different specifications of the DID for processors. The DFMT = 1 model is described in the following section, Master Domain Assignment (MDA_Wr_m_DFMT1).

Access: Privileged Read/write

### 9.4.1.15.3 Diagram




### 9.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This field indicates the domain assignment is valid. It is further qualified by CR[GVLDM] = 1. If CR[GVLDM] is cleared, the DID output is defined by the SoC-specific *default DID* value. See the XRDC chip configuration section for more details. If both MDA_Wr_m[VLD] and CR[GVLDM] are asserted, the DID output is defined by the remaining contents of this register.<br>0b - The Wr domain assignment is invalid.<br>1b - The Wr domain assignment is valid. |
| 30<br>LK1 | 1-bit Lock<br>This field provides a locking mechanism that can be used to limit the ability to write the register. Once set, this bit remains asserted until the next reset.<br>0b - Register can be written by any secure privileged write.<br>1b - Register is locked (read-only) until the next reset. |
| 29<br>DFMT | Domain format<br>Identifies this register's domain assignment.<br>**NOTE:** This bitfield access is ROZ<br>0b - Processor-core domain assignment<br>1b - Non-processor domain assignment |
| 28<br>— | Reserved |
| 27-22<br>— | Reserved |
| 21-16<br>PID | Process Identifier<br>This field specifies that the process identifier is to be combined with the PIDM field and included in the domain hit determination. The optional inclusion of the PID and PIDM is controlled by the MDA_Wr_m[PE] field. |
| 15-14<br>— | Reserved |
| 13-8<br>PIDM | Process Identifier Mask<br>This field provides a masking capability so that multiple process identifiers can be included as part of the domain hit determination. If a bit in the PIDM is set, then the corresponding bit of the PID is ignored in the comparison. The optional inclusion of the PID and PIDM is controlled by the MDA_Wr_m[PE] field. |
| 7-6<br>PE | Process identifier enable<br>This field controls the optional inclusion of the PID, qualified by PIDM, into the domain hit evaluation. It provides the ability to include inclusive or exclusive sets of masked PID values.<br>00b - No process identifier is included in the domain hit evaluation.<br>01b - No process identifier is included in the domain hit evaluation.<br>10b - The process identifier is included in the domain hit evaluation as defined by the expression: `partial_domain_hit = (PE == 2) && ((PID & ~PIDM) == (XRDC_PIDn[PID] & ~PIDM))`<br>11b - The process identifier is included in the domain hit evaluation as defined by the expression: `partial_domain_hit = (PE == 3) && ~((PID & ~PIDM) == (XRDC_PIDn[PID] & ~PIDM))` |
| 5-4<br>DIDS | DID Select<br>This field selects the source of the domain identifier.<br>00b - Use MDAm[3:0] as the domain identifier.<br>01b - Use the input DID as the domain identifier.<br>10b - Use MDAm[3:2] concatenated with the low-order 2 bits of the input DID (DID_in[1:0]) as the domain identifier. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 11b - Reserved for future use. |
| 3-0<br>DID | Domain identifier<br>When the DIDB=1., this 4-bit field is the domain ID attribute that sent on accesses from the bus master connected to the MDAC. |

## 9.4.1.16 Master Domain Assignment (MDA_W0_2_DFMT1 - MDA_W0_34_DFMT1)

### 9.4.1.16.1 Offset

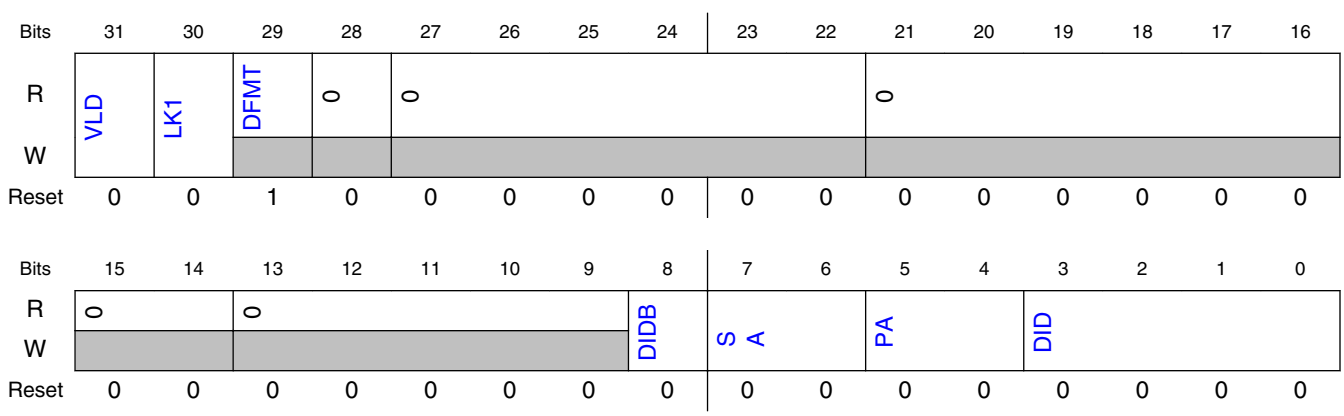
| Register | Offset |
|---|---|
| MDA_W0_2_DFMT1 | 840h |
| MDA_W0_3_DFMT1 | 860h |
| MDA_W0_4_DFMT1 | 880h |
| MDA_W0_33_DFMT1 | C20h |
| MDA_W0_34_DFMT1 | C40h |

### 9.4.1.16.2 Function

This register is identical to Master Domain Assignment (MDA_W0_0_DFMT0 - MDA_W1_36_DFMT0) except that the domain format field, DFMT, is 1. This format supports two different specifications of the DID for non-core bus masters.

Access: Privileged Read/write

### 9.4.1.16.3 Diagram



### 9.4.1.16.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>This field indicates the domain assignment is valid. It is further qualified by CR[GVLDM] = 1. If CR[GVLDM] is cleared, the DID output is defined by the SoC-specific *default DID* value. See the XRDC chip configuration section for more details. If both MDA_Wr_m[VLD] and CR[GVLDM] are asserted, the DID output is defined by the remaining contents of this register.<br>0b - The Wr domain assignment is invalid.<br>1b - The Wr domain assignment is valid. |
| 30<br>LK1 | 1-bit Lock<br>This field provides a locking mechanism that can be used to limit the ability to write the register. Once set, this bit remains asserted until the next reset.<br>0b - Register can be written by any secure privileged write.<br>1b - Register is locked (read-only) until the next reset. |
| 29<br>DFMT | Domain format<br>Identifies this register's domain assignment.<br>**NOTE:** This bitfield access is ROO<br>0b - Processor-core domain assignment<br>1b - Non-processor domain assignment |
| 28<br>— | Reserved |
| 27-22<br>— | Reserved |
| 21-16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13-9<br>— | Reserved |
| 8<br>DIDB | DID Bypass<br>If asserted, this bit enables the bypassing of an input DID value as the domain identifier for this non-processor bus master. This capability allows non-processor bus masters, for example, a DMA to masquerade as a processor.<br>Once set, this field is “sticky” and remains set until the next reset. |

| MDAC | DID Input |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | DMA master ID |
| 3 | 2 |
| 4 | 2 |
| 32 | 1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | <table><tr><th>MDAC</th><th>DID Input</th></tr><tr><td>33</td><td>DMA master ID</td></tr><tr><td>34</td><td>CAU3 master ID</td></tr></table><br>0b - Use MDAn[3:0] as the domain identifier.<br>1b - Use the DID input as the domain identifier. |
| 7-6<br>SA | Secure attribute<br>This field defines the secure/nonsecure attribute for non-processor cores.<br>**NOTE:** The bus master's input secure/nonsecure attribute is used if SA = 1X, or this VLD = 0.<br>00b - Force the bus attribute for this master to secure.<br>01b - Force the bus attribute for this master to nonsecure.<br>10b - Use the bus master's secure/nonsecure attribute directly.<br>11b - Use the bus master's secure/nonsecure attribute directly. |
| 5-4<br>PA | Privileged attribute<br>This field defines the privileged/user attribute for non-processor cores.<br>**NOTE:** The bus master's input privileged/user attribute is used if PA = 1X, or this VLD = 0.<br>00b - Force the bus attribute for this master to user.<br>01b - Force the bus attribute for this master to privileged.<br>10b - Use the bus master's privileged/user attribute directly.<br>11b - Use the bus master's privileged/user attribute directly. |
| 3-0<br>DID | Domain identifier<br>When the DIDB=1., this 4-bit field is the domain ID attribute that sent on accesses from the bus master connected to the MDAC. |

## 9.4.1.17 Peripheral Domain Access Control (PDAC_W0_0_1 - PDAC_W0_2_32)

### 9.4.1.17.1 Offset

| Register | Offset |
|---|---|
| PDAC_W0_0_1 | 1008h |
| PDAC_W0_0_3 | 1018h |
| PDAC_W0_0_4 | 1020h |
| PDAC_W0_0_8 | 1040h |
| PDAC_W0_0_9 | 1048h |
| PDAC_W0_0_12 | 1060h |
| PDAC_W0_0_15 | 1078h |
| PDAC_W0_0_20 | 10A0h |
| PDAC_W0_0_21 | 10A8h |
| PDAC_W0_0_22 | 10B0h |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| PDAC_W0_0_23 | 10B8h |
| PDAC_W0_0_27 | 10D8h |
| PDAC_W0_0_32 | 1100h |
| PDAC_W0_0_33 | 1108h |
| PDAC_W0_0_34 | 1110h |
| PDAC_W0_0_35 | 1118h |
| PDAC_W0_0_36 | 1120h |
| PDAC_W0_0_37 | 1128h |
| PDAC_W0_0_38 | 1130h |
| PDAC_W0_0_39 | 1138h |
| PDAC_W0_0_40 | 1140h |
| PDAC_W0_0_41 | 1148h |
| PDAC_W0_0_42 | 1150h |
| PDAC_W0_0_43 | 1158h |
| PDAC_W0_0_44 | 1160h |
| PDAC_W0_0_45 | 1168h |
| PDAC_W0_0_46 | 1170h |
| PDAC_W0_0_47 | 1178h |
| PDAC_W0_0_48 | 1180h |
| PDAC_W0_0_49 | 1188h |
| PDAC_W0_0_50 | 1190h |
| PDAC_W0_0_51 | 1198h |
| PDAC_W0_0_52 | 11A0h |
| PDAC_W0_0_53 | 11A8h |
| PDAC_W0_0_54 | 11B0h |
| PDAC_W0_0_55 | 11B8h |
| PDAC_W0_0_56 | 11C0h |
| PDAC_W0_0_57 | 11C8h |
| PDAC_W0_0_58 | 11D0h |
| PDAC_W0_0_59 | 11D8h |
| PDAC_W0_0_60 | 11E0h |
| PDAC_W0_0_61 | 11E8h |
| PDAC_W0_0_62 | 11F0h |
| PDAC_W0_0_63 | 11F8h |
| PDAC_W0_0_64 | 1200h |
| PDAC_W0_0_65 | 1208h |
| PDAC_W0_0_66 | 1210h |
| PDAC_W0_0_67 | 1218h |
| PDAC_W0_0_68 | 1220h |
| PDAC_W0_0_69 | 1228h |
| PDAC_W0_0_70 | 1230h |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| PDAC_W0_0_71 | 1238h |
| PDAC_W0_0_72 | 1240h |
| PDAC_W0_0_73 | 1248h |
| PDAC_W0_0_74 | 1250h |
| PDAC_W0_0_75 | 1258h |
| PDAC_W0_0_76 | 1260h |
| PDAC_W0_0_77 | 1268h |
| PDAC_W0_0_78 | 1270h |
| PDAC_W0_0_79 | 1278h |
| PDAC_W0_0_125 | 13E8h |
| PDAC_W0_0_126 | 13F0h |
| PDAC_W0_0_127 | 13F8h |
| PDAC_W0_1_8 | 1440h |
| PDAC_W0_1_9 | 1448h |
| PDAC_W0_1_15 | 1478h |
| PDAC_W0_1_27 | 14D8h |
| PDAC_W0_1_31 | 14F8h |
| PDAC_W0_1_32 | 1500h |
| PDAC_W0_1_33 | 1508h |
| PDAC_W0_1_34 | 1510h |
| PDAC_W0_1_35 | 1518h |
| PDAC_W0_1_36 | 1520h |
| PDAC_W0_1_37 | 1528h |
| PDAC_W0_1_38 | 1530h |
| PDAC_W0_1_39 | 1538h |
| PDAC_W0_1_40 | 1540h |
| PDAC_W0_1_41 | 1548h |
| PDAC_W0_1_42 | 1550h |
| PDAC_W0_1_43 | 1558h |
| PDAC_W0_1_44 | 1560h |
| PDAC_W0_1_45 | 1568h |
| PDAC_W0_1_46 | 1570h |
| PDAC_W0_1_47 | 1578h |
| PDAC_W0_1_48 | 1580h |
| PDAC_W0_1_49 | 1588h |
| PDAC_W0_1_50 | 1590h |
| PDAC_W0_1_51 | 1598h |
| PDAC_W0_1_52 | 15A0h |
| PDAC_W0_1_53 | 15A8h |
| PDAC_W0_1_54 | 15B0h |
| PDAC_W0_1_55 | 15B8h |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| PDAC_W0_1_56 | 15C0h |
| PDAC_W0_2_16 | 1880h |
| PDAC_W0_2_32 | 1900h |

### 9.4.1.17.2 Function

The PDAC_Wr_c_s registers provide a 3-dimensional data structure for defining access control policies per domain for each implemented slave peripheral. Refer to the Access control policies table below . There are 2 word-sized registers assigned for each peripheral "address slot" (Wr).

Each Wr within the PDAC*s* structure is a word-sized definition; a total of two words (64 bits) is needed to define the access control policies for all the domains.

These registers also allow an exclusive access lock to be optionally included in the access control evaluation for peripherals shared by multiple domains. If enabled, the state of the lock dynamically modifies the access control policies so that only the domain "owning" it has write permission into the peripheral. The write permissions for all other domains are revoked based on the lock state. If the lock is disabled, the DxACP fields are evaluated normally.

The PAC submodule uses the peripheral access address to select the appropriate PDAC*s* register, and then uses the domain identifier associated with the access to select the appropriate DxACP field. The selected DxACP field is then evaluated against the attributes associated with the memory transaction to determine the validity of the access. See DxACP evaluation for more details. If allowed, the access proceeds, else an access violation is signaled, the transfer error terminated and the appropriate address and attribute information captured in the corresponding error detail registers.

Access: Privileged Read/write

**Table 9-7. Access control policies**

| Policy | PrivSecure | PrivNonsecure | UserSecure | UserNonsecure |
|---|---|---|---|---|
| 111 | r, w | r, w | r, w | r, w |
| 110 | r, w | r, w | r, w | none |
| 101 | r, w | r, w | r | r |
| 100 | r, w | r, w | none | none |
| 011 | r, w | r | none | none |
| 010 | r, w | none | none | none |
| 001 | r | none | none | none |
| 000 | none | none | none | none |

Access: Privileged Read/write

### 9.4.1.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | | EALO | | | | 0 | | | 0 | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | D2ACP | | | D1ACP | | | D0ACP | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.17.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29-28<br>— | Reserved |
| 27-24<br>EALO | Excessive Access Lock Owner<br>When EAL=11, this 4-bit read only field indicates which domain owns the exclusive access lock. See section Exclusive Access Lock for details on usage of the exclusive access lock. This field is reserved for devices that do not include the exclusive access lock. |
| 23-21<br>— | Reserved. |
| 20-18<br>— | Reserved. |
| 17-15<br>— | Reserved. |
| 14-12<br>— | Reserved. |
| 11-9<br>— | Reserved. |
| 8-6<br>D2ACP | Domain 2 access control policy<br>Refer to Domain 0 access control policy. |
| 5-3 | Domain 1 access control policy |

*Table continues on the next page...*

| Field | Function |
|---|---|
| D1ACP | Refer to Domain 0 access control policy. |
| 2-0<br>D0ACP | Domain 0 access control policy<br>This field defines the access control rights for the selected domain. The encodings specify read and write access capabilities based on the 4 operating states. These fields are only implemented for supported domainIDs; unsupported domainIDs are not implemented and default to a zero value providing no access rights.<br>Refer to the table Table 9-8 |

## 9.4.1.18 Peripheral Domain Access Control (PDAC_W1_0_1 - PDAC_W1_2_32)

### 9.4.1.18.1 Offset

| Register | Offset |
|---|---|
| PDAC_W1_0_1 | 100Ch |
| PDAC_W1_0_3 | 101Ch |
| PDAC_W1_0_4 | 1024h |
| PDAC_W1_0_8 | 1044h |
| PDAC_W1_0_9 | 104Ch |
| PDAC_W1_0_12 | 1064h |
| PDAC_W1_0_15 | 107Ch |
| PDAC_W1_0_20 | 10A4h |
| PDAC_W1_0_21 | 10ACh |
| PDAC_W1_0_22 | 10B4h |
| PDAC_W1_0_23 | 10BCh |
| PDAC_W1_0_27 | 10DCh |
| PDAC_W1_0_32 | 1104h |
| PDAC_W1_0_33 | 110Ch |
| PDAC_W1_0_34 | 1114h |
| PDAC_W1_0_35 | 111Ch |
| PDAC_W1_0_36 | 1124h |
| PDAC_W1_0_37 | 112Ch |
| PDAC_W1_0_38 | 1134h |
| PDAC_W1_0_39 | 113Ch |
| PDAC_W1_0_40 | 1144h |
| PDAC_W1_0_41 | 114Ch |
| PDAC_W1_0_42 | 1154h |
| PDAC_W1_0_43 | 115Ch |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| PDAC_W1_0_44 | 1164h |
| PDAC_W1_0_45 | 116Ch |
| PDAC_W1_0_46 | 1174h |
| PDAC_W1_0_47 | 117Ch |
| PDAC_W1_0_48 | 1184h |
| PDAC_W1_0_49 | 118Ch |
| PDAC_W1_0_50 | 1194h |
| PDAC_W1_0_51 | 119Ch |
| PDAC_W1_0_52 | 11A4h |
| PDAC_W1_0_53 | 11ACh |
| PDAC_W1_0_54 | 11B4h |
| PDAC_W1_0_55 | 11BCh |
| PDAC_W1_0_56 | 11C4h |
| PDAC_W1_0_57 | 11CCh |
| PDAC_W1_0_58 | 11D4h |
| PDAC_W1_0_59 | 11DCh |
| PDAC_W1_0_60 | 11E4h |
| PDAC_W1_0_61 | 11ECh |
| PDAC_W1_0_62 | 11F4h |
| PDAC_W1_0_63 | 11FCh |
| PDAC_W1_0_64 | 1204h |
| PDAC_W1_0_65 | 120Ch |
| PDAC_W1_0_66 | 1214h |
| PDAC_W1_0_67 | 121Ch |
| PDAC_W1_0_68 | 1224h |
| PDAC_W1_0_69 | 122Ch |
| PDAC_W1_0_70 | 1234h |
| PDAC_W1_0_71 | 123Ch |
| PDAC_W1_0_72 | 1244h |
| PDAC_W1_0_73 | 124Ch |
| PDAC_W1_0_74 | 1254h |
| PDAC_W1_0_75 | 125Ch |
| PDAC_W1_0_76 | 1264h |
| PDAC_W1_0_77 | 126Ch |
| PDAC_W1_0_78 | 1274h |
| PDAC_W1_0_79 | 127Ch |
| PDAC_W1_0_125 | 13ECh |
| PDAC_W1_0_126 | 13F4h |
| PDAC_W1_0_127 | 13FCh |
| PDAC_W1_1_8 | 1444h |
| PDAC_W1_1_9 | 144Ch |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| PDAC_W1_1_15 | 147Ch |
| PDAC_W1_1_27 | 14DCh |
| PDAC_W1_1_31 | 14FCh |
| PDAC_W1_1_32 | 1504h |
| PDAC_W1_1_33 | 150Ch |
| PDAC_W1_1_34 | 1514h |
| PDAC_W1_1_35 | 151Ch |
| PDAC_W1_1_36 | 1524h |
| PDAC_W1_1_37 | 152Ch |
| PDAC_W1_1_38 | 1534h |
| PDAC_W1_1_39 | 153Ch |
| PDAC_W1_1_40 | 1544h |
| PDAC_W1_1_41 | 154Ch |
| PDAC_W1_1_42 | 1554h |
| PDAC_W1_1_43 | 155Ch |
| PDAC_W1_1_44 | 1564h |
| PDAC_W1_1_45 | 156Ch |
| PDAC_W1_1_46 | 1574h |
| PDAC_W1_1_47 | 157Ch |
| PDAC_W1_1_48 | 1584h |
| PDAC_W1_1_49 | 158Ch |
| PDAC_W1_1_50 | 1594h |
| PDAC_W1_1_51 | 159Ch |
| PDAC_W1_1_52 | 15A4h |
| PDAC_W1_1_53 | 15ACh |
| PDAC_W1_1_54 | 15B4h |
| PDAC_W1_1_55 | 15BCh |
| PDAC_W1_1_56 | 15C4h |
| PDAC_W1_2_16 | 1884h |
| PDAC_W1_2_32 | 1904h |

### 9.4.1.18.2 Function

Refer to the "Peripheral Domain Access Control (PDAC_W0_c_s)" section for the description.

Access: Privileged Read/write

### 9.4.1.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VLD | LK2 | | 0 | | | EAL | | 0 | | | 0 | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | | 0 | | | 0 | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.18.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Valid. This field indicates the peripheral domain access control definition is valid. It is further qualified by CR[GVLDP] = 1. If either CR[GVLDP] or this VLD flag is cleared, all accesses to the peripheral are allowed. Any write to PDAC_W0_s clears the PDAC_W1_s[VLD] indicator so a coherent register state can be supported.<br>0b - The PDACs assignment is invalid.<br>1b - The PDACs assignment is valid. |
| 30-29<br>LK2 | Lock<br>This 2-bit field provides a mechanism to limit writes to the PDAC*s* register to protect its contents. Once set, these bits individually remain asserted until the next reset.<br>**NOTE:** This 2-bit lock does not apply to the EAL field.<br>00b - Entire PDACs can be written.<br>01b - Entire PDACs can be written.<br>10b - Domain x can only update the DxACP field and the LK2 field; no other PDACs fields can be written.<br>11b - PDACs is locked (read-only) until the next reset. |
| 28-26<br>— | Reserved |
| 25-24<br>EAL | Exclusive Access Lock<br>This 2-bit field provides a mechanism to dynamically modify the DxACP permission evaluation such that only one domain has access permissions at a time. This field is written independently of the rest of the W1 register. Writes with a value of 00 do not affect the exclusive access lock and are used to update other fields of the W1 register. Writes with a non-zero value do not update other fields of the W1 register. Neither the global configuration lock nor the descriptor lock apply to this field. See section Exclusive Access Lock for details on usage of the exclusive access lock. This field is reserved for devices that do not include the exclusive access lock.<br>00b - Lock disabled<br>01b - Lock disabled until next reset<br>10b - Lock enabled, lock state = available<br>11b - Lock enabled, lock state = not available |
| 23-21 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 20-18<br>— | Reserved. |
| 17-15<br>— | Reserved. |
| 14-12<br>— | Reserved. |
| 11-9<br>— | Reserved. |
| 8-6<br>— | Reserved. |
| 5-3<br>— | Reserved. |
| 2-0<br>— | Reserved. |

## 9.4.1.19 Memory Region Descriptor (MRGD_W0_0_0 - MRGD_W0_1_7)

### 9.4.1.19.1 Offset

For m = 0 to 7; n = 0 to 1:

| Register | Offset |
|---|---|
| MRGD_W0_n_m | 2000h + (n × 200h) + (m × 20h) |

### 9.4.1.19.2 Function

The MRGD_W*o*_*n*_*m* registers provide a 3-dimensional data structure for defining access control policies (Table 9-8) per domain for each supported memory region. There are five word-size registers (W*o*) defined in each memory region. There are up to 16 memory region controllers (MRCs), *n*, each supporting up to 16 memory region descriptors (MRGD), *m*. The number of MRCs is defined by HWCFG0[NMRC].

Each instance of a MRC submodule can monitor up to 8 system buses concurrently.

**NOTE**

An AXI port counts as two system buses since the read address and write address channels operate independently.

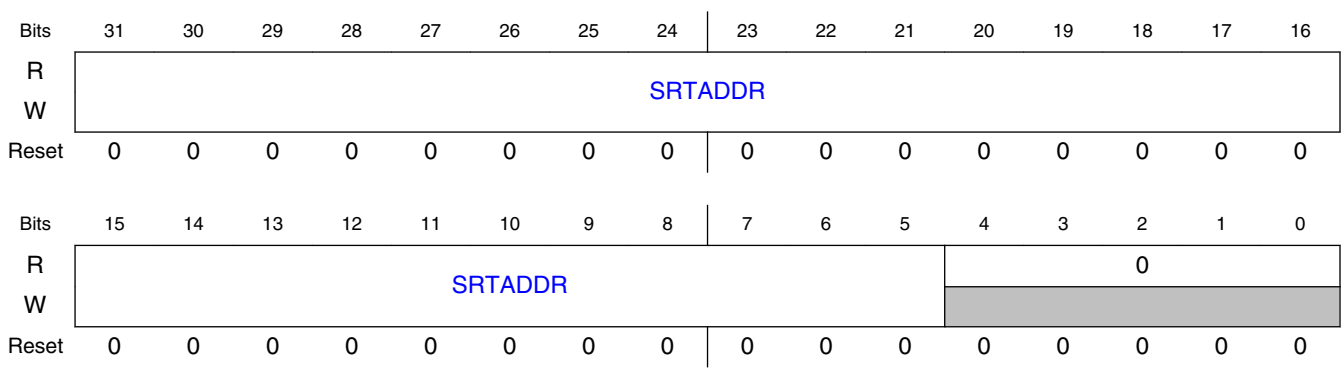
Each W*o* within the MRGD*m* structure is a word-sized definition; a total of four words (128 bits, W0-W3) are needed to completely define the memory region descriptor.

XRDC uses a base address plus a size (0-modulo-size) to specify memory regions. This format provides addressing capabilities that are equivalent to those provided in the ARM Cortex-M processor core memory protection unit.

The region descriptor also allows an exclusive access lock to be optionally included in the access control evaluation for memory regions shared by multiple domains. If enabled, the state of the lock dynamically modifies the access control policies so that only the domain "owning" it has write permission into the region. The write permissions for all other domains are revoked based on the lock state. If the semaphore is not owned by any domain, the D*x*ACP fields are evaluated normally.

The MRC submodule evaluates multiple system bus access addresses versus the programmed address ranges to determine a region hit. Once a region hit has been determined, the MRC then uses the domain identifier associated with the access to select the appropriate D*x*ACP field from the corresponding MRGDn. The selected D*x*ACP field is then evaluated against the attributes associated with the memory transaction to determine the validity of the access. See Memory region descriptor hit determination and Memory region DxACP evaluation for more details. If allowed, the access proceeds, else an access violation is signaled, the transfer error terminated and the appropriate address and attribute information captured in the corresponding error detail registers.

### 9.4.1.19.3 Diagram



#### 9.4.1.19.4 Fields

| Field | Function |
|---|---|
| 31-5<br>SRTADDR | Start Address<br>This field defines the most significant bits of the 0-modulo-32 byte start address of the memory region. The minimum region size is 32 bytes. |
| 4-0<br>— | Reserved |

### 9.4.1.20 Memory Region Descriptor (MRGD_W1_0_0 - MRGD_W1_1_7)

#### 9.4.1.20.1 Offset

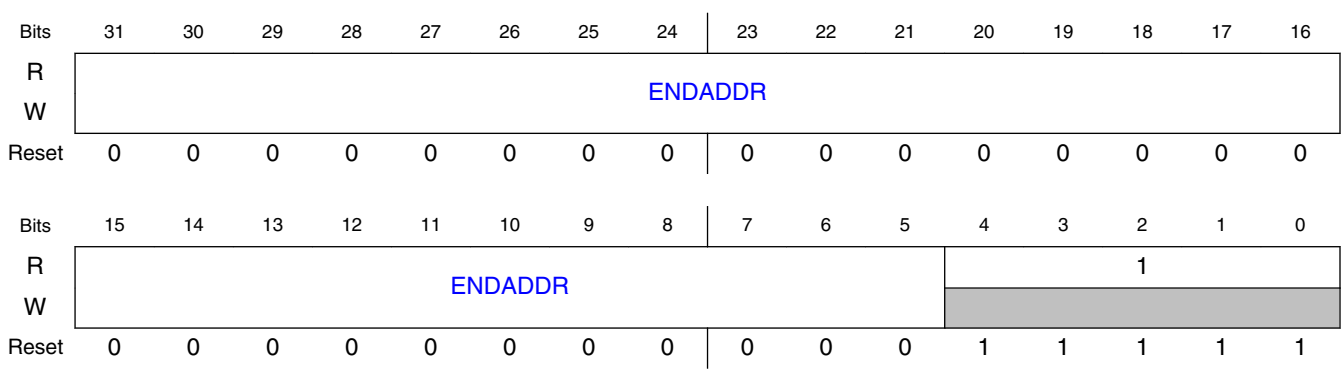
For m = 0 to 7; n = 0 to 1:

| Register | Offset |
|---|---|
| MRGD_W1_n_m | 2004h + (n × 200h) + (m × 20h) |

#### 9.4.1.20.2 Function

Refer to Memory Region Descriptor (MRGD_W0_0_0 - MRGD_W0_1_7) for description.

#### 9.4.1.20.3 Diagram



#### 9.4.1.20.4 Fields

| Field | Function |
|---|---|
| 31-5<br>ENDADDR | End Address<br>This field defines the most significant bits of the 31-modulo-32 byte end address of the memory region. |
| 4-0<br>— | Reserved |

### 9.4.1.21 Memory Region Descriptor (MRGD_W2_0_0 - MRGD_W2_1_7)

#### 9.4.1.21.1 Offset

For m = 0 to 7; n = 0 to 1:

| Register | Offset |
|---|---|
| MRGD_W2_n_m | 2008h + (n × 200h) + (m × 20h) |

#### 9.4.1.21.2 Function

Refer to Memory Region Descriptor (MRGD_W0_0_0 - MRGD_W0_1_7) for description.

**Table 9-8. Policy**

| | PS > PN > US > UN | PS > PN > US > UN |
|---|---|---|
| DxSEL | CodeRegion = 0 | CodeRegion = 1 |
| 111 | 110_110_110_110 = 0xDB6 | 110_110_111_111 = 0xDBF |
| 110 | 110_110_110_000 = 0xDB0 | 110_110_111_000 = 0xDB8 |
| 101 | 110_110_100_100 = 0xDA4 | 110_111_000_000 = 0xDC0 |
| 100 | 110_110_000_000 = 0xD80 | 111_000_000_000 = 0xE00 |
| 011 | 110_000_000_000 = 0xC00 | 001_001_001_001 = 0x249 |
| 010 | ACCSET2 | ACCSET2 |
| 001 | ACCSET1 | ACCSET1 |
| 000 | 000_000_000_000 = 0x000 | 000_000_000_000 = 0x000 |

### 9.4.1.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | EALO | | | | 0 | | | 0 | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | D2SEL | | | D1SEL | | | D0SEL | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.21.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>EALO | Exclusive Access Lock Owner<br>When EAL=11, this 4-bit read only field indicates which domain owns the exclusive access lock. See section Exclusive Access Lock for details on usage of the exclusive access lock. This field is reserved for devices that do not include the exclusive access lock. |
| 23-21<br>— | Reserved. |
| 20-18<br>— | Reserved. |
| 17-15<br>— | Reserved. |
| 14-12<br>— | Reserved. |
| 11-9<br>— | Reserved. |
| 8-6<br>D2SEL | Domain 2 select<br>Refer to Domain 0 select. |
| 5-3<br>D1SEL | Domain 1 select<br>Refer to Domain 0 select. |
| 2-0<br>D0SEL | Domain 0 select<br>This field selects the access flags in Table 9-8 (with the coderegion=0,coderegion=1). 3-bit select of 4*{r,w,x} access flag sets for domain 0.<br>• DxSEL = 0, 3-7 Select one predefined constant {r,w,x} flags, = f(MRGD_W4_n[CR])<br>• DxSEL = 1 Select ACCSET1 4*{r,w,x} fully-programmable access flags<br>• DxSEL = 2 Select ACCSET2 4*{r,w,x} fully-programmable access flags<br>See Table 9-8. |

## 9.4.1.22 Memory Region Descriptor (MRGD_W3_0_0 - MRGD_W3_1_7)

### 9.4.1.22.1 Offset

For m = 0 to 7; n = 0 to 1:

| Register | Offset |
|---|---|
| MRGD_W3_n_m | 200Ch + (n × 200h) + (m × 20h) |

### 9.4.1.22.2 Function

Refer to Memory Region Descriptor (MRGD_W0_0_0 - MRGD_W0_1_7) for description.

### 9.4.1.22.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CR | 0 | | 0 | | | EAL | | 0 | | | 0 | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | 0 | | | 0 | | | 0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 9.4.1.22.4 Fields

| Field | Function |
|---|---|
| 31<br>CR | Code Region Indicator<br>Code region indicator (0=data, 1=code); selects between two sets of constant {r,w,x} flags(See Table 9-8) |
| 30-29<br>— | Reserved |
| 28-26<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 25-24<br>EAL | Exclusive Access Lock<br>This 2-bit field provides a mechanism to dynamically modify the DxACP permission evaluation such that only one domain has access permissions at a time. This field is written independently of the rest of the W3 register. Writes with a value of 00 do not affect the exclusive access lock and are used to update other fields of the W1 register. Writes with a non-zero value do not update other fields of the W3 register. Neither the global configuration lock nor the descriptor lock apply to this field. See section Exclusive Access Lock for details on usage of the exclusive access lock. This field is reserved for devices that do not include the exclusive access lock.<br>00b - Lock disabled<br>01b - Lock disabled until next reset<br>10b - Lock enabled, lock state = available<br>11b - Lock enabled, lock state = not available |
| 23-21<br>— | Reserved. |
| 20-18<br>— | Reserved. |
| 17-15<br>— | Reserved. |
| 14-12<br>— | Reserved. |
| 11-9<br>— | Reserved. |
| 8-6<br>— | Reserved. |
| 5-3<br>— | Reserved. |
| 2-0<br>— | Reserved. |

## 9.4.1.23 Memory Region Descriptor (MRGD_W4_0_0 - MRGD_W4_1_7)

### 9.4.1.23.1 Offset

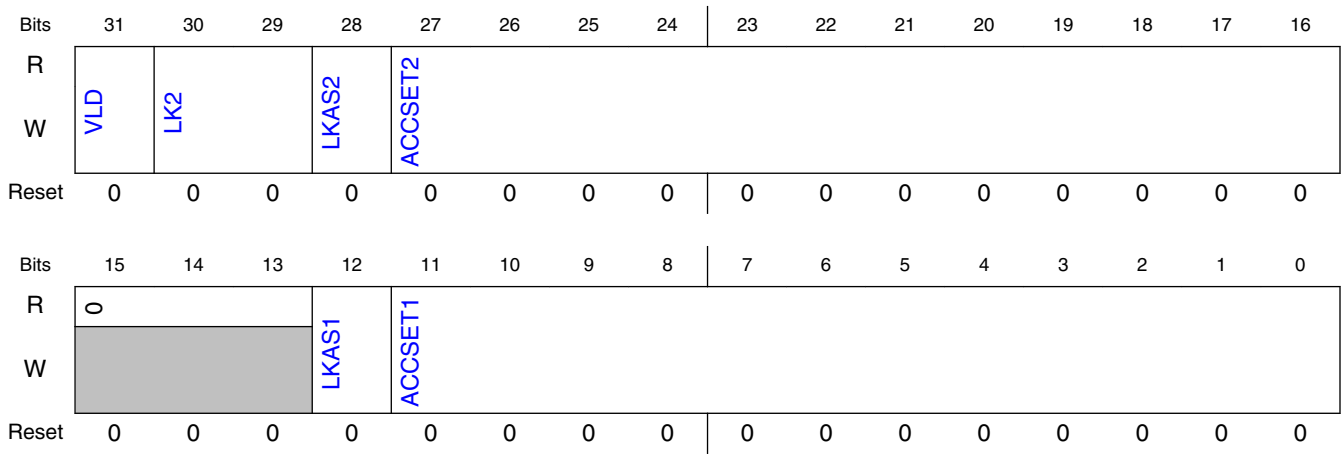
For m = 0 to 7; n = 0 to 1:

| Register | Offset |
|---|---|
| MRGD_W4_n_m | 2010h + (n × 200h) + (m × 20h) |

### 9.4.1.23.2 Function

Refer to Memory Region Descriptor (MRGD_W0_0_0 - MRGD_W0_1_7) for description.

### 9.4.1.23.3 Diagram




### 9.4.1.23.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>This field indicates the peripheral domain access control definition is valid. It is further qualified by CR[GVLDC] = 1. If CR[GVLDC] is set and VLD is cleared, every access to the peripherals are not allowed.<br>Any write to MRGD_W[0-3]_*n* clears the MRGD_W4_*n*[VLD] indicator so a coherent register state can be supported.<br>0b - The MRGDn assignment is invalid.<br>1b - The MRGDn assignment is valid. |
| 30-29<br>LK2 | Lock<br>This 2-bit field provides a mechanism to limit writes to the MRGDn register to protect its contents. Once set, these bits individually remain asserted until the next reset.<br>**NOTE:** This 2-bit lock does not apply to the EAL field.<br>00b - Entire MRGDn can be written.<br>01b - Entire MRGDn can be written.<br>10b - Domain *x* can only update the D*x*ACP field and the LK2 field; no other MRGDn fields can be written.<br>11b - MRGDn is locked (read-only) until the next reset. |
| 28<br>LKAS2 | Lock ACCSET2<br>If mode = PrivSecure, then writes update all 4 sets of {r,w,x} access flags<br>If mode = PrivNonsecure, then writes update {PN, US, UN}{r,w,x} flags, but PS{r,w,x} flags are not affected |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE:** Any write to MRGD_W[0-3]_n clears the MRGD_W4_n[LKAS2] field.<br>0b - Writes to ACCSET2 affect lesser modes<br>1b - ACCSET2 cannot be modified |
| 27-16<br>ACCSET2 | SET 2 of Programmable access flags.<br>• BIT 11:9 – PSR, PSW, PSX<br>PS{R,W,X} Read, write, execute flags for PrivSec mode (PS); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 8:6 – PNR, PNW, PNX<br>PN{R,W,X} Read, write, execute flags for PrivNonsec mode (PN); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 5:3 – USR, USW, USX<br>US{R,W,X} Read, write, execute flags for UserSec mode (US); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 2:0 – UNR, UNW, UNX<br>UN{R,W,X} Read, write, execute flags for UserNonsec mode (UN); flag = 0 inhibits access, flag = 1 allows access<br>**NOTE:** Any write to MRGD_W[0-3]_n clears the MRGD_W4_n[ACCSET2] field. |
| 15-13<br>— | Reserved |
| 12<br>LKAS1 | Lock ACCSET1<br>If mode = PrivSecure, then writes update all 4 sets of {r,w,x} access flags<br>If mode = PrivNonsecure, then writes update {PN, US, UN}{r,w,x} flags, but PS{r,w,x} flags are not affected<br>**NOTE:** Any write to MRGD_W[0-3]_n clears the MRGD_W4_n[LKAS1] field.<br>0b - Writes to ACCSET1 affect lesser modes<br>1b - ACCSET1 cannot be modified |
| 11-0<br>ACCSET1 | SET 1 of Programmable access flags.<br>• BIT 11:9 – PSR, PSW, PSX<br>PS{R,W,X} Read, write, execute flags for PrivSec mode (PS); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 8:6 – PNR, PNW, PNX<br>PN{R,W,X} Read, write, execute flags for PrivNonsec mode (PN); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 5:3 – USR, USW, USX<br>US{R,W,X} Read, write, execute flags for UserSec mode (US); flag = 0 inhibits access, flag = 1 allows access<br>• BIT 2:0 – UNR, UNW, UNX<br>UN{R,W,X} Read, write, execute flags for UserNonsec mode (UN); flag = 0 inhibits access, flag = 1 allows access<br>**NOTE:** Any write to MRGD_W[0-3]_n clears the MRGD_W4_n[ACCSET1] field. |

## 9.5 Functional description

This section provides more details on the operation and implementation of the XRDC submodules.

### 9.5.1 Manager (XRDC_MGR)

The MGR submodule is responsible for coordinating XRDC's programming model reads and writes.

The complete programming model is large, requiring four 4-Kbyte slave peripheral address space slots, although most devices define a sparsely populated subset. Additionally, the programming model is distributed across the various submodule instances (MDAC, MRC, PAC). The MGR submodule provides the required hardware management for all programming model accesses, generating and distributing the appropriate address decodes and write data to the MDAC, MRC and PAC submodule instances, collecting and combining all the register read data buses and responses.

MGR implements the Control Register, all the read-only hardware configuration registers (HWCFG[0-3], MDACFGn, MRCFG) and the program-visible versions of the domain error capture registers. To support the domain error reporting functionality, the MGR submodule collects the individual captured error indicator output signals from all the submodule instances and remaps them so they are organized into a instance bitmap by domain as specified by the DERRLOCn register array.

### 9.5.2 Master Domain Assignment Controller (XRDC_MDAC)

The MDAC submodule is responsible for the generation of the domainID on every memory transaction for every bus master in the device. The resulting domainID and {nonsecure, privileged} signals are generated and then treated as address attributes and associated with each transaction as it moves through the system.

MDAC modules that drive the domainIDs (DIDs) for processor cores include an option of processor-identifier (PID) matching in order to select between DIDs. This is accomplished by first enabling PID checking by setting XRDC_MDA_Wm_n[PE] = [2,3]. With PID matching enabled, a domain hit is determined by matching XRDC_MDA_Wm_n[PID] with the PID input or PID in the XRDC_PID register. The PIDM field can be used to match specific groups of PIDs.

As a simple example, assume a system with two domainIDs.

- domainID 1 is for critical tasks (“critical” is system-specific and may refer to timing, safety, and so on.).
- domainID 2 is for non-critical tasks.
- Two domainIDs require two MDA_Wm_n registers, MDA_W0 and MDA_W1.

Further assume the processor’s task identifier defines whether a task is critical or non-critical as defined by system software.

- PIDs [0-15] are assigned to critical tasks.
- PIDs != [0-15] are assigned to noncritical tasks.

During startup, software initializes the MDA_W[0-1] registers. MDA_W0 defines DID = 1 for PIDs [0-15] and MDA_W1 defines DID = 2 for PIDs != [0-15].

1. MDA_W0 = 0x8000_0F81
    - VLD = 1
    - PID = 0x0
    - PIDM = 0xF
    - PE = 2
    - DIDS = 0
    - DID = 1
2. MDA_W1 = 0x8000_0FC2
    - VLD = 1
    - PID = 0x0
    - PIDM = 0xF
    - PE = 3
    - DIDS = 0
    - DID = 2

As the processor executes, it loads the appropriate task ID into its PID register as the task is started. The processor’s PID register value is input to the corresponding MDAC module and used by the MDA_Wm comparison logic. The system then dynamically generates two domainIDs and the downstream XRDC access check logic can distinguish and enforce different access control rights based on the different domainIDs.

A special mode is provided for driving the DID of non-processor masters. This is enabled with the DID bypass (DIDB) bit. The MDAC has a DID input that can be driven by a master and used directly as the DID output. This allows variability in the DID as dictated by the master and is intended to be used by DMA masters that masquerade as, (that is, drive the DID of) the master that programmed it.

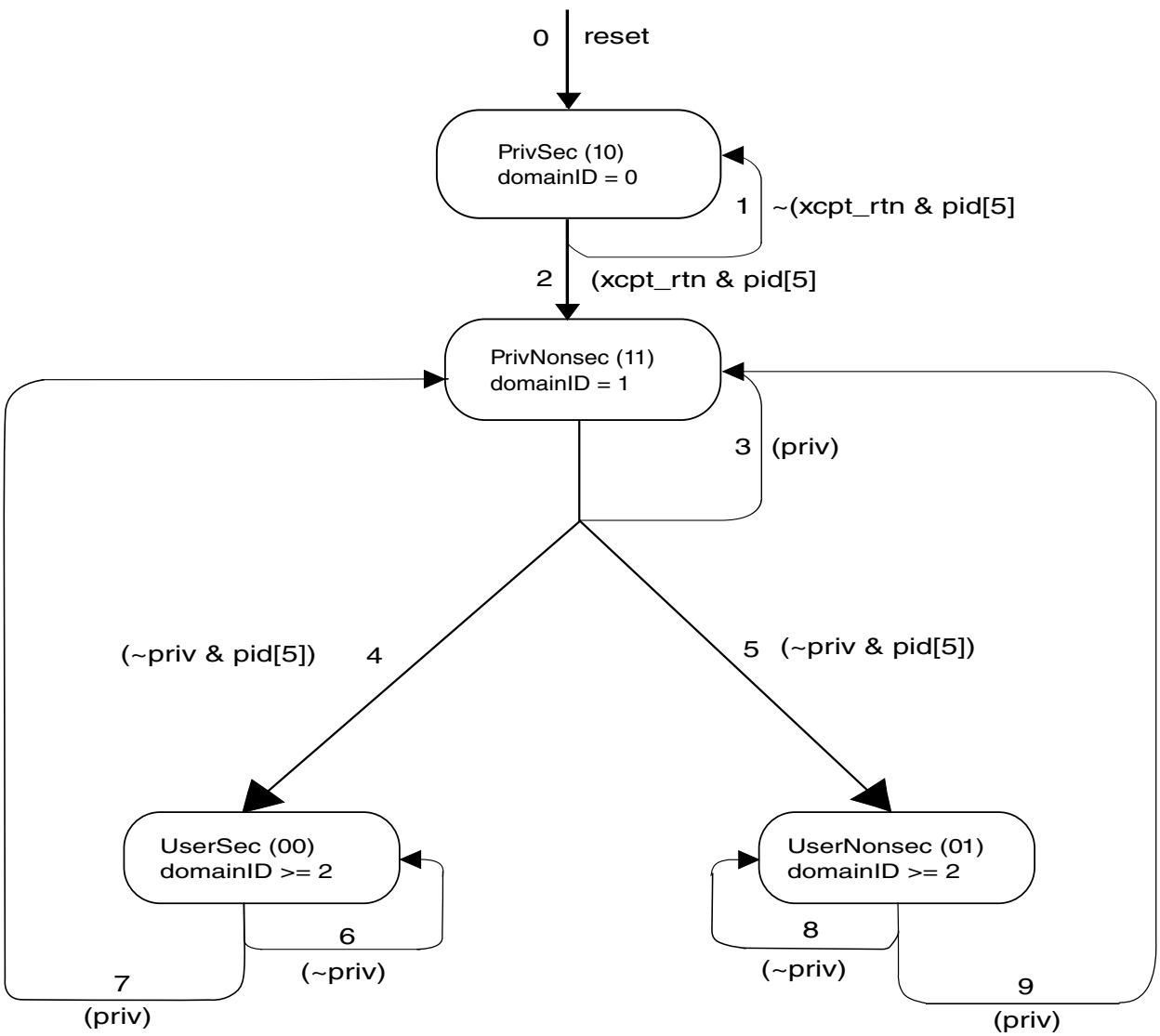
**NOTE**

Do not change the XRDC_MDAC configuration of a master while that master is active and XRDC_CR[GVLD] is set.

## 9.5.3 Special 4-State Model

When the special 4-state model (SP4SM) is enabled, a Mealy finite state machine in each processor MDACn tracks the current 2-bit access mode and generates specific domainIDs for specific modes.

The mode states and their transitions are shown in the diagram. In this configuration, the access mode states include PrivSecure (10), PrivNonsecure (11), UserSecure (00) and UserNonsecure (01).



### 9.5.4 DxACP evaluation

Fundamental to the XRDC's operation is the actual access violation check performed by the memory region (MRC) and peripheral access (PAC) controllers.

Previous descriptions have detailed the domain assignment mechanisms and the tracking of the domain identifier as an address attribute through the system bus switching fabric(s). As transactions reach the slave memory and peripheral controllers, the address is used to select the appropriate memory region descriptor or peripheral access control register. Once the appropriate register has been selected, the next function is the actual access evaluation.

For this step, the domainID of the current transaction selects the appropriate 3-bit DxACP (domain "x" access control policy) from the source register. The DxACP field defines the access rights for the domain based on the 4-level model combined with the transaction's access attributes (read = 0/write = 1, secure = 0/nonsecure = 1, user = 0/privileged = 1).

The 4-level model is defined by the concatenation of two address attributes: secure/nonsecure and user/privileged attributes. These two attribute signals (nonsecure, privileged) and the resulting access levels are defined as:

```
if {privileged, nonsecure} == 0b10, then level = PrivSecure
if {privileged, nonsecure} == 0b11, then level = PrivNonsecure
if {privileged, nonsecure} == 0b00, then level = UserSecure
if {privileged, nonsecure} == 0b01, then level = UserNonsecure
```

Recall the resulting hierarchical access control model specifies:

```
PrivSecure > PrivNonsecure > UserSecure > UserNonsecure
```

The XRDC's evaluation of the access control policy is defined by Table 9-9.

**Table 9-9. Domain x Access Control Policy Definition**

| Policy | Allowable Accesses | | | |
|---|---|---|---|---|
| | Secure Priv | Secure User | Nonsecure Priv | Nonsecure User |
| 111 | r, w | r, w | r, w | r, w |
| 110 | r, w | r, w | r, w | none |
| 101 | r, w | r, w | r | r |
| 100 | r, w | r, w | r | none |
| 011 | r, w | r, w | none | none |
| 010 | r, w | none | none | none |
| 001 | r | r | none | none |
| 000 | none | none | none | none |

Unimplemented domains default to an all-zero DxACP field providing no access rights.

After the DxACP evaluation is complete and the access error signal generated, the transaction is allowed to continue if it has sufficient access rights, else the access is aborted with the address and attribute information captured in the appropriate error registers.

Recall XRDC optionally supports the inclusion of a hardware semaphore to *dynamically alter* the access control policy of a memory region or peripheral access. See Hardware semaphores and dynamic access rights for details.

### 9.5.5 Exclusive Access Lock

The exclusive access lock is a hardware mechanism to dynamically modify the DxACP permission evaluation such that only one domain has access to a peripheral or memory region at any given time. Once an exclusive access lock is enabled, it is enabled until the next reset and the lock must be acquired by a domain to enable its access permissions. Only one domain can own the lock at a time and only the lock owner's access permissions are enabled, read and write permissions for all other domains are revoked. Some devices do not have exclusive access locks, see the appropriate section of the reference manual to determine if a device includes this feature.

Owning an exclusive access lock only enables permissions granted by the DxACP field, it does not grant permissions. Only domains with some read or write permissions (non-zero DxACP value) can acquire or release a lock. A lock can be held indefinitely. Typically a lock is released after required accesses are completed to allow other domains to acquire the lock as needed. In normal operation, only the domain that owns the lock can release it. There is a safeguarded means for a domain which does not own a lock to force a lock release.

The exclusive access lock is enabled, acquired and released by writing to the 2-bit EAL descriptor field. Neither the global configuration lock nor the descriptor lock apply to this field. The EAL field can be written to independently of the rest of the descriptor fields. Non-zero writes (01,10,11) to the EAL field do not affect other fields of the register. Likewise, a write of 00 does not affect the exclusive access lock and other fields of the register are updated. This convention eliminates the need to perform a read, modify, write operation to control the exclusive access lock. For devices without the exclusive access lock feature, the EAL field is a read only, reserved space and register writes are not affected by non-zero writes to the EAL field.

The EAL values are shown table below.

| EAL | Lock Enabled | Lock State | Access Permissions | Allowed Transitions |
|---|---|---|---|---|
| 00 | No | --- | Per DxACP evaluation | 01 disable lock |

*Table continues on the next page...*

| EAL | Lock Enabled | Lock State | Access Permissions | Allowed Transitions |
|---|---|---|---|---|
| | | | | 10 enable lock |
| 01 | No | --- | Per DxACP evaluation | None |
| 10 | Yes | Available | Write permissions revoked for all domains | 11 acquire lock |
| 11 | Yes | Not available | Per DxACP evaluation for the domain that owns the lock, write permissions revoked for all other domains | 10 release lock |

If EAL=11, the owner of the lock is indicated by the 4-bit, read only EALO field. The EALO field is used in the lock acquisition procedure to determine when the lock acquisition is successful. Domains with a DxACP field value of 000 can not acquire or release the lock.

The EAL field has a reset value of 00 which disables the exclusive access lock. The lock is disabled until the next reset by writing 0x01000000 to the W1 register (PAC/MSC) or W3 register (MRC) which sets EAL=01. Alternatively, the exclusive access lock is enabled until the next reset and set to the available state by writing 0x02000000 to the W1 register (PAC/MSC) or W3 register (MRC) which sets EAL=10.

Once the exclusive access lock is enabled, domains must acquire the lock to enable their access permissions. Only one domain can own the lock at a time. The procedure for acquiring the lock is as follows:

1. Set EAL = 11 by writing 0x03000000 to the W1 register (PAC/MSC) or W3 register (MRC)
2. Read EALO
3. If EALO = DID then the lock is acquired, else delay and repeat steps 1-3.

The DID value can be determined by reading the SR.

The exclusive access lock is released by writing 0x02000000 to the W1 register (PAC/MSC) or W3 register (MRC) which sets EAL=10. Only the domain that owns the lock (EALO=DID) may release the lock with the normal lock release write.

A lock can be forced to the available state (EAL=10) by a domain that does not own the lock through the forced lock release procedure. The procedure to force a exclusive access lock release is as follows:

1. Write 0x02000046 to W1 register (PAC/MSC) or W3 register (MRC)
2. Write 0x02000052 to W1 register (PAC/MSC) or W3 register (MRC)

Only the EAL field and the least significant byte of the forced release writes are significant and detected by hardware. The two writes must be consecutive, any intervening write to the register resets the sequence. No other register fields are affected by the forced release write. Once the lock is released, it can be acquired by any domain. As a convention, normal lock release writes should use 0x02000000 to clearly distinguish normal release writes from other configuration register writes and forced release writes.

### 9.5.6 Memory Region Controller (XRDC_MRC)

The XRDC's Memory Region Controller provides domain-based, hardware access control for all system bus references targeted at non-peripheral memory spaces. Using pre-programmed *region descriptors* which define memory spaces and their associated access rights per domain identifier, the MRC concurrently monitors system bus transactions and evaluates the appropriateness of each transfer. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

The XRDC architectural framework supports up to 16 MRC instances, where each MRC instance can support {4, 8, 12, 16} region descriptors, concurrently monitoring up to 8 slave memory ports. Note, monitoring an AXI protocol port counts as two ports since there are independent read address and write address channels. Conversely, an AHB connection counts as a single port.

A partial, simplified one-dimensional block diagram of an instance of the MRC module is shown in Figure 9-2. In this figure, a single slave bus port is shown. When the remaining slave ports are considered, the MRC hardware's two-dimensional connection matrix, broadcasting each system bus slave port access across the implemented memory region descriptors, is the resulting structure with the basic access evaluation macro shown as the replicated submodule block. In Figure 9-2, the system bus fabric slave port is shown on the left, the memory region descriptor registers are in the middle, and the peripheral bus interface on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and, based on the domainID associated with the reference, access violations.

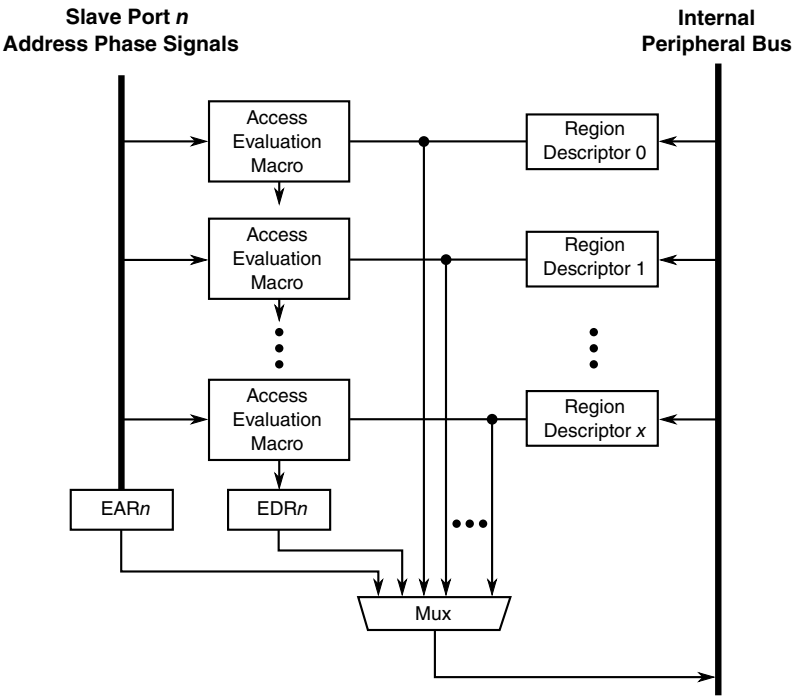



**Figure 9-2. MRC access evaluation macro hardware structure**

For each access, the MRC hardware performs a two-step evaluation process. First, the access is compared to all memory region descriptors to determine hit/miss status, and then, using the domainID associated with the reference plus the address attributes {write, nonsecure, privileged}, the access rights are examined using the method detailed in DxACP evaluation.

### 9.5.6.1 Memory region descriptor hit determination

To determine if the current reference hits in a given region, two magnitude comparators are used in conjunction with the region's start and end addresses. The boolean equation for this portion of the hit determination is defined as:

```
region_hit_n =
   ((addr >= start_addr) & (addr <= end_addr)) & valid
```

where `addr` is the current system reference address, `start_addr]` and `end_addr` are the start and end addresses, and `valid` is the valid bit, all from memory region descriptor n.

Note there are *no hardware checks* to verify that `end_addr ≥ start_addr`, and it is software's responsibility to properly load appropriate values into these fields of the region descriptor.

Also, the region hit determination is based only on an address comparison of the first byte being accessed; that is, the MRC does *not* check the size of the access to make sure it entirely fits within the region.

### 9.5.6.2 Memory region DxACP evaluation

For each region descriptor hit, the MRC logic then evaluates the access rights defined by the MRGD_Wm_n registers. Specifically, the domainID attribute selects the appropriate DxACP field from the MRGD_W{2,3}_n registers and then the {write, nonsecure, privileged} address attributes are compared with the access policy defined by the selected DxACP. See DxACP evaluation for additional details on this function.

While CR[GVLDC] = 1, and for each bus slave port being monitored, the MRC performs a **reduction-AND** of all the individual *(no_hit | error)* terms from each access evaluation macro. Recall as specified in Memory region descriptor hit determination, an *invalid* MRGD_Wm_n (MRGD_W3_n[VLD] = 0) forces the region_hit_n evaluation to be negated.

This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. If the access does not hit in *any* region descriptor, an access error is reported.
2. If the access hits in a single region descriptor and that region signals a domain violation, then an access error is reported.
3. If the access hits in multiple (overlapping) regions and one region signals a violation, then an access error is reported.

The third condition reflects that priority is given to access denying over access allowing for overlapping regions.

Unimplemented domain identifiers default to all-zero DxACP fields providing no access rights. Additionally, these fields are RAZ/WI.

### 9.5.6.3 MRC C code description

The functionality of the MRC evaluation is described in the C code of Listing 9-1 on page 308. This code begins with the description of the data structures corresponding to the MRC-associated registers (lines 1-12). Note the memory region descriptors are

described as a 3-dimensional unsigned 32-bit integer array indexed by the MRC instance number, the RGD number within the instance and then the word offset within the region descriptor.

The access evaluation consists of four basic functions: a check of the XRDC's global valid bit (lines 43-45), the address hit/miss determination (lines 73-78), the selection of the DxACP field and its evaluation (lines 80-96) and finally, the combination of all the individual MRGD_Wm_n hit and error indicators to determine the final state of the memory reference (lines 99-134).

**Listing 9-1. C code for MRGDn R,W,X Evaluation**

```
  unsigned int evaluate_MRGDn (
    unsigned int n, /* mrgdID */
    unsigned int dID, /* domainID */
    unsigned int r, /* dataRead */
    unsigned int w, /* dataWrite */
    unsigned int x, /* instFetch */
    unsigned int accMode) /* accessMode */
{
    unsigned int dxSel; /* domain "x" select */
    unsigned int codeRegion; /* code or data region identifier */
    unsigned int rwx12; /* selected 12-bit {r,w,x} flags */
    unsigned int rwx3; /* final 3-bit {r,w,x} flags */
    unsigned int memAccess; /* encoded memory access: r=4, w=2, x=1 */
    unsigned int accError; /* access error: noError = 0, error = 1 */

     memAccess = 4 r + 2 w + x; /* encoded access type */

    if (dID < 8) {
      dxSel = (MRGD[n][2] >> (dID * 3)) & 0x7; /* extract dxSEL */
    } else {
      dxSel = (MRGD[n][3] >> ((dID-8) * 3)) & 0x7; /* extract dxSEL */
    }

    codeRegion = (MRGD[n][4] /* extract CR indicator */

/* use the dxSEL value to select the 12-bit set of {r,w,x} flags */

  switch (dxSel) {

    case 0:
          rwx12 = 0x000; /* r,w,x = 000_000_000_000 */
        break;

    case 1:
          rwx12 = ( MRGD[n][4]) & 0xfff;
        break;

    case 2:
          rwx12 = (MRGD[n][4] >> 16) & 0xfff;
        break;

    case 3:
        if (codeRegion == 1)
            rwx12 = 0x249; /* r,w,x = 001_001_001_001 */
        else
            rwx12 = 0xc00; /* r,w,x = 110 000 000 000 */
        break;

    case 4:
        if (codeRegion == 1)
            rwx12 = 0xe00; /* r,w,x = 111_000_000_000 */
        else
```

```
            rwx12 = 0xd80; /* r,w,x = 110_110_000_000 */
        break;

    case 5:
        if (codeRegion == 1)
            rwx12 = 0xdc0; /* r,w,x = 110_111_000_000 */
        else
            rwx12 0xda4; /* r,w,x 110 110 100 100 */
        break;
    case 6:
        if (codeRegion == 1)
            rwx12 = 0xdb8; /* r,w,x = 110_110_111_000 */
        else
            rwx12 = 0xdb0; /* r,w,x = 110_110_110_000 */
        break;
    case 7:
        if (codeRegion == 1)
            4 r + 2 w + x; / / rwx12 0xdbf; /* r,w,x 110 110 111 111 */
        else
            rwx12 = 0xdb6; /* r,w,x = 110_110_110_110 */
        break;
    }

/* extract the 3-bit {r,w,x} flags based on the accMode */

      switch (accMode & 3) {
    case 0:    /* accMode = userSecure */
        rwx3 = (rwx12 >> 3) &7;
        break;
    case 1:    /* accMode = userNonsecure */
        rwx3 = (rwx12 >> 0) &7;
        break;
    case 2:    /* accMode = privSecure */
        rwx3 = (rwx12 >> 9) &7;
        break;
    case 3:    /* accMode = privNonsecure */
        rwx3 = (rwx12 >> 6) &7;
        break;
    }

    rwx3 ^= 7; /* invert 3-bit {r,w,x} flags */
    if ((rwx3 & memAccess) == 0) {
        accError = 0;
    } else {
        accError = 1;
    }
    return(accError);
}
```

## 9.5.7 Peripheral Access Controller (XRDC_PAC)

The peripheral access controller performs an access control evaluation similar to that of the memory region controller (Memory Region Controller (XRDC_MRC), but using dedicated "memory region descriptors" with one unique PDAC_W[0-1]_n register per slave peripheral slot. The presence of a one and only one memory region descriptor per slave peripheral address space slot in the PAC simplifies the required functionality compared to the MRC.

The functionality of the PAC evaluation is described in the C code of Listing 9-2 on page 310. This code begins with the description of the data structures corresponding to the PAC-associated registers (lines 1-3). It then performs a check of the global valid bit (CR[GVLDP]) (lines 25-27), followed by the extraction of the peripheral slot number from the system address (line 32), then followed by the access of the selected PDAC_W[0-1]_n registers (lines 33-34). Next, a check of the optional inclusion of a hardware semaphore and its current state is performed (lines 37-46). Finally, if the access is still allowed, an evaluation of the appropriate DxACP permission field is made (line 48) as detailed in DxACP evaluation and the state of the peripheral access error control signal returned (line 52).

**Listing 9-2. PAC Evaluation C Code Description**

```
 1      // PAC registers
 2      unsigned int    xrdc_cr_gvldp;
 3     unsigned int    xrdc_pdac[512][2];
 4
 5   evaluate_PAC (sysAddr, write, privileged, nonsecure, did)
 6       unsigned int    sysAddr;                    // system address
 7       unsigned int    write;                      // read/write attribute
 8       unsigned int    privileged;                 // user/privileged attribute
 9       unsigned int    nonsecure;                  // secure/nonsecure attribute
10       unsigned int    did;                        // domain identifier
11   {
12       unsigned int    ips_slot_number;            // f(system address)
13       unsigned int    pdacw0;                     // selected PAC_W0_n register
14       unsigned int    pdacw1;                     // selected PAC_W1_n register
15       unsigned int    pac_ealo;                 // local copy of referenced ealo
16       unsigned int    peripheral_access_error;
17
18       // the evaluation of the PACn involves a 4-step process:
19       //   1) select the appropriate PDACn using the system address
20       //   2) using the DID for the access, select the appropriate DxACP field
21       //   3) evaluate the state of the (optional) ealo for this access
22       //   4) using the {secure/nonsecure, user/priviledged} attributes, eval DxACP
23
24       // first, check the global XRDC valid bit
25       if (xrdc_cr_gvldp == 0) {
26           peripheral_access_error = 0;
27             }
28
29       else {
30           // extract the peripheral slot number from the system address
31           // sysAddr[20:12] = slot select [0-511]
32           ips_slot_number = (sysAddr & 0x001ff000) >> 12;
33           pdacw0 = xrdc_pdac[ips_slot_number][0];
34           pdacw1 = xrdc_pdac[ips_slot_number][1];
35
36           peripheral_access_error = 0;
37           if ((pdacw1 & 0x80000000) == 0x80000000) {          // PDACn is valid
38               // semaphore check
39               pac_ealo = ealo[(pdacw0 & 0x3f000000) >> 24]; // local copy
40               // check if it is enabled on a write and if not idle, nor == (did+1)
41               if (((pdacw0 & 0x40000000) == 0x40000000)   &&  // se
42                    (write == 1)                           &&  // wt
43                    (pac_ealo != 0)                        &&  // ealo != idle
44                    (pac_ealo != (did+1)))                 {   // ealo != did
45                   peripheral_access_error = 1;
46               } else {
47                   // perform the standard DxACP check
48                   peripheral_access_error = evaluate_DxACP(pdacw0, pdacw1, did,
\                                                   nonsecure, privileged, write);
```

```
49                  }
50              }
51                }
52                return peripheral_access_error;
53    }
```

## 9.6 Initialization information

Out of reset, CR[GVLDC, GVLDP, GVLDM] bits are cleared and XRDC disabled allowing secure privileged startup code to configure the entire programming model. The initialization process typically is performed in 3 steps:

1. Read the various hardware configuration registers (HWCFG{0-3}, MDACFGn, MRCFGn) to obtain the implemented hardware capabilities for the device.
2. Using the information retrieved in Step 1 coupled with the desired domain architecture, program all the register data structures for domain assignment (MDA_Wm_n), memory region descriptors (MRGD_Wm_n) and peripheral access control register (PDAC_Wm_n). There are individual valid bits included in all these registers which typically would be asserted. Additionally, there are also lock mechanisms available if register settings need to be configured and marked as read-only.
3. Once the XRDC programming model is loaded, the CR is written with GVLD {C, P, M} asserted. At that point, the XRDC is fully operational.

## 9.7 Application information

As described in Introduction, the XRDC architecture is intended to provide a broad, highly-capable framework for access control, system memory protection and peripheral isolation. The resulting microarchitecture implementation is distributed throughout the core platform, highly configurable via hardware design parameters and software programmability. The considerable number of hardware design parameters allows the XRDC architecture to be a very scalable solution, able to service a wide spectrum of SoC devices from ultra-low end MCUs to multi-core high performance embedded microprocessors.

### 9.7.1 Master domain assignments

The typical use case related to master domain assignments is to include one (or more) processor core(s) in a single domain, possibly coupled with other bus master devices like DMA, etc. The definition of a domain may be static, based simply on the combination of a processor and other optional bus masters. Alternatively, the processor's operation may be configured to dynamically select between a very small number of domains. The maximum number of possible CPU domain definitions is limited by MDACFGn value. In particular, the optional inclusion of PID value(s) to be used to create multiple classes of CPU with different domain values.

For example, "critical" tasks, whether safety-critical, performance-critical, etc. can be grouped together in one domain and all other tasks into a second domain. Non-processor bus masters typically have a single MDA_Wm_n configuration register associated with them. The domainID and {nonsecure, privileged} attributes are usually statically assigned, unless the module can "inherit" attributes from a processor programming it (like certain modules like DMAs).

The MDA_Wm_n registers, along with the memory region descriptors and the peripheral domain access control registers all have lock features that allow the register resources to be converted into read-only resources to protect the written configuration.

### 9.7.2 Memory region descriptor management

There are two important concepts to consider in managing the memory region descriptors.

First, recall that the association between each MRCn instance and the slave memory regions it monitors is SoC-dependent. The device-specific configuration specifies the number of implemented memory region descriptors in a given instance and the specific port numbers associated with the slave memories being monitored.

Second, as detailed in Memory region DxACP evaluation, recall that, once the XRDC is enabled, *every memory reference must hit in one or more regions*, else an access violation is reported. These are also two other conditions were access errors are reported:

1. If the access hits in a single region descriptor and that region signals a domain violation, then an access error is reported.
2. If the access hits in multiple (overlapping) regions and *all* regions signal violations, then an access error is reported.

### 9.7.3 Domain error capture management

Recall when a domain access violation is detected by either a memory region controller or a peripheral access controller, address and attribute information of the offending access is captured. The DERRLOCn read-only registers provide additional information signaling the instance number of the submodule where the access violation(s) occurred. Since the resulting exception handler needs the submodule instance to retrieve the captured address and attribute information from DERR_W0_n and DERR_W1_n, the DERRLOCn registers provide the instance number details.

These registers are organized as a word array, *indexed by the faulting domain number*, with the contents of each register providing a bitmap signaling the instance number(s) associated with all submodules containing captured error information for that domain.

It is recommended that the exception handler begin by reading the HWCFG1 register to determine its domainID. Next, it uses the just-retrieved domainID to index into the DERRLOCn array. The resulting DERRLOCn value is then examined to determine the instance number of the reporting MRC and/or PAC submodule.

There may be multiple access violations, across multiple instances, pending for a given domain. It is suggested that exception handler use a "find first one" instruction (alternatively known as "count leading zeroes") to quickly and efficiently find the instance number containing access violation details. Once the instance number has been determined, the captured error address (DERR_W0_n) and attribute information (DERR_W1_n) can easily be retrieved from the domain error registers. The 2-bit DERR_W1_n[EST] field provides information signaling whether one or more errors have been detected by the submodule instance. A special write to DERR_W3_n is required to reset (and rearm) the EST error capture state machine.

This process is repeated until all errors associated with a given domainID have been processed.

A graphical representation of this 2-step retrieval of the domain error address and attributes is shown in Figure 9-3.

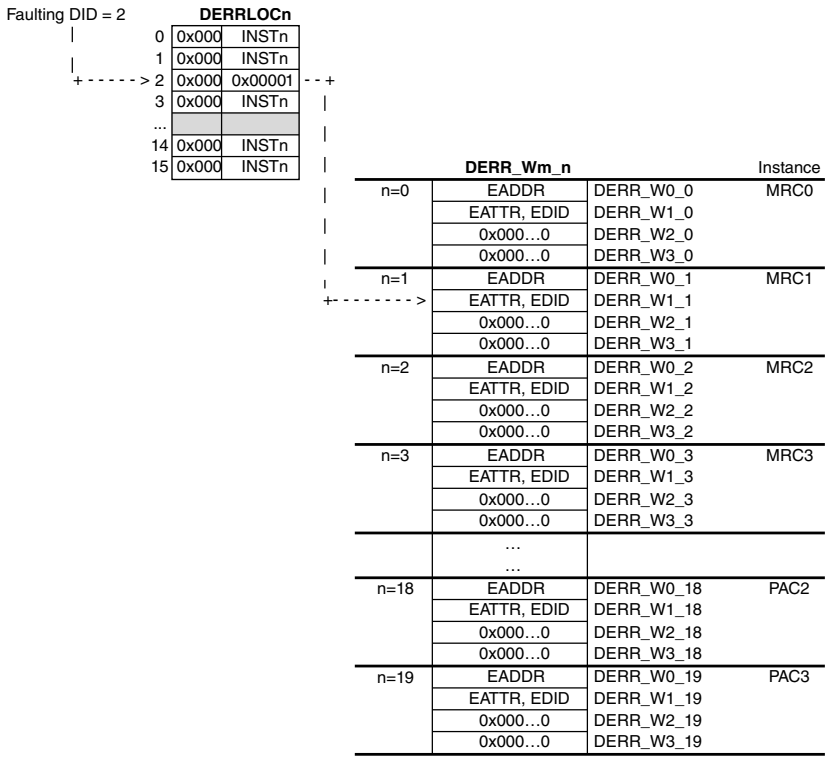



**Figure 9-3. Two-step retrieval of domain error address and attributes**

# Chapter 10
# Semaphores2 (SEMA42)

## 10.1 Introduction

The SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

### 10.1.1 Multi-core programming 101: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism that is then used by system software to control access to shared data structures, shared hardware resources, and so on. These gating mechanisms are used by the software to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case. Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered to be locking (or closing) a software gate. Once the gate has been locked, a properly architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate; in other words, other processes/processors are locked out. Many software implementations include a *spin-wait loop* within the lock function until the locking of the gate is accomplished. Once the lock has been obtained, processor X continues execution and updates the data values protected by the particular lock. Once the updates are complete, processor X unlocks (or opens) the software gate, allowing other processes/processors access to the updated data values.

There are three important rules that must be followed for a correctly implemented system solution:

1. All writes to shared data values or shared hardware resources must be protected by a gate variable.
2. Once a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
3. The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor can be extremely useful for system-level debugging.

The Hennessy/Patterson text on computer architecture offers the following description of software gating.

"One of the major requirements of a shared-memory architecture multiprocessor is being able to coordinate processes that are working on a common task. Typically, a programmer will use *lock variables* to synchronize the processes.

The difficulty for the architect of a multiprocessor is to provide a mechanism to decide which processor gets the lock and to provide the operation that locks a variable. Arbitration is easy for shared-bus multiprocessors, since the bus is the only path to memory. The processor that gets the bus locks out all the other processors from memory. If the CPU and bus provide an atomic swap operation, programmers can create locks with the proper semantics. The adjective *atomic* is key, for it means that a processor can both read a location **and** set it to the locked value in the same bus operation, preventing any other processor from reading or writing memory." [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*]

The classic text continues with a description of the steps required to lock/unlock a variable using an *atomic swap instruction*.

"Assume that 0 means unlocked and 1 means locked. A processor first reads the lock variable to test its state. A processor keeps reading and testing until the value indicates that the lock is unlocked. The processor then races against all other processes that were similarly "spin waiting" to see who can lock the variable first. All processes use a swap instruction that reads the old value and stores a 1 into the lock variable. The single winner will see the 0, and the losers will see a 1 that was placed there by the winner. (The losers will continue to set the variable to the locked value, but this does not matter.) The winning processor executes the code after the lock and then stores a 0 into the lock when

it exits, starting the race all over again. Testing the old value and then setting to a new value is why the atomic swap instruction is called *test and set* in some instruction sets." [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*]

Next, consider the features, programming model, and functional operation of the hardware semaphore module.

## 10.1.2 Features

The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Supports 16 hardware-enforced gates in a multi-processor configuration, where up to 15 processors can be supported; cp*X* is meant to represent core processor *X*
  - Gates appear as an 16-entry byte-size array with read and write accesses.
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked
      if gate = 0x1, then state = locked by processor (master) 0
      if gate = 0x2, then state = locked by processor (master) 1
      …
      if gate = 0xF, then state = locked by processor (master) 14
    - Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
    - Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.
  - Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.
- Memory-mapped IPS slave peripheral platform module

- Interface to the IPS bus for programming-model accesses

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2), gate (n-1) include the finite state machines implementing the semaphore gates.

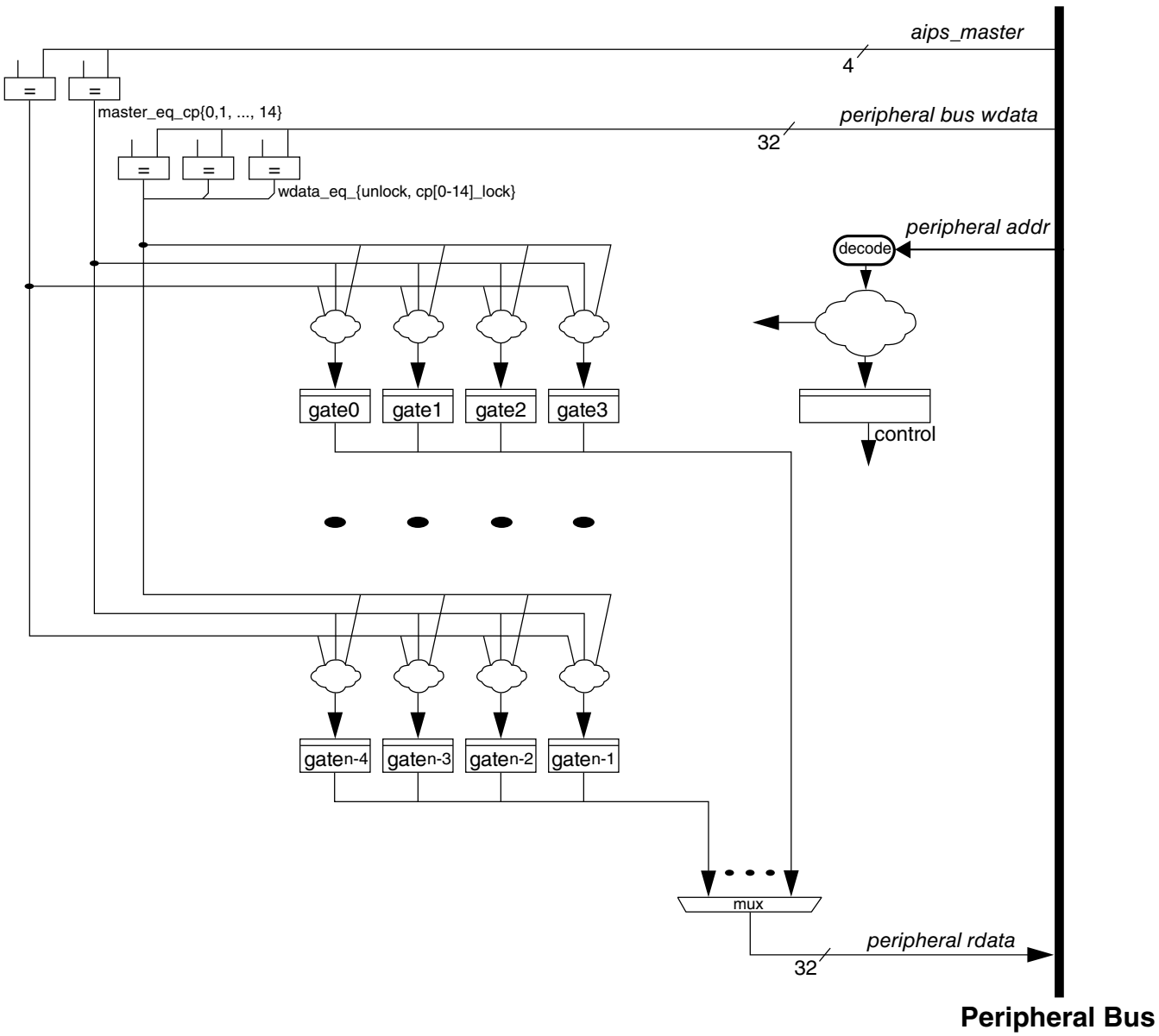



**Figure 10-1. Semaphores2 block diagram**

## 10.2 Memory map/register definition

Only Supervisor Mode accesses are allowed on these registers. User accesses will be terminated with an error.

## 10.2.1 SEMA420 register descriptions

### 10.2.1.1 SEMA420 Memory map

SEMA420 base address: 4001_B000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Gate Register (GATE3) | 8 | RW | 00h |
| 1h | Gate Register (GATE2) | 8 | RW | 00h |
| 2h | Gate Register (GATE1) | 8 | RW | 00h |
| 3h | Gate Register (GATE0) | 8 | RW | 00h |
| 4h | Gate Register (GATE7) | 8 | RW | 00h |
| 5h | Gate Register (GATE6) | 8 | RW | 00h |
| 6h | Gate Register (GATE5) | 8 | RW | 00h |
| 7h | Gate Register (GATE4) | 8 | RW | 00h |
| 8h | Gate Register (GATE11) | 8 | RW | 00h |
| 9h | Gate Register (GATE10) | 8 | RW | 00h |
| Ah | Gate Register (GATE9) | 8 | RW | 00h |
| Bh | Gate Register (GATE8) | 8 | RW | 00h |
| Ch | Gate Register (GATE15) | 8 | RW | 00h |
| Dh | Gate Register (GATE14) | 8 | RW | 00h |
| Eh | Gate Register (GATE13) | 8 | RW | 00h |
| Fh | Gate Register (GATE12) | 8 | RW | 00h |
| 42h | Reset Gate Read (RSTGT_R) | 16 | RO | 0000h |
| 42h | Reset Gate Write (RSTGT_W) | 16 | WO | See description. |

### 10.2.1.2 Gate Register (GATE0 - GATE15)

#### 10.2.1.2.1 Offset

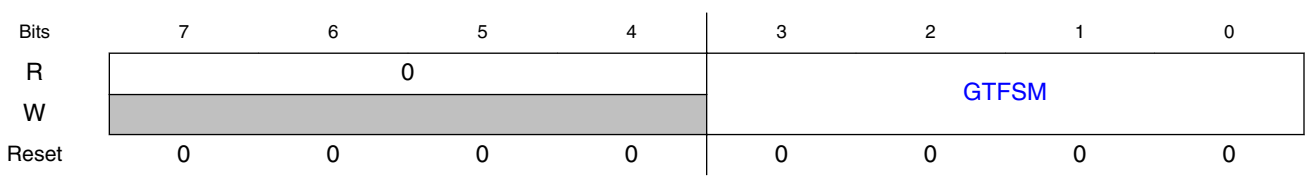
| Register | Offset |
|---|---|
| GATE3 | 0h |
| GATE2 | 1h |
| GATE1 | 2h |
| GATE0 | 3h |
| GATE7 | 4h |
| GATE6 | 5h |
| GATE5 | 6h |
| GATE4 | 7h |
| GATE11 | 8h |
| GATE10 | 9h |
| GATE9 | Ah |
| GATE8 | Bh |
| GATE15 | Ch |
| GATE14 | Dh |
| GATE13 | Eh |
| GATE12 | Fh |

#### 10.2.1.2.2 Function

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

#### 10.2.1.2.3 Diagram



#### 10.2.1.2.4 Fields

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3-0<br>GTFSM | GTFSM<br>Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug.<br>The hardware gate is maintained in a 16-state implementation, defined as:<br>0000b - The gate is unlocked (free).<br>0001b - The gate has been locked by processor 0.<br>0010b - The gate has been locked by processor 1.<br>0011b - The gate has been locked by processor 2.<br>0100b - The gate has been locked by processor 3.<br>0101b - The gate has been locked by processor 4.<br>0110b - The gate has been locked by processor 5.<br>0111b - The gate has been locked by processor 6.<br>1000b - The gate has been locked by processor 7.<br>1001b - The gate has been locked by processor 8.<br>1010b - The gate has been locked by processor 9.<br>1011b - The gate has been locked by processor 10.<br>1100b - The gate has been locked by processor 11.<br>1101b - The gate has been locked by processor 12.<br>1110b - The gate has been locked by processor 13.<br>1111b - The gate has been locked by processor 14. |

### 10.2.1.3 Reset Gate Read (RSTGT_R)

#### 10.2.1.3.1 Offset

| Register | Offset |
|---|---|
| RSTGT_R | 42h |

#### 10.2.1.3.2 Function

**NOTE**

This section and the following section, Reset Gate Write (RSTGT_W), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is read. Reset Gate Write (RSTGT_W) shows how the register fields appear when the register is written.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same domain to force the clearing of the specified gate(s). The required access pattern is:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte (RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte (RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.
3. Reads of the RSTGT location return information on the 2-bit state machine (RSTGT[RSTGSM]) that implements this function, the domain performing the reset (RSTGT[RSTGMS]), and the gate number(s) last cleared (RSTGT[RSTGTN]). Reads of the RSTGT register do not affect the secure reset finite state machine in any manner.

### 10.2.1.3.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ROZ | | RSTGSM | | RSTGMS | | | | RSTGTN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 10.2.1.3.4 Fields

| Field | Function |
|---|---|
| 15-14<br>ROZ | ROZ<br>This field always returns the value 0 when read. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13-12<br>RSTGSM | RSTGSM<br>Reset Gate Finite State Machine. Reads of the RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:<br>00b - Idle, waiting for the first data pattern write.<br>01b - Waiting for the second data pattern write.<br>10b - The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state.<br>11b - This state encoding is never used and therefore reserved. |
| 11-8<br>RSTGMS | RSTGMS<br>Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs.<br>The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information. |
| 7-0<br>RSTGTN | RSTGTN<br>Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write.<br>If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. |

## 10.2.1.4 Reset Gate Write (RSTGT_W)

### 10.2.1.4.1 Offset

| Register | Offset |
|---|---|
| RSTGT_W | 42h |

### 10.2.1.4.2 Function

This section describes how the Reset Gate register fields appear when the register is written. See Reset Gate Read (RSTGT_R) for the description of the register's overall operation and how the register fields appear when the register is read.

#### 10.2.1.4.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | RSTGDP | | | | | | | | RSTGTN | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| [1] | | | | | | | | | | | | | | | | |

1. 0 - Reset value is not applicable for writes.

#### 10.2.1.4.4 Fields

| Field | Function |
|---|---|
| 15-8<br>RSTGDP | RSTGDP<br>Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh. |
| 7-0<br>RSTGTN | RSTGTN<br>Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write.<br>If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. |

## 10.2.2 SEMA421 register descriptions

### 10.2.2.1 SEMA421 Memory map

SEMA421 base address: 4101_B000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Gate Register (GATE3) | 8 | RW | 00h |
| 1h | Gate Register (GATE2) | 8 | RW | 00h |
| 2h | Gate Register (GATE1) | 8 | RW | 00h |
| 3h | Gate Register (GATE0) | 8 | RW | 00h |
| 4h | Gate Register (GATE7) | 8 | RW | 00h |
| 5h | Gate Register (GATE6) | 8 | RW | 00h |
| 6h | Gate Register (GATE5) | 8 | RW | 00h |
| 7h | Gate Register (GATE4) | 8 | RW | 00h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | Gate Register (GATE11) | 8 | RW | 00h |
| 9h | Gate Register (GATE10) | 8 | RW | 00h |
| Ah | Gate Register (GATE9) | 8 | RW | 00h |
| Bh | Gate Register (GATE8) | 8 | RW | 00h |
| Ch | Gate Register (GATE15) | 8 | RW | 00h |
| Dh | Gate Register (GATE14) | 8 | RW | 00h |
| Eh | Gate Register (GATE13) | 8 | RW | 00h |
| Fh | Gate Register (GATE12) | 8 | RW | 00h |
| 42h | Reset Gate Read (RSTGT_R) | 16 | RO | 0000h |
| 42h | Reset Gate Write (RSTGT_W) | 16 | WO | See description. |

## 10.2.2.2 Gate Register (GATE0 - GATE15)

### 10.2.2.2.1 Offset

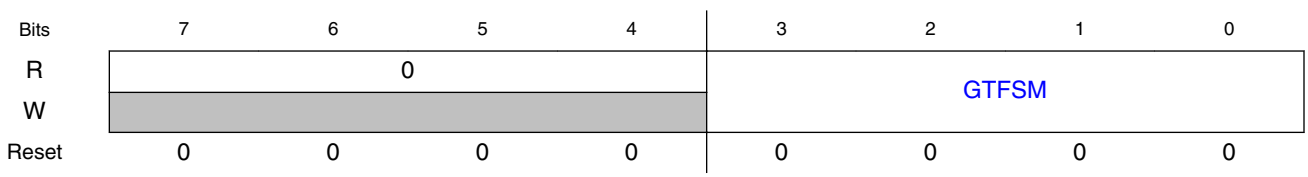
| Register | Offset |
|---|---|
| GATE3 | 0h |
| GATE2 | 1h |
| GATE1 | 2h |
| GATE0 | 3h |
| GATE7 | 4h |
| GATE6 | 5h |
| GATE5 | 6h |
| GATE4 | 7h |
| GATE11 | 8h |
| GATE10 | 9h |
| GATE9 | Ah |
| GATE8 | Bh |
| GATE15 | Ch |
| GATE14 | Dh |
| GATE13 | Eh |
| GATE12 | Fh |

#### 10.2.2.2.2 Function

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

#### 10.2.2.2.3 Diagram




#### 10.2.2.2.4 Fields

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3-0<br>GTFSM | GTFSM<br>Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug.<br>The hardware gate is maintained in a 16-state implementation, defined as:<br>0000b - The gate is unlocked (free).<br>0001b - The gate has been locked by processor 0.<br>0010b - The gate has been locked by processor 1.<br>0011b - The gate has been locked by processor 2.<br>0100b - The gate has been locked by processor 3.<br>0101b - The gate has been locked by processor 4.<br>0110b - The gate has been locked by processor 5.<br>0111b - The gate has been locked by processor 6.<br>1000b - The gate has been locked by processor 7.<br>1001b - The gate has been locked by processor 8.<br>1010b - The gate has been locked by processor 9.<br>1011b - The gate has been locked by processor 10.<br>1100b - The gate has been locked by processor 11.<br>1101b - The gate has been locked by processor 12. |

| Field | Function |
|---|---|
| | 1110b - The gate has been locked by processor 13.<br>1111b - The gate has been locked by processor 14. |

### 10.2.2.3 Reset Gate Read (RSTGT_R)

#### 10.2.2.3.1 Offset

| Register | Offset |
|---|---|
| RSTGT_R | 42h |

#### 10.2.2.3.2 Function

**NOTE**

This section and the following section, Reset Gate Write (RSTGT_W), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is read. Reset Gate Write (RSTGT_W) shows how the register fields appear when the register is written.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same domain to force the clearing of the specified gate(s). The required access pattern is:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte (RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte (RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (RSTGT[RSTGTN]) specifies the gate(s) to be

reset. This gate field can specify a single gate be cleared, or that all gates are to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.

3. Reads of the RSTGT location return information on the 2-bit state machine (RSTGT[RSTGSM]) that implements this function, the domain performing the reset (RSTGT[RSTGMS]), and the gate number(s) last cleared (RSTGT[RSTGTN]). Reads of the RSTGT register do not affect the secure reset finite state machine in any manner.

### 10.2.2.3.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ROZ | | RSTGSM | | RSTGMS | | | | RSTGTN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 10.2.2.3.4 Fields

| Field | Function |
|---|---|
| 15-14<br>ROZ | ROZ<br>This field always returns the value 0 when read. |
| 13-12<br>RSTGSM | RSTGSM<br>Reset Gate Finite State Machine. Reads of the RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:<br>00b - Idle, waiting for the first data pattern write.<br>01b - Waiting for the second data pattern write.<br>10b - The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state.<br>11b - This state encoding is never used and therefore reserved. |
| 11-8<br>RSTGMS | RSTGMS<br>Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs.<br>The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information. |
| 7-0<br>RSTGTN | RSTGTN<br>Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write.<br>If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. |

## 10.2.2.4 Reset Gate Write (RSTGT_W)

### 10.2.2.4.1 Offset

| Register | Offset |
|---|---|
| RSTGT_W | 42h |

### 10.2.2.4.2 Function

This section describes how the Reset Gate register fields appear when the register is written. See Reset Gate Read (RSTGT_R) for the description of the register's overall operation and how the register fields appear when the register is read.

### 10.2.2.4.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | RSTGDP | | | | | | | | RSTGTN | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |
| [1] | | | | | | | | | | | | | | | | |

1. 0 - Reset value is not applicable for writes.

### 10.2.2.4.4 Fields

| Field | Function |
|---|---|
| 15-8<br>RSTGDP | RSTGDP<br>Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh. |
| 7-0<br>RSTGTN | RSTGTN<br>Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write.<br>If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. |

## 10.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in Figure 10-2.

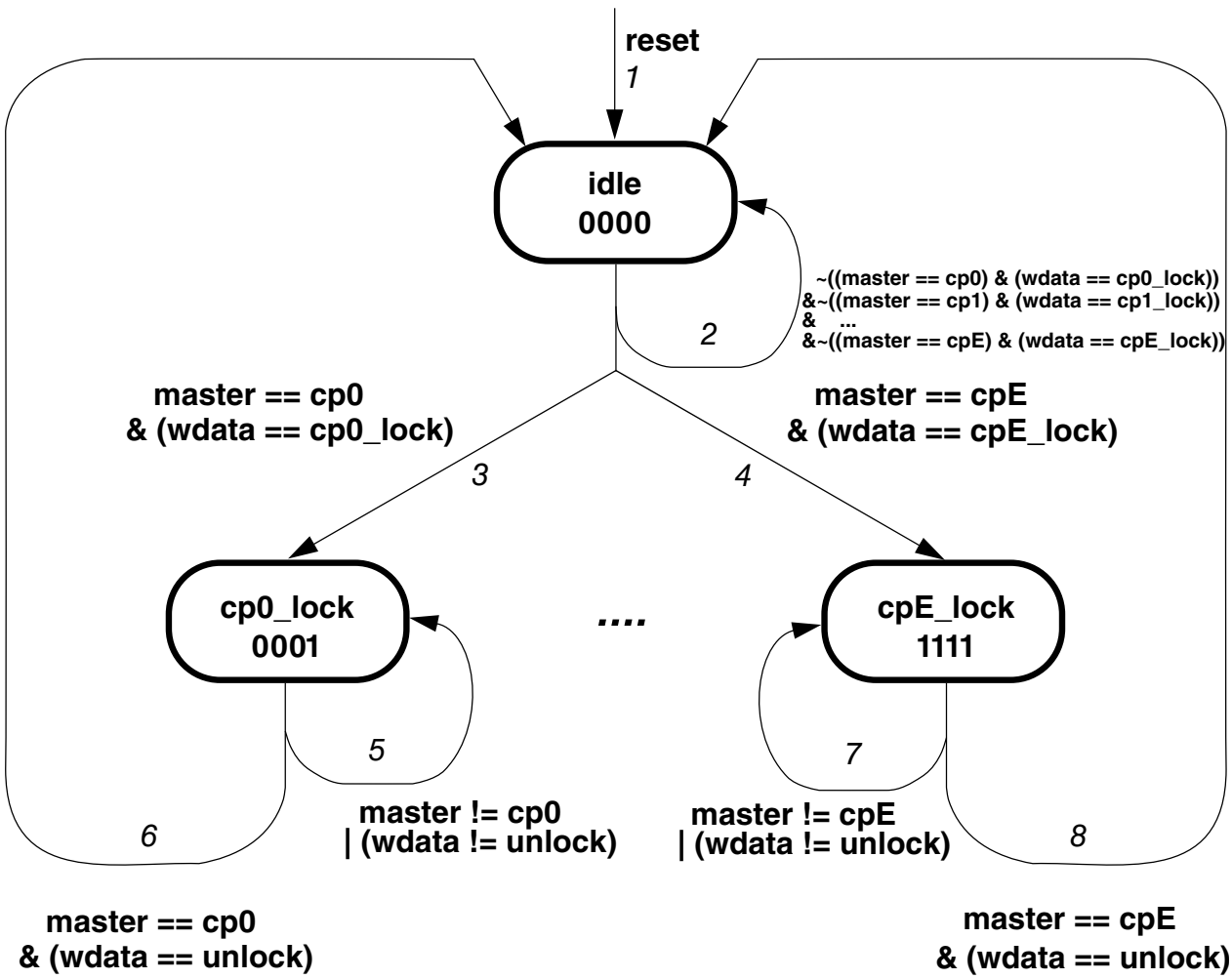



**Figure 10-2. SEMA42_GATEn state machine**

The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the Semaphores module.

The state transitions for SEMA42_GATEn are defined in Table 10-1.

**Table 10-1. SEMA42_GATEn state transitions**

| Current State | Next State | Transition | Description |
|---|---|---|---|
| – | idle | 1 | Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state. |
| idle | idle | 2 | Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state. |
| idle | cp0_lock | 3 | When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state. |
| idle | cpE_lock | 4 | When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state. |
| cp0_lock | cp0_lock | 5 | Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value. |
| cp0_lock | idle | 6 | The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs. |
| cpE_lock | cpE_lock | 7 | Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value. |
| cpE_lock | idle | 8 | The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs. |

The following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

# Chapter 11
# Memory-Mapped Divide and Square Root (MMDVSQ)

## 11.1 Introduction

ARM processor cores in the Cortex-M family implementing the ARMv6-M instruction set architecture do not include hardware support for integer divide operations. The affected processors include the Cortex-M0+ core. However, in certain deeply embedded application spaces, hardware support for this class of arithmetic operation (along with an unsigned square root function) is important to maximize system performance and minimize device power dissipation. Accordingly, the MMDVSQ module is included in select microcontrollers, to serve as a memory-mapped co-processor located in a special address space (within the system memory map) that is accessible only to the processor core.

The MMDVSQ module supports execution of the integer divide operations defined in the ARMv7-M instruction set architecture, plus an unsigned integer square root operation. The supported integer divide operations include 32/32 signed (SDIV) and unsigned (UDIV) calculations.

### 11.1.1 Features

The key features of the MMDVSQ include:

- Lightweight implementation of 32-bit integer divide and square root arithmetic operations
  - Supports 32/32 signed and unsigned divide (or remainder) calculations
  - Supports 32-bit unsigned square root calculations
- Simple programming model includes input data and result registers plus a control/status register
- Programming model interface optimized for activation from inline code or software library call

- "Fast Start" configuration minimizes the memory-mapped register write overhead
- Supports two methods to determine when result is valid, including software polling
- Configurable divide-by-zero response

• Pipelined design processes 2 bits per cycle with early termination exit for minimum execution time

## 11.1.2 Block diagram

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown in Figure 11-1. The MMDVSQ module's location as a memory-mapped co-processor is highlighted.

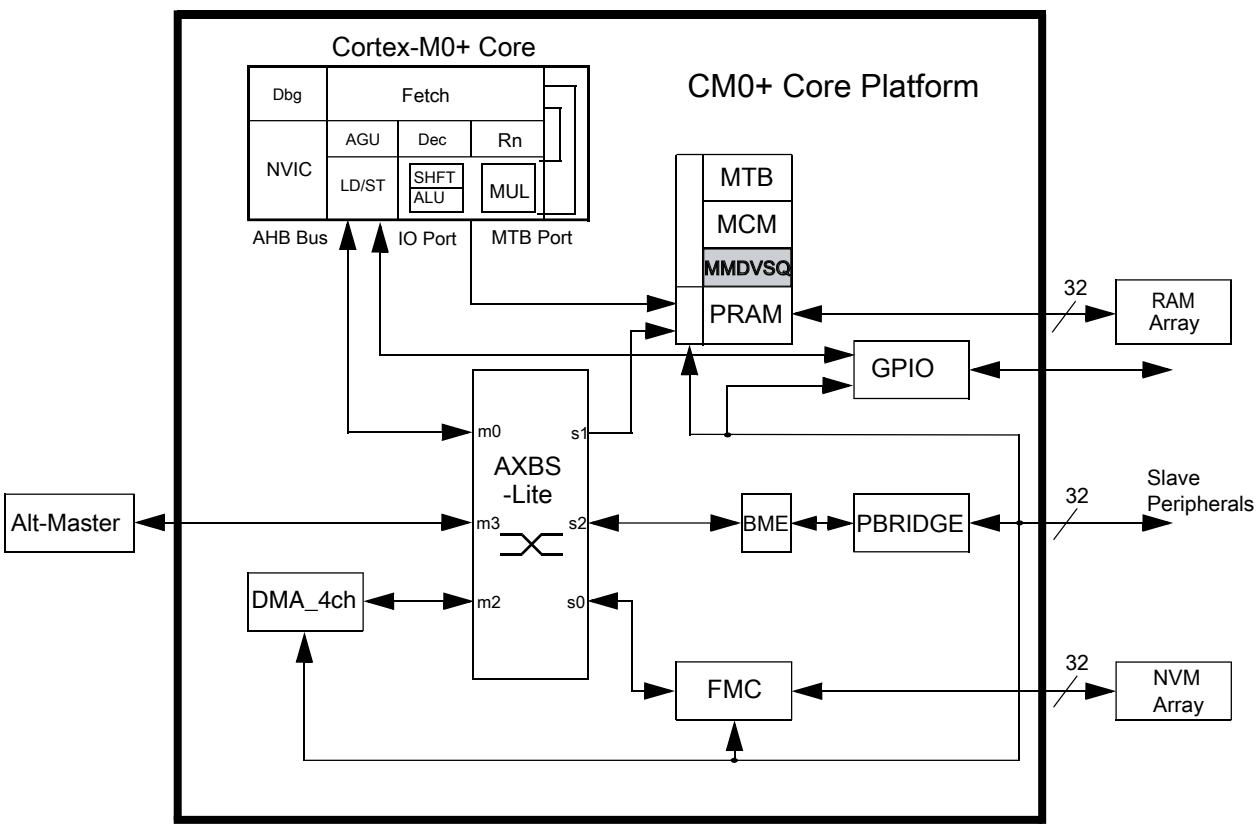



**Figure 11-1. Generic Cortex-M0+ Core Platform Block Diagram**

Next, a block diagram of the internal structure of the MMDVSQ module is presented. See Figure 11-2.

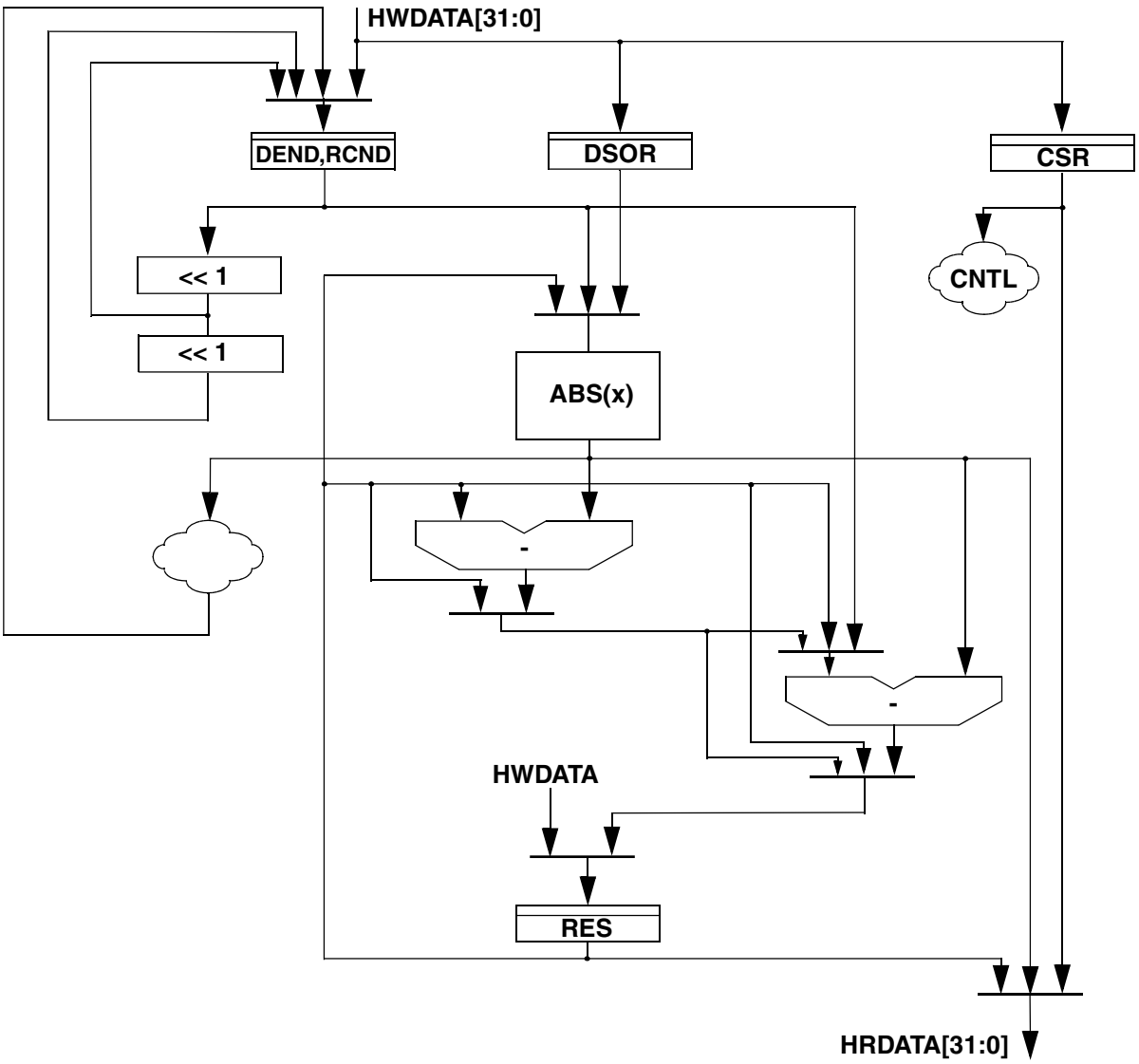



**Figure 11-2. MMDVSQ Block Diagram**

## 11.1.3 Modes of operation

The MMDVSQ module does not support any special modes of operation. As a memory-mapped device located on a crossbar slave AHB system bus port, MMDVSQ responds based strictly on memory addresses to its programming model.

All functionality associated with the MMDVSQ module resides in the core platform's clock domain; this includes its connections with the crossbar slave port. To minimize power dissipation, the design supports an architectural clock gate for the entire module, that is, the MMDVSQ is only clocked when responding to bus requests to its programming model or is busy performing a calculation.

## 11.2 External signal description

The MMDVSQ module does not directly support any external interfaces.

The internal interface includes a standard 32-bit AHB bus as shown in Figure 11-1.

## 11.3 Memory map and register definition

The MMDVSQ module supports a small number of program-visible registers used for passing input operands and retrieving the output result plus a configuration/status register.

The programming model occupies the first 20 bytes of a standard 4 Kb address slot. It can only be accessed via word-sized (32 bit) accesses. Attempted accesses using smaller data sizes, reading the write-only location or to reserved space are terminated with an error.

At any instant in time, the MMDVSQ can perform either a divide or square root calculation. The basic integer operations supported by the MMDVSQ are:

For divide:

```
MMDVSQ_RES = quotient  (MMDVSQ_DEND / MMDVSQ_DSOR)

MMDVSQ_RES = remainder (MMDVSQ_DEND % MMDVSQ_DSOR)
```

For square root:

```
MMDVSQ_RES = integer   (√MMDVSQ_RCND)
```

The register usage, based on the operation (divide, square root), is detailed in Table 11-1.

**Table 11-1. Register Usage = f(Divide, Square Root)**

| Register | Divide | Square Root | Description |
|---|---|---|---|
| Dividend (MMDVSQ_DEND) | Yes | No | Input dividend (numerator) for the divide |
| Divisor (MMDVSQ_DSOR) | Yes | No | Input divisor (denominator) for the divide |
| Control/Status (MMDVSQ_CSR) | Yes | Yes | Control for divide, status for divide and square root |
| Result (MMDVSQ_RES) | Yes | Yes | Output result |
| Radicand (MMDVSQ_RCND) | No | Yes | Input "square" data |

## 11.3.1 MMDVSQ register descriptions

### 11.3.1.1 MMDVSQ Memory map

MMDVSQ1 base address: F000_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Dividend Register (DEND) | 32 | RW | See description. |
| 4h | Divisor Register (DSOR) | 32 | RW | See description. |
| 8h | Control/Status Register (CSR) | 32 | RW | See description. |
| Ch | Result Register (RES) | 32 | RW | See description. |
| 10h | Radicand Register (RCND) | 32 | WO | See description. |

### 11.3.1.2 Dividend Register (DEND)

#### 11.3.1.2.1 Offset

| Register | Offset |
|---|---|
| DEND | 0h |

#### 11.3.1.2.2 Function

This register is loaded with the input dividend operand before a divide operation is initiated. The register is updated by the MMDVSQ hardware during the execution of a divide or square root calculation. Any memory access (read or write) of the DEND register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes.

#### 11.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIVIDEND | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIVIDEND | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 11.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DIVIDEND | Dividend<br>This is the input dividend operand for divide calculations. |

### 11.3.1.3 Divisor Register (DSOR)

#### 11.3.1.3.1 Offset

| Register | Offset |
|---|---|
| DSOR | 4h |

#### 11.3.1.3.2 Function

This register is loaded with the input divisor operand before a divide operation is initiated. If CSR[DFS] = 0, a write to this register inititates a divide operation. Any memory access (read or write) of the DSOR register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes. If a divide operation is inititated with DSOR = 0, the hardware signals a divide-by-zero condition and sets RES = 0 and CSR[DZ] = 1. If CSR[DZE] = 1, an attempted read of the RES result is error terminated.

#### 11.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIVISOR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIVISOR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 11.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DIVISOR | Divisor<br>This is the input divisor operand for divide calculations. |

### 11.3.1.4 Control/Status Register (CSR)

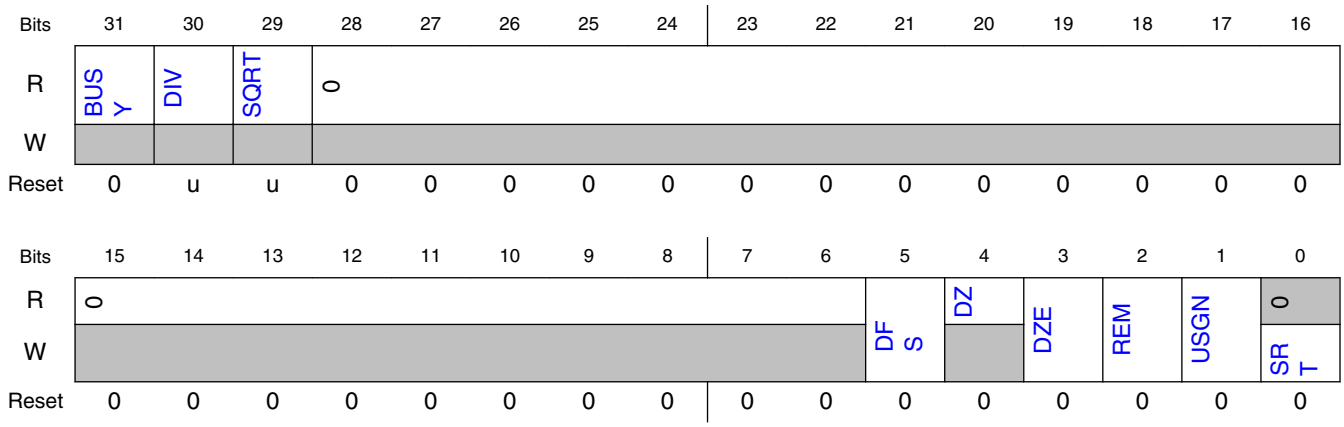
#### 11.3.1.4.1 Offset

| Register | Offset |
|---|---|
| CSR | 8h |

#### 11.3.1.4.2 Function

This register defines the operating configuration of divide operations and provides status information. The upper 3 bits provide busy status indicators, while the low-order byte defines the configuration for divide operations. The read-only status bits in CSR[31:29] are valid for both divide and square root operations; the configuration and status bit in CSR[5:0] are only valid for divides. A memory write access of the CSR register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes.

### 11.3.1.4.3 Diagram




### 11.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>BUSY | BUSY<br>This read-only bit is asserted when the MMDVSQ is performing a divide or square root. When an operation is initiated, the hardware sets this flag. It remains asserted until the operation completes and the hardware automatically clears the indicator. This bit can be used to poll the DVSQ's execution status. The combined CSR[BUSY, DIV, SQRT] indicators provide an encoded module status:<br>• If 0b001, then MMDVSQ is idle and the last calculation was a square root<br>• If 0b010, then MMDVSQ is idle and the last calculation was a divide<br>• If 0b101, then MMDVSQ is busy processing a square root calculation<br>• If 0b110, then MMDVSQ is busy processing a divide calculation<br><br>The remaining encodings of CSR[BUSY, DIV, SQRT] are reserved.<br>0b - MMDVSQ is idle<br>1b - MMDVSQ is busy performing a divide or square root calculation |
| 30<br>DIV | DIVIDE<br>Current or last operation was a divide. This read-only indicator bit signals if the current or last operation performed by the MMDVSQ was a divide.<br>0b - Current or last MMDVSQ operation was not a divide<br>1b - Current or last MMDVSQ operation was a divide |
| 29<br>SQRT | SQUARE ROOT<br>Current or last operation was a square root. This read-only indicator bit signals if the current or last operation performed by the MMDVSQ was a square root.<br>0b - Current or last MMDVSQ operation was not a square root<br>1b - Current or last MMDVSQ operation was a square root |
| 28-6<br>— | Reserved |
| 5<br>DFS | Disable Fast Start<br>The MMDVSQ supports 2 mechanisms for initiating a divide operation. The default mechanism is a "fast start" where a write to the DSOR register begins the divide. Alternatively, the start mechanism can begin after a write to the CSR register with CSR[SRT] set. The CSR[DFS] indicator selects the divide start mechanism. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - A divide operation is initiated by a write to the DSOR register<br>1b - A divide operation is initiated by a write to the CSR register with CSR[SRT] = 1 |
| 4<br>DZ | Divide-by-Zero<br>This read-only status indicator signals the last divide operation had a zero divisor, that is, DSOR = 0x0000_0000. For this case, RES is set to 0x0000_0000 and this indicator bit set. After a divide-by-zero operation, a read of the RES register returns either the zero result, or, if CSR[DZE] = 1, terminates the read with an error. The CSR[DZ] indicator is cleared by the hardware at the beginning of each operation.<br>0b - The last divide operation had a non-zero divisor, that is, DSOR != 0<br>1b - The last divide operation had a zero divisor, that is, DSOR = 0 |
| 3<br>DZE | Divide-by-Zero-Enable<br>This indicator configures the MMDVSQ's response to divide-by-zero calculations. If both CSR[DZ] and CSR[DZE] are set, then a subsequent read of the RES register is error terminated to signal the processor of the attempted divide-by-zero.<br>0b - Reads of the RES register return the register contents<br>1b - If CSR[DZ] = 1, an attempted read of RES register is error terminated to signal a divide-by-zero, else the register contents are returned |
| 2<br>REM | REMainder calculation<br>This indicator selects whether the quotient or the remainder is returned in the RES register. The combined CSR[REM] and CSR[USGN] bits define four possible divide operations:<br>• If CSR[REM, USGN] = 0b00, perform a signed divide, returning the quotient<br>• If CSR[REM, USGN] = 0b01, perform an unsigned divide, returning the quotient<br>• If CSR[REM, USGN] = 0b10, perform a signed divide, returning the remainder<br>• If CSR[REM, USGN] = 0b11, perform an unsigned divide, returning the remainder<br><br>0b - Return the quotient in the RES for the divide calculation<br>1b - Return the remainder in the RES for the divide calculation |
| 1<br>USGN | Unsigned calculation<br>This indicator selects whether a signed (default) or unsigned divide is performed. See the CSR[REM] description for the encoding of the four possible divide operations.<br>0b - Perform a signed divide<br>1b - Perform an unsigned divide |
| 0<br>SRT | Start<br>When written with a logical one and CSR[DFS] = 1, this flag initiates a divide operation. If written as a logical one with CSR[DFS] = 0, it is ignored. This bit always reads as a zero. The state of the register write data defines this bit's function.<br>0b - No operation initiated<br>1b - If CSR[DFS] = 1, then initiate a divide calculation, else ignore |

## 11.3.1.5 Result Register (RES)

### 11.3.1.5.1 Offset

| Register | Offset |
|---|---|
| RES | Ch |

#### 11.3.1.5.2 Function

This register is loaded with the result of the divide or square root calculation. It is updated by the MMDVSQ hardware at the completion of the calculation. When a square root operation is performed (on an unsigned 32-bit number), the result is limited to a 16-bit value with RES[31:16] = 0x0000. Any memory access (read or write) of the RES register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes and the new result written into the register.

#### 11.3.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 11.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>RESULT | Result<br>This is the output result for a divide or square root calculation. |

### 11.3.1.6 Radicand Register (RCND)

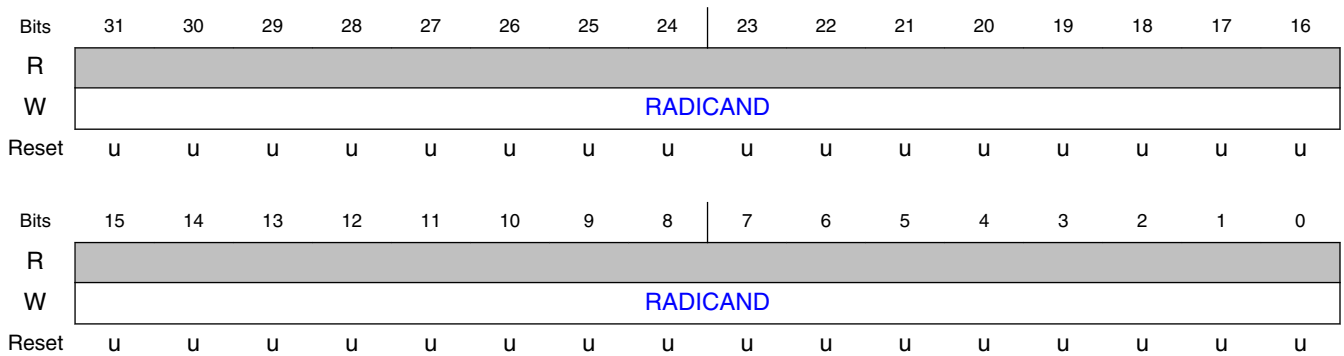
#### 11.3.1.6.1 Offset

| Register | Offset |
|---|---|
| RCND | 10h |

#### 11.3.1.6.2 Function

The write-only radicand register is loaded with the input "square" number. A memory write to the radicand register initiates a square root calculation. While the MMDVSQ module is busy performing a square root calculation, any memory write access to the RCND register causes the write access to be stalled (using wait states) until the square root calculation finishes. Any attempted read of the radicand register terminates with an error.

#### 11.3.1.6.3 Diagram




#### 11.3.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>RADICAND | Radicand<br>This is the input radicand for a square root calculation, that is, the input "square" number. |

## 11.4 Functional description

This section details the algorithms, execution times of the MMDVSQ, and the software interface to the module.

### 11.4.1 Algorithms

This section provides more details on the integer divide and square root algorithms.

## 11.4.1.1 Integer divide including special cases

### 11.4.1.1.1 Overview

The MMDVSQ module implements a "shift, test, and restore" radix-2 algorithm for unsigned integer divide operations. When performing a signed divide calculation, negative input operands are converted into 2's complement positive numbers first, an unsigned divide performed, and the sign of the results based on the input operand signs, namely:

- The sign of the remainder is the same as the sign of the dividend
- The quotient is negated if the signs of the dividend and divisor are different

The hardware implementation processes two bits per machine cycle and includes "early termination" logic where the execution time is data dependent, based on the magnitude of the positive dividend. See Table 11-4 for more execution time details.

### 11.4.1.1.2 Special case: Overflow

There is a single "special overflow case" affecting signed integer divides. If the dividend = 0x8000_0000 and the divisor = 0xFFFF_FFFF, the result of this ($-2^{31}/-1$) operation cannot be expressed as a 32-bit 2's complement number. For this case, the MMDVSQ exactly follows the ARM Cortex-Mx definition and returns 0x8000_0000 (the lower 32 bits of the $+2^{31}$ result) as the quotient with no indication of the overflow condition. If the remainder is selected as the output of this calculation, it returns 0x0000_0000.

### 11.4.1.1.3 Special case: Divide-by-Zero

For both signed and unsigned divides, if the divisor is zero, the MMDVSQ hardware detects this condition and the CSR[DZ] indicator set. The quotient result is forced to 0x0000_0000. If the remainder is selected as the output of this calculation, it also returns 0x0000_0000. Additionally, if CSR[DZE] = 1, then an attempted read of the Result register (RES) is error terminated to provide a simple mechanism to signal software of the divide-by-zero condition.

### 11.4.1.2 Integer square root

#### 11.4.1.2.1 Overview

The unsigned square root algorithm begins by creating a 32-bit "one-hot" bit vector signaling the highest power of four of the contents of the Radicand register (RCND). It then iterates through an algorithm involving magnitude comparisons of the RCND register versus the working result plus bit vector summation, conditional decrementing of the radicand, a 1-bit right shift of the result, and a 2-bit right shift of the one-hot bit vector.

Processing two bits of the radicand per cycle, the result register finishes with the integer portion of the square root calculation. The module includes early termination logic so that the execution time is data dependent, based on the magnitude of the input radicand. See Table 11-5 for more execution time details. Since both algorithms share common hardware structures, the incremental cost of the square root logic is an extremely small delta to the basic divide hardware.

The square root algorithm was exhaustively compared (that is, all $2^{32}$ possible input values) against the standard GNU C library implementation, which converts the unsigned integer input into a double-precision floating-point number, calculates the double-precision square root and then converts it back into an unsigned integer. Each input value calculated identical square root results.

#### 11.4.1.2.2 Square root using Q notation

Consider the use of Q notation for square root calculations returning fractional values. The following description is taken from http://en.wikipedia.org/wiki/Q_(number_format).

*Q is a fixed point number format where the number of fractional bits (and optionally the number of integer bits) is specified. For example, a Q15 number has 15 fractional bits; a Q1.14 number has 1 integer bit and 14 fractional bits. Q format is often used in hardware that does not have a floating-point unit and in applications that require constant resolution.*

*Q format numbers are (notionally) fixed point numbers (but not actually a number itself); that is, they are stored and operated upon as regular binary numbers (i.e. signed integers), thus allowing standard integer hardware/ALU to perform rational number calculations. The number of integer bits, fractional bits and the underlying word size are to be chosen by the programmer on an application-specific basis - the programmer's choices of the foregoing will depend on the range and resolution needed for the numbers. The machine itself remains oblivious to the notional fixed point representation being*

*employed - it merely performs integer arithmetic the way it knows how. Ensuring that the computational results are valid in the Q format representation is the responsibility of the programmer.*

*The Q notation is written as Qm.n, where:*

- *Q designates that the number is in the Q format notation - the Texas Instruments representation for signed fixed-point numbers (the "Q" being reminiscent of the standard symbol for the set of rational numbers).*
- *m is the number of bits set aside to designate the two's complement integer portion of the number, exclusive of the sign bit (therefore if m is not specified it is taken as zero).*
- *n is the number of bits used to designate the fractional portion of the number, i.e. the number of bits to the right of the binary point. (If n = 0, the Q numbers are integers - the degenerate case).*

*Note that the most significant bit is always designated as the sign bit (the number is stored as a two's complement number) in order to allow standard arithmetic-logic hardware to manipulate Q numbers. Representing a signed fixed-point data type in Q format therefore always requires m+n+1 bits to account for the sign bit. Hence the smallest machine word size required to accommodate a Qm.n number is m+n+1, with the Q number left justified in the machine word.*

*For a given Qm.n format, using an m+n+1 bit signed integer container with n fractional bits:*

- *its range is $[-2^m, 2^m - 2^{-n}]$*
- *its resolution is $2^{-n}$*

For the unsigned integer format used in the MMDVSQ's square root calculation, an u(nsigned)Qm.n notation requires m+n bits (m+n = 32) for the input radicand. An uQm.n format produces an uQ(m/2).(n/2) square root. As examples, consider the following tables involving the square root of 2 and square root of "pi" calculations. As expected, as the number of fractional bits (n) increases, the error between the calculated square root and the "actual" result decreases.

**Table 11-2. Square Root of 2 Calculations (√2 = 1.4142135623)**

| RCND [Hex] | RCND Q format | Results [Hex] | RES Q Format | Decimal | % Error |
|---|---|---|---|---|---|
| 0x0000_0002 | uQ32.00 | 0x0000_0001 | uQ16.00 | 1.0 | -29.289% |
| 0x0002_0000 | uQ16.16 | 0x0000_016A | uQ08.08 | 1.4140625 | -0.011% |
| 0x0200_0000 | uQ08.24 | 0x0000_16A0 | uQ04.12 | 1.4140625 | -0.011% |
| 0x2000_0000 | uQ04.28 | 0x0000_5A82 | uQ02.14 | 1.4141845703 | -0.002% |
| 0x8000_0000 | uQ02.30 | 0x0000_B504 | uQ01.15 | 1.4141845703 | -0.002% |

**Table 11-3. Square Root of Pi Calculations (√Pi = 1.7724538509)**

| RCND [Hex] | RCND Q format | Results [Hex] | RES Q Format | Decimal | % Error |
|---|---|---|---|---|---|
| 0x0000_0003 | uQ32.0 | 0x0000_0001 | uQ16.00 | 1.0 | -43.581% |
| 0x0003_243F | uQ16.16 | 0x0000_01C5 | uQ08.08 | 1.76953125 | -0.165% |
| 0x0324_3F6A | uQ08.24 | 0x0000_1C5B | uQ04.12 | 1.772216769 | -0.013% |
| 0x3243_F6A8 | uQ04.28 | 0x0000_716F | uQ02.14 | 1.7723999023 | -0.003% |
| 0xC90F_DAA0 | uQ02.30 | 0x0000_E2DF | uQ01.15 | 1.7724304199 | -0.001% |

The application of the Q notation for square root calculations provides a powerful extension for these types of fractional numeric computations using fixed-point integer processing hardware.

## 11.4.2 Execution times

The MMDVSQ module includes early termination logic to finish both divide and square root calculations as quickly as possible, based on the magnitude of the input operand. Accordingly, the execution time for the calculations is data dependent as defined in Table 11-4 and Table 11-5. In this context, the execution time is defined from the register write to initiate the calculation until the result register has been updated and available to read. Stated differently, it represents the time CSR[BUSY] is asserted for a given calculation. In the following two tables, "x" signals a bit with a don't care value.

**Table 11-4. Divide Execution Times**

| CSR[USGN]<br>? DEND[31:0] // unsigned divide<br>: abs(DEND[31:0]) // signed divide | Execution Time with CSR[BUSY] = 1 [cycles] |
|---|---|
| (01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 17 |
| 00(01,1x)_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 16 |
| 0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 15 |
| 0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 14 |
| 0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx | 13 |
| 0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx | 12 |
| 0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx | 11 |
| 0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx | 10 |
| 0000_0000_0000_0000__(01,1x)xx_xxxx_xxxx_xxxx | 9 |
| 0000_0000_0000_0000__00(01,1x)_xxxx_xxxx_xxxx | 8 |
| 0000_0000_0000_0000__0000_(01,1x)xx_xxxx_xxxx | 7 |
| 0000_0000_0000_0000__0000_00(01,1x)_xxxx_xxxx | 6 |
| 0000_0000_0000_0000__0000_0000_(01,1x)xx_xxxx | 5 |
| 0000_0000_0000_0000__0000_0000_00(01,1x)_xxxx | 4 |

*Table continues on the next page...*

**Table 11-4. Divide Execution Times (continued)**

| CSR[USGN]<br>? DEND[31:0] // unsigned divide<br>: abs(DEND[31:0]) // signed divide | Execution Time with CSR[BUSY] = 1 [cycles] |
|---|---|
| 0000_0000_0000_0000__0000_0000_0000_(01,1x)xx | 3 |
| 0000_0000_0000_0000__0000_0000_0000_00(01,1x) | 2 |
| 0000_0000_0000_0000__0000_0000_0000_0000 | 1 |

**Table 11-5. Square Root Execution Times**

| RCND[31:0] | Execution Time with CSR[BUSY] = 1 [cycles] |
|---|---|
| (01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 17 |
| 00(01,1x)_xxxx_xxxx_xxxx_x_xxx_xxxx_xxxx_xxxx | 16 |
| 0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 15 |
| 0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx | 14 |
| 0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx | 13 |
| 0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx | 12 |
| 0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx | 11 |
| 0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx | 10 |
| 0000_0000_0000_0000__(01,1x)xx_xxxx_xxxx_xxxx | 9 |
| 0000_0000_0000_0000__00(01,1x)_xxxx_xxxx_xxxx | 8 |
| 0000_0000_0000_0000__0000_(01,1x)xx_xxxx_xxxx | 7 |
| 0000_0000_0000_0000__0000_00(01,1x)_xxxx_xxxx | 6 |
| 0000_0000_0000_0000__0000_0000_(01,1x)xx_xxxx | 5 |
| 0000_0000_0000_0000__0000_0000_00(01,1x)_xxxx | 4 |
| 0000_0000_0000_0000__0000_0000_0000_(01,1x)xx | 3 |
| 0000_0000_0000_0000__0000_0000_0000_00(01,1x) | 2 |
| 0000_0000_0000_0000__0000_0000_0000_0000 | 2 |

## 11.4.3 Software interface

The programming model of the MMDVSQ is organized to be similar to the input arguments passed to software libraries for integer divide and square root functions.

### 11.4.3.1 Operation activation and result retrieval

The MMDVSQ supports 2 mechanisms for initiating a divide operation:

- The default mechanism is a "fast start" where a write to the DSOR register begins the divide.
- Alternatively, the start mechanism can begin after a write to the CSR register with the CSR[SRT] set.

The CSR[DFS] indicator selects the divide start mechanism.

```
if CSR[DFS] = 0
   then a divide is initiated by a write to the DSOR register
   else a divide is initiated by a write to the CSR register with CSR[SRT] = 1
```

A square root calculation is initiated by a write to the RCND register.

For both divide and square root calculations, the result of the operation is retrieved by reading the RES register. A memory read of this register while the calculation is still being performed causes the access to be stalled via the insertion of bus wait states until the new result is loaded into the register. Note a stalled bus cycle cannot be interrupted, so if system interrupt latency is a concern, the processor should execute a simple wait loop, for example, polling CSR[BUSY], before reading the RES register. This code construct is fully interruptible, so interrupt latency is minimized.

### 11.4.3.2 Context save and restore

Given that multiple memory-mapped register accesses are needed for each divide and square root calculation, interrupts may occur during the required sequence of operations. As a result, the MMDVSQ's programming model can be saved at entry to an interrupt service routine (ISR) and then restored when redispatching to the interrupted task.

The module's context can be saved by reading the DEND, DSOR, CSR, and RES registers and storing them as part of the task state. There is one special consideration for the task state save. If the last calculation was a zero divide and the divide-by-zero enable is set (CSR[DZE] = 1), then a read of the RES register is error terminated. To avoid a zero-divide error termination during a context save, the following sequence can be used:

1. Read DEND, DSOR, and CSR registers and save the values as part of the task state.
2. Clear CSR[DZE].
3. Read the RES register and save its value as part of the task state.

When restoring the context, special care must be taken to not initiate another divide calculation. Specifically, CSR[DFS] must be set first before reloading the DEND and DSOR registers. For example, the following sequence can be used for the context reload:

1. Write 0x0000_0020 to the CSR to disable the fast start mechanism.
2. Reload DEND, DSOR, CSR, and RES registers from the saved state.

Since the original context save of the control/status register is guaranteed to have CSR[SRT] = 0, there is no divide operation initiated when this register is reloaded in step 2.

# Chapter 12
# ARM Core Debug

## 12.1 Introduction

The debug system of this device is based on the ARM CoreSight™ architecture, and is configured to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

The MCU has one CM4F core for customer application use (Core0), and one CM0+ core mainly for wireless application (Core1).

Debug provides register and memory accessibility from the external debugger interface, basic run/halt control. The CM4F core implements full trace ability, FPB function and 4 watch points. The CM0+ core supports ARM's Micro Trace Buffer (MTB) capability to provide simple program trace and has 4 breakpoints and 2 watch points. Three debug interfaces are supported:

- IEEE 1149.1 JTAG
- Multi-drop Serial Wire Debug (SWD) debug interface.
- ARM Real-Time Trace Interface

The debug interfaces provide the capability for debugger tools to interface to either CPU.

## 12.2 Debug system block diagram

The following diagram shows the ARM CoreSight™ debug configuration for this device.

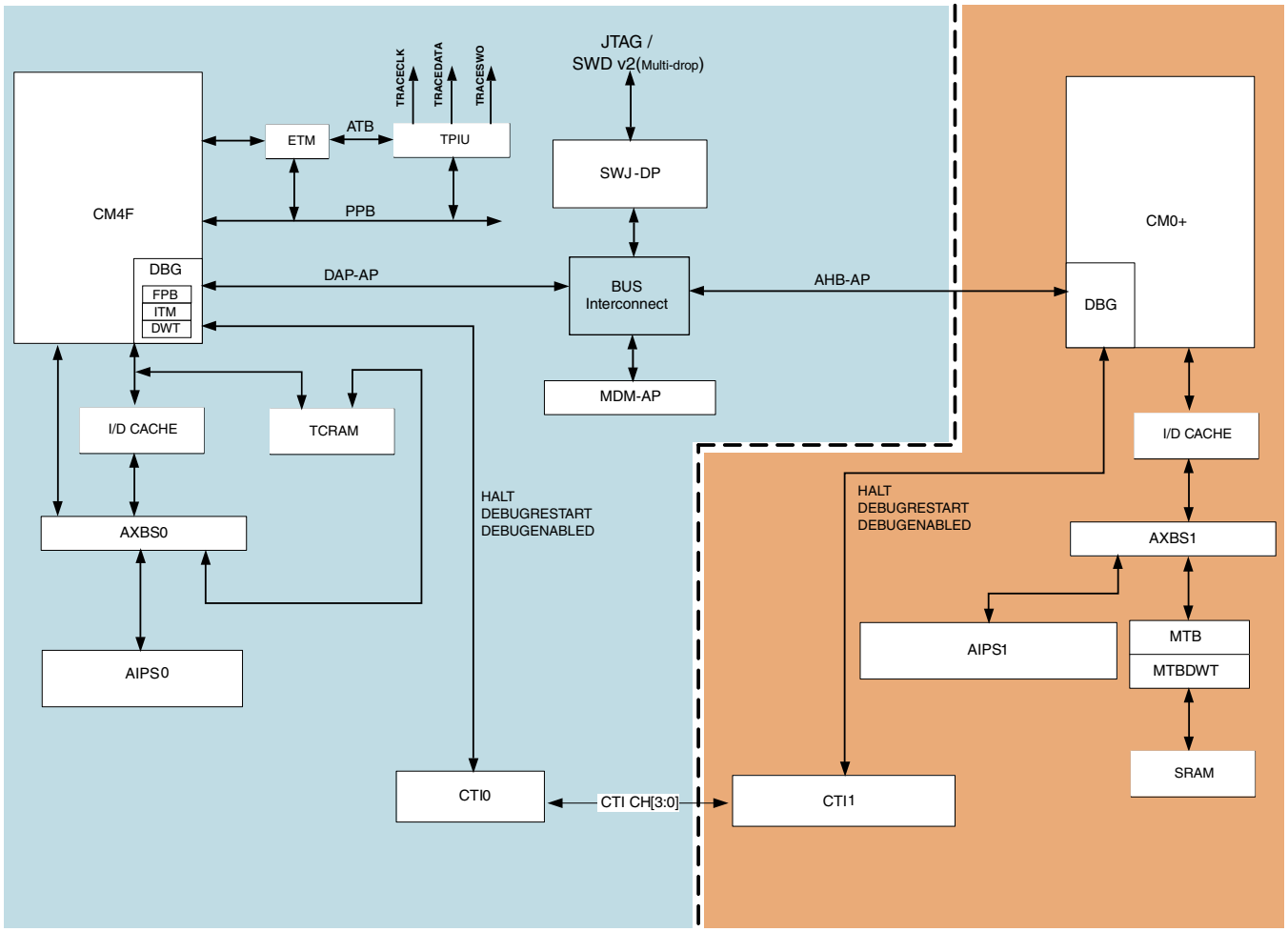



**Figure 12-1. Debug and trace block diagram**

## 12.2.1 Debug components

**Table 12-1. Debug components**

| Core | Module | Description |
|---|---|---|
| CM4F and CM0+ | SWJ-DP | Debug Port with support for SWD, JTAG |
| | CTI | Cross-triggering is achieved via the ARM-based Cross Trigger Interface (CTI0 and CTI1), which allows events on one Core to halt/start code execution and enable debug on the other Core, and vice versa. Each of the CTI modules has 4 output channels and 4 input channels:<br>• Each CTI can configure individual debug events of its Core (DebugStart, CPU halt etc) to 1 of 4 output channels. |

*Table continues on the next page...*

## Table 12-1. Debug components (continued)

| Core | Module | Description |
|---|---|---|
| | | • Each CTI can configure any of the input channels to trigger a CPU halt, Debug Enable, debug re-start.<br>• Thus, CTI0's 4 output channels are connected to CTI1's 4 input channels and vice versa |
| | MDM-AP | Provides centralized control and status registers for an external debugger to control the device. |
| CM4F | DAP-AP | |
| | ETM(Embedded Trace Macrocell) | ETMv3.5 Architecture |
| | ITM | S/W Instrumentation Messaging + Simple Data Trace Messaging + Watchpoint Messaging |
| | DWT (Data and Address Watchpoints) | 4 data and address watchpoints |
| | FPB (Flash Patch and Breakpoints) | The FPB implements hardware breakpoints and patches code and data from code space to system space.<br>The FPB unit contains two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space.<br>The FPB also contains six instruction comparators for matching against instruction fetches from Code space, and remapping to a corresponding area in System space. Alternatively, the six instruction comparators can individually<br>configure the comparators to return a Breakpoint Instruction (BKPT) to the processor core on a match, so providing hardware breakpoint capability. |
| | TPIU (Trace Port Inter face Unit) | Synchronous Mode (5-pin) = TRACE_D[3:0] + TRACE_CLKOUT<br>Synchronous Mode (3-pin) = TRACE_D[1:0] + TRACE_CLKOUT<br>Asynchronous Mode (1-pin) = TRACE_SWO (available on<br>JTAG_TDO) |
| CM0+ | AHB-AP | The AHB-AP2 module connects to the CM0+ core via a private AHB bus, allowing the debugger tool to set up watch points, breakpoints, instruction stepping, halting/ starting CPU execution |
| | MTB | Provides a simple execution trace capability for the Cortex-M0+ processor |

*Table continues on the next page...*

**Table 12-1. Debug components (continued)**

| Core | Module | Description |
|---|---|---|
| | MTBDWT | Supports triggering via TSTART and TSTOP control functions |

**Table 12-2. Debug access ports and AP numbers**

| Debug access ports | AP number |
|---|---|
| CM4F DAP-AP | AP 0 |
| MDM-AP | AP 1 |
| CM0+ AHB-AP | AP 2 |

### 12.2.2 JTAG-to-SWD change sequence

The following are JTAG to SWD change sequence:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111_1001_1110_0111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

**NOTE**

See ARM documentation for the CoreSight DAP Lite for the restrictions.

## 12.3 Debug port pin descriptions

The debug port pins default after POR to their SWD functionality.

**Table 12-3. Debug pin description**

| Pin name | Type | Description |
|---|---|---|
| JTAG_TCK / SWD_CLK | Input | JTAG Test Clock / Serial Wire Clock – This pin is the clock for debug logic when in the Serial Wire Debug mode. This pin is pulled down internally. |
| JTAG_TMS / SWD_DIO | Input / Output | JTAG Test Mode Select/ Serial Wire Data Input / Output – The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally. |
| JTAG_TDI | Input | JTAG Test Data Input |
| JTAG_TDO/ TRACE_SWO | Output | JTAG Test Data Output / Trace output over a single pin |
| TRACE_CLK_OUT | Output | Trace clock out. |
| TRACE_DATA0~3 | Output | Trace data output |

## 12.4 System TAP connection

The system JTAG controller is connected in parallel to the ARM TAP controller. The system JTAG controller IR codes overlay the ARM JTAG controller IR codes without conflict. Refer to the IR codes table for a list of the available IR codes. The output of the TAPs (TDO) are muxed based on the IR code which is selected. This is fully JTAG compliant and appears to the JTAG chain as a single TAP. At power on reset, NXP's IDCODE (IR=4'b0000) is selected.

**Table 12-4. JTAG Instructions**

| Instruction | Code[3:0] | Instruction summary |
|---|---|---|
| IDCODE | 0000 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 0010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 0011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 0100 | Selects boundary scan register while applying preloaded values to output pins and asserting functional reset |
| HIGHZ | 1001 | Selects bypass register while three-stating all output pins and asserting functional reset |
| CLAMP | 1100 | Selects bypass register while applying preloaded values to output pins and asserting functional reset |
| ARM_IDCODE | 1110 | ARM JTAG-DP Instruction |
| BYPASS | 1111 | Selects bypass register for data operations |
| Factory debug reserved | 0101, 0110, 0111, 1000 | Intended for factory debug only |
| ARM JTAG-DP Reserved | 1000, 1010, 1011 | These instructions will go the ARM JTAG-DP controller. Refer to ARM JTAG-DP documentation for more information on these instructions. |

## 12.5 JTAG/SWD status and control registers

Through the ARM Debug Access Port (DAP), the debugger has access to the status and control elements, implemented as registers on the DAP bus as shown in the following figure.

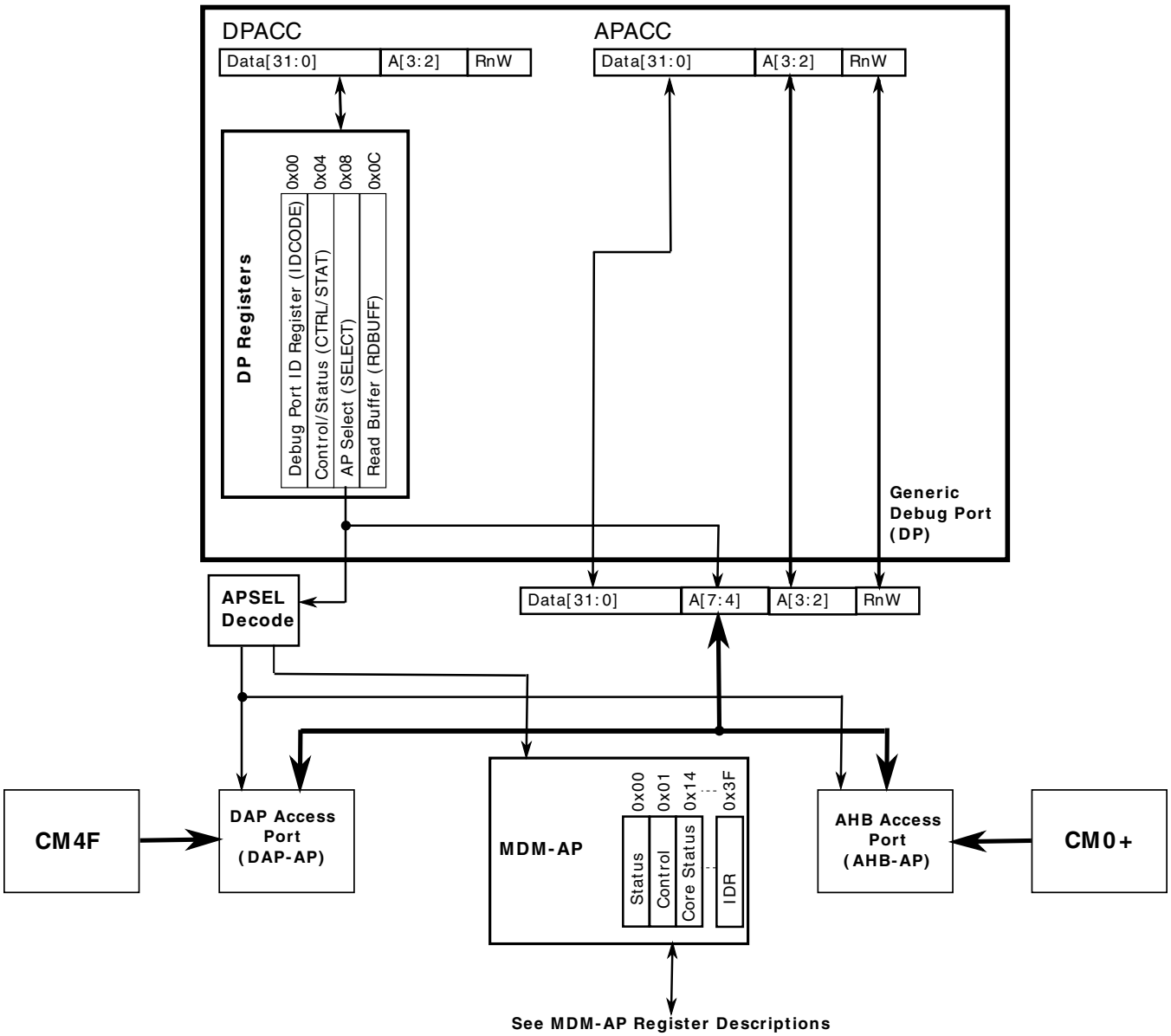



**Figure 12-2. MDM AP addressing**

These registers provide additional control and status for low power mode recovery and typical run-control scenarios. The status register bits also provide a means for the debugger to get updated status of the two cores, without having to initiate a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

It is important to note that these DAP control and status registers are not memory-mapped within the system memory map, and are only accessible via the Debug Access Port using SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in the following table.

**Table 12-5. MDM-AP register summary**

| Address | Register | Description |
|---|---|---|
| 0x0100_0000 | Status | See MDM-AP Status Register |
| 0x0100_0004 | Control | See MDM-AP Control Register |
| 0x0100_0050 | Core Status | See MDM-AP Core Status Register |
| 0x0100_0054 | Core Clock Status | See MDM-AP Core Clock Status Register |
| 0x0100_0058 | System Clock Source Status | See MDM-AP System Clock Source Status Register |
| 0x0100_005C | System Power Status | See MDM-AP System Power Status Register |
| 0x0100_0070 ~ 0x0100_0080 | PCC0_CGC0~ PCC0_CGC4 | See MDM-AP PCC0_CGC0~4 Register |
| 0x0100_0084 ~ 0x0100_0094 | PCC1_CGC0~ PCC1_CGC4 | See MDM-AP PCC1_CGC0~4 Register |
| 0x0100_00FC | Identification Register | Read-only identification register that always reads as 0x001C_0040 |

## 12.5.1 MDM-AP status registers

**Table 12-6. MDM-AP status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0 | Flash Mass Erase Acknowledge | The Flash Mass Erase Acknowledge bit is cleared after a Power-On Reset (POR) or when there is a new mass erase request before acknowledged. The Flash Mass Erase Acknowledge is set after Flash control logic has started the mass erase operation. When mass erase is disabled (via MEEN and SEC settings), an erase request due to setting of Flash Mass Erase in Progress bit is not acknowledged. If the system is not in reset status, the Flash Mass Erase is not acknowledged,either. |
| 1 | Flash Ready | Indicates Flash has been initialized and the debugger can be configured, even if the system is continuing to be held in reset by the debugger. |
| 2 | System Security | Indicates the security state. When secure, the debugger does not have access to the system bus or any memory-mapped peripherals. |
| 3 | System Reset | Indicates the system reset state.<br>0 System is in reset.<br>1 System is not in reset. |
| 4 | FSLACC | Flash security -Factory Security Level Access, see details from FSEC[FSLACC] in FTFE chapter |

*Table continues on the next page...*

## Table 12-6. MDM-AP status register assignments (continued)

| Bit | Name | Description |
|---|---|---|
| | | 0 indicates Factory access denied.<br>1 indicates Factory access granted |
| 5 | Mass Erase Enable | Indicates if the MCU can be mass erased or not<br>0 Mass erase is disabled.<br>1 Mass erase is enabled. |
| 6 | Backdoor Access Key Enable | Indicates if the MCU has the backdoor access key enabled.<br>0 Disabled<br>1 Enabled |
| 7 | LP Stop Mode Enabled | Decode of MSMC_SMCx_PMCTRL[STOPM] field to indicate that VLPS, LLS, or VLLSx are the selected power mode for the next time that the ARM Core enters Deep Sleep. This bit is used for debug operations in which Run to VLPS is attempted. Per the debug definition, the system actually enters the Stop state. A debugger should interpret a deep sleep indication (with SLEEPDEEP and SLEEPING asserted), in conjunction with LP Enabled bit asserted, as the debugger-VLPS status indication.<br>0 Low Power Stop Mode is not enabled.<br>1 Low Power Stop Mode is enabled. |
| 8 | Very Low Power Mode | Indicates that the current power mode is VLPS. This bit is not ‘sticky’ and should always represent whether VLPS is enabled or not. This bit is used to throttle SWD_CLK frequency up/down. |
| 9 | LLS Mode Exit | Indicates an exit from LLS mode has occurred. The debugger will lose communication while the system is in LLS (including access to this register). After communication is re-established, this bit indicates that the system is in LLS. Because the debug modules held their state during LLS, they do not need to be reconfigured. This bit is set during the LLS recovery sequence, and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in MDM AP Control register. |
| 10 | VLLSx Modes Exit | This bit indicates an exit from VLLSx mode has occurred. The debugger will lose communication while the system is in VLLSx (including access to this register). After communication is reestablished, this bit indicates that the system is in VLLSx. Because the debug modules lose their state information during VLLSx modes, they need to be reconfigured. This bit is set during the VLLSx recovery sequence, and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in MDM AP Control register. |
| 11 – 29 | Reserved for future use | Always read 0. |
| 30 | CM0+ Debug Restart Done | Indicates that the CM0+ debug restart procedure done<br>Set after CM0+ Debug Restart bit in MDM-AP Control Register is set |
| 31 | CM4F Debug Restart Done | Indicates that the CM4F debug restart procedure done<br>Set after CM4F Debug Restart bit in MDM-AP Control Register is set |

## 12.5.2 MDM-AP Control Register

**Table 12-7. MDM-AP control register assignments**

| Bit | Name | Secure[1] | Description |
|---|---|---|---|
| 0 | Flash Mass Erase in Progress | Y | Set to cause mass erase. Cleared by a Power-On Reset (POR) or when the mass erase is finished. When mass erase is disabled (via MEEN and SEC settings), the erase request does not occur and the Flash Mass Erase in Progress bit continues to assert until the next system reset. NOTE: When the reset pin has been disabled and security has been enabled by means of the FSEC register, a mass erase can be performed only by setting both the mass erase and system reset request bits in the MDM-AP register. |
| 1 | Debug Disable | N | Set to disable debug. Clear to allow debug operation. When set, it overrides the C_DEBUGEN bit within the DHCSR and force disables Debug logic. |
| 2 | Debug Request | N | Set to force the core to halt. If the core is in a Stop or Wait mode, then Debug Request bit can be used to wake the core to a halted state. |
| 3 | System Reset Request | Y | Set to force a system reset. The system remains held in reset until System Reset Request bit is cleared. |
| 4 | Core Hold Reset | N | Configuration bit to control core operation at the end of system reset sequencing.<br>0 Normal operation: At the end of system reset sequencing, release the core from reset (along with the rest of the system).<br>1 Suspend operation: At the end of reset sequencing, hold the core in reset. After the system enters this suspended state, clearing this control bit immediately releases the core from reset and CPU operation begins. |
| 5 | Debug Hold Request (DBGHOLDREQ) | N | Set to request the system to be held for any next system reset |
| 6 | Debug Hold Acknowledge (DBGHOLDACK) | N | Set to release system from hold after Debug Hold Request |
| 7 | LLS, VLLSx Status Acknowledge | N | Set this bit to acknowledge that the DAP LLS and VLLS Status bits have been read. This acknowledge automatically clears the status bits. This bit is used by the debugger to clear the sticky LLS and VLLSx mode entry status bits. This bit is asserted and cleared by the debugger. |
| 8 | Reserved for future use | N | |
| 9 | Inhibit Sleep | N | 1 = Device enters pseudo-sleep mode when debugger is attached<br>0 = Device enters full stop mode, and ignores attached debugger |
| 10-29 | Reserved for future use | N | |
| 30 | CM0+ Debug Restart | N | If CM0+ in halt state, set to bring CM0+ out of halt state after restart |
| 31 | CM4F Debug Restart | N | If CM4F in halt state, set to bring CM4F out of halt state after restart |

1. Register bit is writable in secure mode.

## 12.5.3 MDM-AP Core Status Register

**Table 12-8. MDM-AP Core Status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0 | CM4F DEEPSLEEP Status | Set to indicate CM4F in DEEPSLEEP mode |
| 1 | CM4F SLEEP Status | Set to indicate CM4F in SLEEP mode |
| 2 | CM4F Reset Status | Set to indicate CM4F in Reset state |
| 3 | CM4F CPO Status | Set to indicate CM4F in Compute operation mode(CPO) |
| 4-6 | Reserved for future use | Always read 0 |
| 7 | CM4F Halted Status | Set to indicate CM4F in Halt state |
| 8 | CM0+ DEEPSLEEP Status | Set to indicate CM0+ in DEEPSLEEP mode |
| 9 | CM0+ SLEEP Status | Set to indicate CM0+ in SLEEP mode |
| 10 | CM0+ Reset Status | Set to indicate CM4F in Reset state |
| 11 | CM0+ CPO Status | Set to indicate CM0+ in Compute operation mode (CPO) |
| 12-14 | Reserved for future use | Always read 0 |
| 15 | CM0+ Halted Status | Set to indicate CM0+ in Halt state |
| 16-31 | Reserved for future use | Always read 0 |

## 12.5.4 MDM-AP Core Clock Status Register

**Table 12-9. MDM-AP Core Clock Status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0 | CM4F Platform Clock Status | Set to indicate CM4F platform clock is enabled<br>Clear to indicate CM4F platform clock is disabled |
| 1 | CM4F Bus Clock Status | Set to indicate CM4F Bus clock is enabled<br>Clear to indicate CM4F Bus clock is disabled |
| 2 | CM4F Slow Clock Status | Set to indicate CM4F Slow clock is enabled<br>Clear to indicate CM4F Slow clock is disabled |

*Table continues on the next page...*

**Table 12-9. MDM-AP Core Clock Status register assignments (continued)**

| Bit | Name | Description |
|---|---|---|
| 3 | CM4F External Clock Status | Set to indicate Flexbus clock is enabled<br>Clear to indicate Flexbus clock is disabled |
| 4-7 | Reserved for future use | Always read 0 |
| 8 | CM0+ Platform Clock Status | Set to indicate CM0+ platform clock is enabled<br>Clear to indicate CM0+ platform clock is disabled |
| 9 | CM0+ Bus Clock Status | Set to indicate CM0+ bus clock is enabled<br>Clear to indicate CM0+ bus clock is disabled |
| 10 | CM0+ Slow Clock Status | Set to indicate CM0+ slow clock is enabled<br>Clear to indicate CM0+ slow clock is disabled |
| 11-31 | Reserved for future use | Always read 0 |

## 12.5.5 MDM-AP System Clock Status Register

**Table 12-10. MDM-AP System Clock Status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-3 | DIVSLOW | Indicates DIVSLOW value |
| 4-7 | DIVBUS | Indicates DIVBUS value[1] |
| 8-11 | DIVEXT | Indicates DIVEXT value[1] |
| 12-15 | Reserved | Always read 0 |
| 16-19 | DIVCORE | Indicate DIVCOR value[1] |
| 20-23 | System Clock Source Status | 0010 RF OSC<br>0011 SIRC<br>0100 FIRC<br>0101 RTC OSC<br>0110 LPFLL |
| 24 | FIRC ON Status | Set to indicate FIRC clock alive |
| 25 | SIRC ON Status | Set to indicate SIRC clock alive |
| 26 | OSC ON Status | Set to indicate OSC clock alive |
| 27 | RTC32K ON Status | Set to indicate RTC32K clock alive |
| 28 | LPFLL ON Status | Set to indicate LPFLL clock alive |
| 29 | Reserved | Always read 0 |
| 30 | LPO_1KHz ON Status | Set to indicate LPO_1KHz clock alive |

*Table continues on the next page...*

**Table 12-10. MDM-AP System Clock Status register assignments (continued)**

| Bit | Name | Description |
|---|---|---|
| 31 | Reserved | Always read 0 |

1. This value is configured by SCG_RCCR in Run mode, SCG_VCCR in VLPR mode, and SCG_HCCR in HSRUN mode.

## 12.5.6 MDM-AP System Power Status Register

**Table 12-11. MDM-AP System Power Status register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-7 | System Power Mode Status[1] | 001xxxx - RUN mode<br>010xxxx - HSRUN mode<br>100xxxx - VLPR mode<br>0000000 - STOP mode<br>0000001 - VLPS mode<br>0000011 - LLS mode<br>0000111 - VLLS2/3 mode<br>0001111 – VLLS0/1 mode |
| 8 | PMC LDO Status | Set to indicate PMC LD regulator enabled |
| 9 | PMC DCDC Status | Set to indicate PMC DCDC regulator enabled |
| 10 | PMC IOREG Status | Set to indicate PMC IOREG regulator enabled |
| 11-12 | Reserved for future use | Always read 0 |
| 13 | USB VREGIN OK | Set to indicate USB VREGIN power available |
| 14 | VBAT OK | Set to indicate VBAT power supply available |
| 15 | VDDIO2 OK | Set to indicate VDDIO2 power supply available |
| 16 | Flash Disable Status | Flash accesses are disabled (and generate a bus error) and the flash memory is placed in a low-power state, see details from SIM_FCFG1[FLASHDIS]<br>Set to indicate Flash is enabled<br>Clear to indicate Flash is disabled |
| 17-19 | Flash Standby Status | Flash Standby, see details from FSTBDY register in FTFE chapter<br>Set to indicate Standby mode enabled for relative flash block if FTFE_FSTDBYCTL[STDBYDIS] is clear<br>Clear to indicate Standby mode not enabled for relative flash block |
| 20-23 | CM4F Stop Mode Configuration | 0001 – VLPS<br>0011 – LLS<br>0111 – VLLS2/3<br>1000 – VLLS0/1 |
| 24-27 | RF Power Mode Status | 1000 – RRUN |

*Table continues on the next page...*

**Table 12-11. MDM-AP System Power Status register assignments (continued)**

| Bit | Name | Description |
|---|---|---|
| | | 0011 – RLLS<br>0111 – RVLLS |
| 28-31 | CM0+ Stop Mode Configuration | 0001 – VLPS<br>0011 – LLS<br>0111 – VLLS2/3<br>1000 – VLLS0/1 |

1. it refers to MCU requested power mode, see details in system power mode table from power chapter.

## 12.5.7 MDM-AP PCC0_CGC0 Register

**Table 12-12. MDM-AP PCC0_CGC0 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC0_CGC[0:31] | Set to indicate corresponding clock gated |

## 12.5.8 MDM-AP PCC0_CGC1 Register

**Table 12-13. MDM-AP PCC0_CGC1 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC0_CGC[32:63] | Set to indicate corresponding clock gated |

## 12.5.9 MDM-AP PCC0_CGC2 Register

**Table 12-14. MDM-AP PCC0_CGC2 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC0_CGC[64:95] | Set to indicate corresponding clock gated |

## 12.5.10 MDM-AP PCC0_CGC3 Register

**Table 12-15. MDM-AP PCC0_CGC3 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC0_CGC[96:127] | Set to indicate corresponding clock gated |

## 12.5.11 MDM-AP PCC0_CGC4 Register

**Table 12-16. MDM-AP PCC0_CGC4 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-7 | PCC0_CGC[128:135] | Set to indicate corresponding clock gated |
| 8-31 | Reserved for future use | |

## 12.5.12 MDM-AP PCC1_CGC0 Register

**Table 12-17. MDM-AP PCC1_CGC0 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC1_CGC[0:31] | Set to indicate corresponding clock gated |

## 12.5.13 MDM-AP PCC1_CGC1 Register

**Table 12-18. MDM-AP PCC1_CGC1 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC1_CGC[32:63] | Set to indicate corresponding clock gated |

## 12.5.14 MDM-AP PCC1_CGC2 Register

**Table 12-19. MDM-AP PCC1_CGC2 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC1_CGC[64:95] | Set to indicate corresponding clock gated |

## 12.5.15 MDM-AP PCC1_CGC3 Register

**Table 12-20. MDM-AP PCC1_CGC3 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-31 | PCC1_CGC[96:127] | Set to indicate corresponding clock gated |

## 12.5.16 MDM-AP PCC1_CGC4 Register

**Table 12-21. MDM-AP PCC1_CGC4 register assignments**

| Bit | Name | Description |
|---|---|---|
| 0-7 | PCC1_CGC[128:135] | Set to indicate corresponding clock gated |
| 8-31 | Reserved for future use | |

# 12.6 Debug resets

The debug system receives the following sources of reset:

- Debug reset (the CDBGRSTREQ field within the DP CTRL/STAT register) that allows the debugger to reset the debug logic.
- System POR reset

Conversely, the debug system is capable of generating system reset using the following mechanism:

- A system reset in the MDM-AP control register which allows the debugger to hold the system in reset.
- SYSRESETREQ field in the NVIC Application Interrupt and Reset control register

## 12.7 SWD Multi-Drop

ARM Multi-drop SWD technology brings the benefits of SWD to complex, multi-device-based system by enabling simultaneous access to a number of devices through a single connection, providing complex-device developers with a low power 2-pin debug and trace solution.

The Multi-drop SWD is fully backwards compatible, retaining existing single point-to-point host equipment connections, and permits a device to power down completely while that device is not selected, reducing power consumption.

Each target must be configured with a unique combination of target ID and instance ID, to enable a debugger to select a single target to communicate with:

- The target ID is a 32-bit field that uniquely identifies the system accessed by the SWJ-DP.

**Table 12-22. Target ID**

| Bits | Value | Description |
|---|---|---|
| [31:28] | SIM_SDID[REVID] | The revision of the part. |
| [27:12] | 1011_0000_0010_0000 | Identifies the part. |
| [11:1] | 000_0000_1110 | Identifies the designer of the device. |
| [0] | 1 | Reserved. |

- The instance ID is the inverse of FTFE_FOPT[31:28]. It is used to distinguish between multiple instances of the same target in a system, for example, because the same chip is used more than once on a board.

## 12.8 ITM

The ITM is an application-driven trace source that supports "printf" style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace -- Software can write directly to ITM stimulus registers. This emits packets.
2. Hardware trace -- The DWT generates these packets, and the ITM emits them.

3. Time stamping -- Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex-M4 clock or the bitclock rate of the Serial Wire Viewer (SWV) output clocks the counter.
4. Global system timestamping. Timestamps can optionally be generated using a system-wide 48-bit count value. The same count value can be used to insert timestamps in the ETM trace stream, allowing coarse-grain correlation.

## 12.9 Core trace connectivity with ETM

The following figure assumes the device contains an ETM.

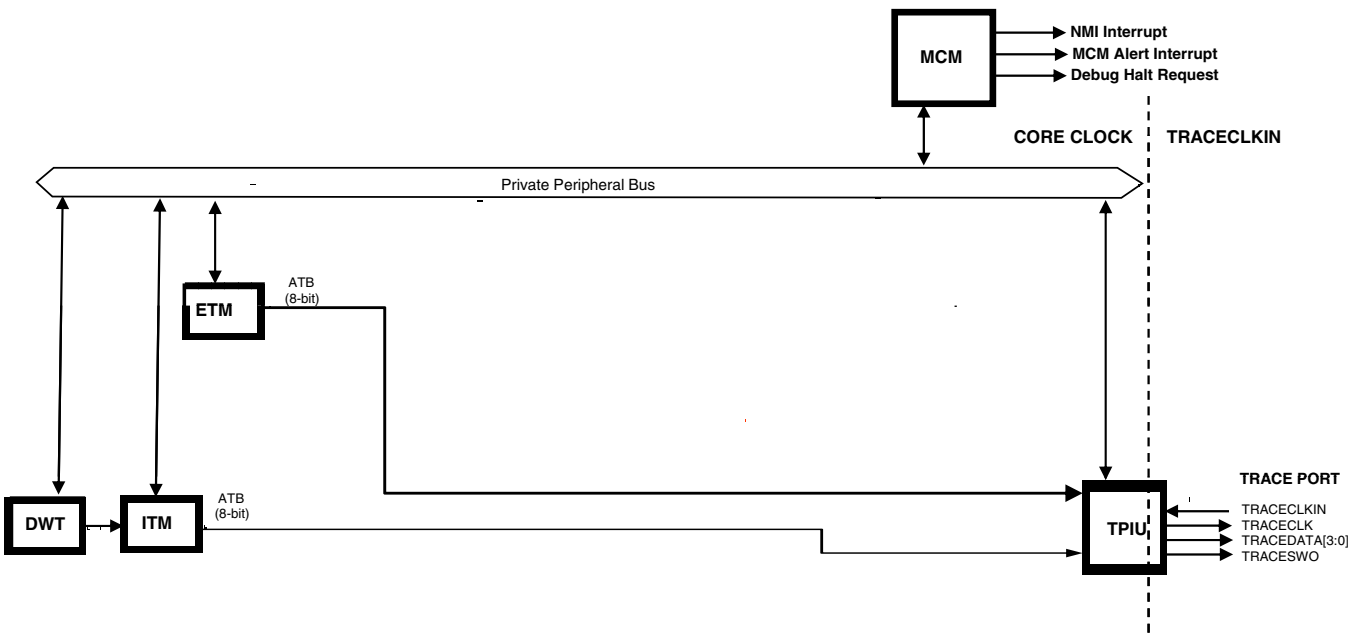



**Figure 12-3. Core Trace Connectivity with ETM only**

The following combinations paths are supported:

- ETM -> TPIU(4 pin or 2 pin parallel)
- ITM->TPIU(1 pin SWO, 2 pin or 4 pin parallel)
- ETM and ITM -> TPIU

## 12.10 Embedded Trace Macrocell v3.5 (ETM)

The Cortex-M4 Embedded Trace Macrocell (ETM-M4) is a debug component that enables a debugger to reconstruct program execution. The CoreSight ETM-M4 supports only instruction trace. You can use it with the Cortex-M4 Trace Port Interface Unit (M4-TPIU).

The main features of an ETM are:

- Tracing of 16-bit and 32-bit Thumb instructions
- Four EmbeddedICE watchpoint inputs
- Trace Start/Stop block with EmbeddedICE inputs
- One reduced function counter
- Two external inputs
- 24-byte FIFO queue
- Global timestamping

## 12.11 TPIU

The TPIU acts as a bridge between the on-chip trace data from the Embedded TraceMacrocell (ETM) and the Instrumentation Trace Macrocell (ITM), with separate IDs, to a data stream, encapsulating IDs where required, that is then captured by a Trace PortAnalyzer (TPA). The TPIU is specially designed for low-cost debug.

## 12.12 DWT

The DWT is a unit that performs the following debug functionality:

- It contains four comparators that you can configure as a hardware watchpoint, an ETM trigger, a PC sampler event trigger, or a data address sampler event trigger. The first comparator, DWT_COMP0, can also compare against the clock cycle counter, CYCCNT. The second comparator, DWT_COMP1, can also be used as a data comparator.
- The DWT contains counters for:
  - Clock cycles (CYCCNT)
  - Folded instructions
  - Load store unit (LSU) operations
  - Sleep cycles
  - CPI (all instruction cycles except for the first cycle)
  - Interrupt overhead

**NOTE**

An event is emitted each time a counter overflows.

- The DWT can be configured to emit PC samples at defined intervals, and to emit interrupt event information.

## 12.13 Micro Trace Buffer (MTB)

The Micro Trace Buffer (MTB) provides a simple execution trace capability for the Cortex-M0+ processor.

When enabled, the MTB records changes in program flow reported by the Cortex-M0+ processor, via the execution trace interface, into a configurable region of the SRAM. Subsequently, an off-chip debugger may extract the trace information, which would allow reconstruction of an instruction flow trace. The MTB does not include any form of load/store data trace capability or tracing of any other information.

In addition to providing the trace capability, the MTB also operates as a simple AHB-Lite SRAM controller. The system bus masters, including the processor, have read/write access to all of the SRAM via the AHB-Lite interface, allowing the memory to be also used to store program and data information. The MTB simultaneously stores the trace information into an attached SRAM and allows bus masters to access the memory. The MTB ensures that trace information write accesses to the SRAM take priority over accesses from the AHB-Lite interface.

The MTB includes trace control registers for configuring and triggering the MTB functions. The MTB also supports triggering via TSTART and TSTOP control functions in the MTB DWT module.

## 12.14 Debug in low-power modes

In low-power modes, in which the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low-power mode.

- In the case that the debugger is held static, the debug port returns to full functionality as soon as the low-power mode exits and the system returns to a state with active debug.
- In the case that the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured once the low-power mode is exited.

Power mode entry logic monitors Debug Power Up and System Power Up signals from the debug port as indications that a debugger is active. These signals can be changed in RUN, VLPR, WAIT and VLPW. When system attempts to enter any stop or low power mode, if debug signal is active and MDM-AP_Control[Inhibit Sleep] is enabled, FCLK continues to run to support core register access, but device enters pseudo-sleep mode. In these modes, in which FCLK is left active, the debug modules have access to core registers but not to system memory resources accessed via the crossbar.

With debug enabled, transitions from Run directly to VLPS result in the system entering Stop mode instead. Status bits within the MDM-AP Status register can be evaluated to determine this pseudo-VLPS state.

**NOTE**

With the debug enabled, transitions from Run --> VLPR --> VLPS are still possible.

In VLLS mode, all debug modules are powered off and reset at wakeup. In LLS mode, the debug modules retain their state but no debug activity is possible.

Waking up from a VLLSx mode causes all the debug controls and settings to be reset. To give time to the debugger to sync up with the HW, the MDM-AP Control register can be configured to hold the system in reset on recovery so that the debugger can regain control and reconfigure debug logic prior to the system exiting reset and resuming operation.

## 12.15 Debug module state in low power modes

The following table shows the state of the debug modules in low power modes. These terms are used: .

- FF = Full functionality. In VLPR and VLPW the system frequency is limited, but if a module does not have a limitation in its functionality, it is still listed as FF.
- Static = Module register states and associated memories are retained.
- OFF = Modules are powered off; module is in reset state upon wakeup.

**Table 12-23. Debug module state in low power modes**

| Module | STOP | VLPR | VLPW | VLPS | LLS | VLLSx |
|---|---|---|---|---|---|---|
| Debug Port | Static | FF | FF | Static | Static | OFF |
| AHB-AP | Static | FF | FF | Static | Static | OFF |
| DAP-AP | Static | FF | FF | Static | Static | OFF |
| ITM | Static | FF | FF | Static | Static | OFF |
| ETM | Static | FF | FF | Static | Static | OFF |
| DWT | Static | FF | FF | Static | Static | OFF |
| TPIU | Static | FF | FF | Static | Static | OFF |
| MTB | Static | FF | FF | Static | Static | OFF |

## 12.16 Debug and security

When flash security is enabled, the debug port capabilities are limited in order to prevent exploitation of secure data.

In the secure state, the debugger still has access to the MDM-AP status register and can determine the current secruity state of the device. In the case of a secure device, the debugger has the capability of only performing a mass erase or system reset operation. In the case of a secure device that has mass erase disabled (FSEC[MEEN] = 10), attempts to mass erase via the debug interface are blocked.

# Chapter 13
# RISC-V Core Debug

## 13.1 Introduction

RISC-V debug has separated JTAG ports and cannot access ARM CoreSight debug components. RI5CY core and ZERO-RISCY core both have interface to RISC-V debug. So RISC-V debug can configure debug registers in RI5CY core and ZERO-RISCY core to execute basic debug controls, which including run/halt, signal step, breakpoint etc.

For RISC-V debug registers, see RI5CY Debug Registers and ZERO-RISCY Debug Registers for more details.

## 13.2 Memory map for RISC-V debug port

0x0000_0000 – 0xF8FF_FFFF: system memory map, Debug port cannot access CM4/CM0P PPB modules except MCM2 in CM4/RI5CY platform.

0xF900_0000 – 0xF900_7FFF: RI5CY core registers, 0xF900_0000 + Address in the table.

**Table 13-1. RI5CY debug unit address map**

| Offset address | Name | Description |
|---|---|---|
| 0x0000-0x007F | Debug Registers | Always accessible, even when the core is running |
| 0x400-0x47F | GPR (x0-x31) | General Purpose Registers<br>Only accessible if the core is halted |
| 0x2000-0x20FF | Debug Registers | Only accessible if the core is halted |
| 0x4000-0x7FFF | CSR | Control and Status Registers<br>Only accessible if the core is halted |

0xF900_8000 – 0xF900_FFFF: ZERORISCY core registers, 0xF900_8000 + Address in the table.

**Table 13-2. ZERO-RISCY debug unit address map**

| Offset address | Name | Description |
|---|---|---|
| 0x0000-0x007F | Debug Registers | Always accessible, even when the core is running |
| 0x400-0x47F | GPR (x0-x31) | General Purpose Registers<br>Only accessible if the core is halted |
| 0x2000-0x20FF | Debug Registers | Only accessible if the core is halted |
| 0x4000-0x7FFF | CSR | Control and Status Registers<br>Only accessible if the core is halted |

0xF901_0000 – 0xF901_00FF: MDM-AP

## 13.3 Advanced debug interface

The Advanced Debug Interface (ADI) is a hardware module which creates an interface between a JTAG Test Access Port (TAP) and the system bus AXI. It is part of the system which allows software running on an SoC to be controlled and debugged by a software debugger such as GDB, running on a separate host PC. Here is block diagram of ADI.

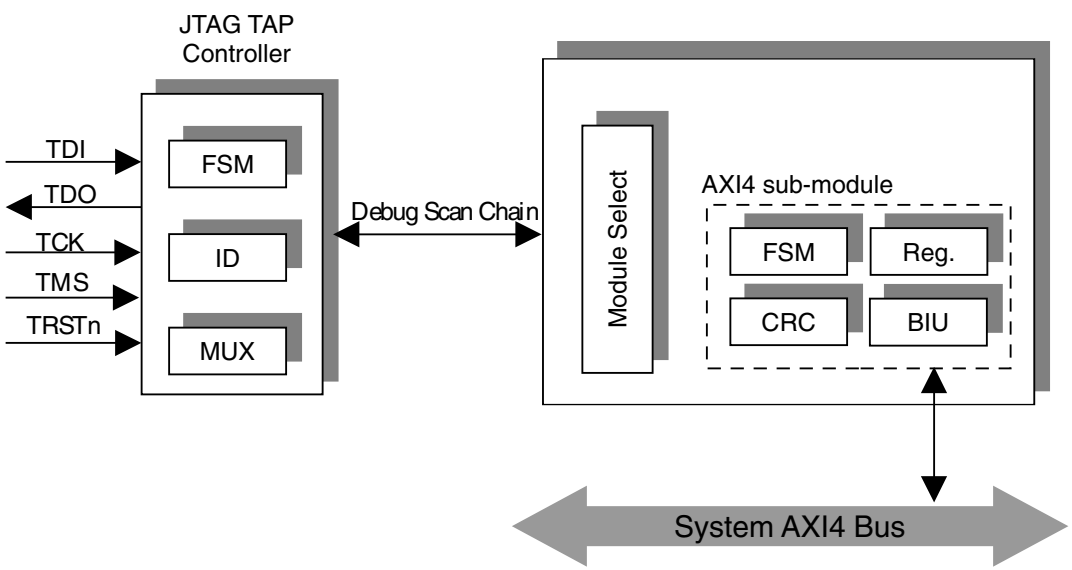



**Figure 13-1. ADI block diagram**

This is AXI4 module register summary in ADI.

**Table 13-3. ADI AXI4 register summary**

| Index | Register name |
|---|---|
| 0x0 | Error Register |
| 0x1 | DEBUG_PATH_SELECTION Register |
| 0x2 | POWER_UP Register |

## 13.3.1 Error register

The error register captures bus errors during transactions. If an error is present, then the error bit in the error register is set to '1', and the address of the failed access is stored in the rest of the error register. The error bit may only be reset by writing a '1' to the error bit.

Error Register format:

**Table 13-4. Error regiser format**

| Bit | Access | Description |
|---|---|---|
| 33:1 | R | When error bit = '1', contains the address of the failed transaction |
| 0 | R | Error bit. Set to '1' when a AXI4 error has occurred since the last time the error bit was reset. |
| 0 | W | Write as '1' to reset the error bit to '0' and re-enable error detection. |

## 13.3.2 DEBUG_PATH_SELECTION register

RISCV debug port can access system memory through CM4/RI5CY platform or CM0+/ ZERO-RISCY platform. The path for debug port to access system memory is configurable. The default path is determained by IFR boot configuration.

**Table 13-5. RISC-V debug port access path**

| BOOT CORE SEL | RI5CY_DIS | ZERORISCY_DIS | Path |
|---|---|---|---|
| 0 | 0 | 0 | ZERORISCY |
| 0 | 0 | 1 | RI5CY |
| 0 | 1 | 0 | ZERORISCY |
| 0 | 1 | 1 | RI5CY(NA) |
| 1 | 0 | 0 | RI5CY |
| 1 | 0 | 1 | RI5CY |

*Table continues on the next page...*

**Table 13-5. RISC-V debug port access path (continued)**

| BOOT CORE SEL | RI5CY_DIS | ZERORISCY_DIS | Path |
|---|---|---|---|
| 1 | 1 | 0 | ZERORISCY |
| 1 | 1 | 1 | RI5CY(NA) |

The user can change the path by writing DEBUG_PATH_SELECTION register in ADI. This register is a new addition to the ADI for this application. It's an AXI4 module register with index = 1.

DEBUG_PATH_SELECTION register format:

**Table 13-6. DEBUG_PATH_SELECTION register format**

| Bit | Access | Description |
|---|---|---|
| 1 | R/W | 0: Use default selection<br>1: Use bit 0 config |
| 0 | R/W | 0: RI5CY<br>1: ZERORISCY |

## 13.3.3 POWER_UP register

This register is to power up RISC-V debug components. The first step to debug RISC-V core is to write "1" to POWER_UP register.

POWER_UP register format:

**Table 13-7. POWER_UP register format**

| Bit | Access | Description |
|---|---|---|
| 0 | R/W | Write '1' to enable clock of RISC-V debug components |

# Chapter 14
# Micro Trace Buffer (MTB)

## 14.1 Introduction

Microcontrollers using the Cortex-M0+ processor core include support for a CoreSight Micro Trace Buffer to provide program trace capabilities.

The proper name for this function is the CoreSight Micro Trace Buffer for the Cortex-M0+ Processor; in this document, it is simply abbreviated as the MTB.

The simple program trace function creates instruction address change-of-flow data packets in a user-defined region of the system RAM. Accordingly, the system RAM controller manages requests from two sources:

- AMBA-AHB reads and writes from the system bus
- program trace packet writes from the processor

As part of the MTB functionality, there is a DWT (Data Watchpoint and Trace) module that allows the user to define watchpoint addresses, or optionally, an address and data value, that when triggered, can be used to start or stop the program trace recording.

This document details the functionality of both the MTB and DWT capabilities.

### 14.1.1 Overview

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown as follows:

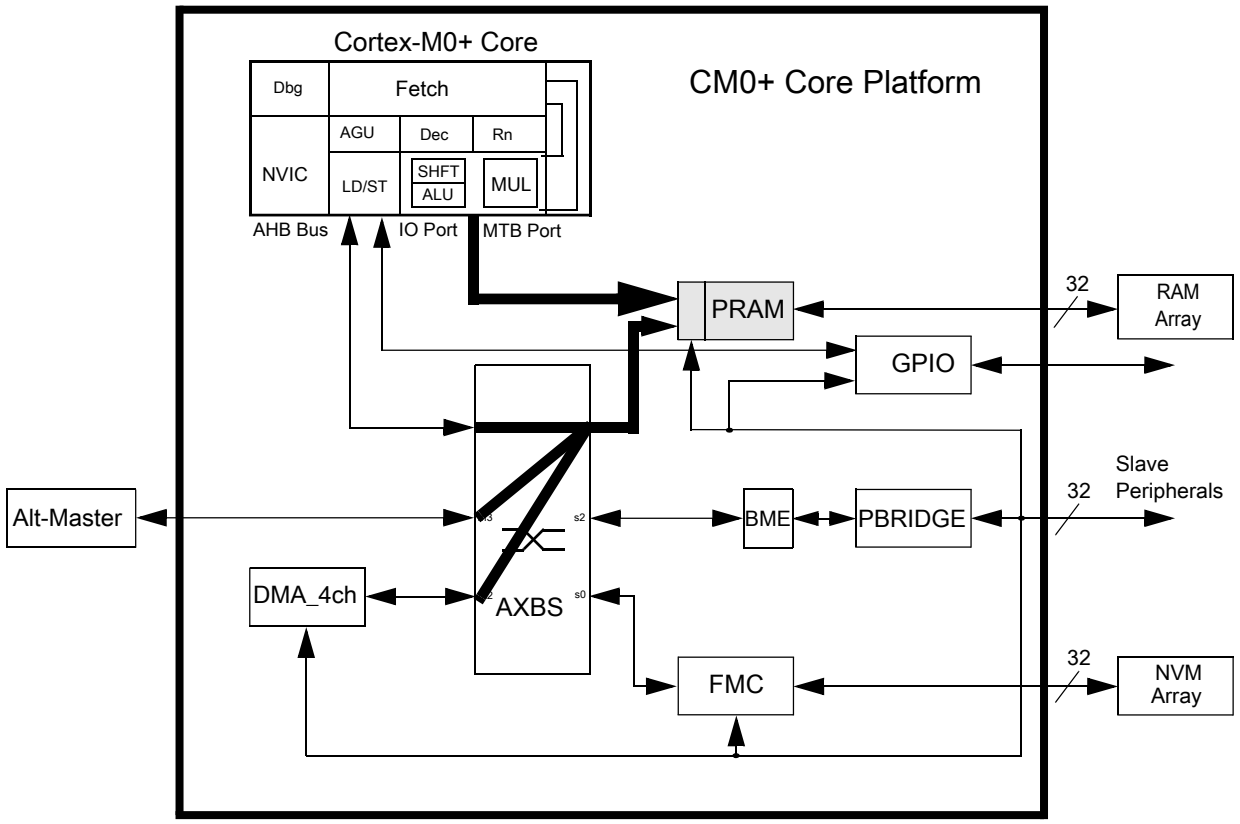



**Figure 14-1. Generic Cortex-M0+ core platform block diagram**

As shown in the block diagram, the platform RAM (PRAM) controller connects to two input buses:

- the crossbar slave port for system bus accesses
- a "private execution MTB port" from the core

The logical paths from the crossbar master input ports to the PRAM controller are highlighted along with the private execution trace port from the processor core. The private MTB port signals the instruction address information needed for the 64-bit program trace packets written into the system RAM. The PRAM controller output interfaces to the attached RAM array. In this document, the PRAM controller is the MTB controller.

The following information is taken from the ARM CoreSight Micro Trace Buffer documentation.

"The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry.

The processor can cause a trace packet to be generated for any instruction.

The following figure shows how the execution trace information is stored in memory as a sequence of packets.

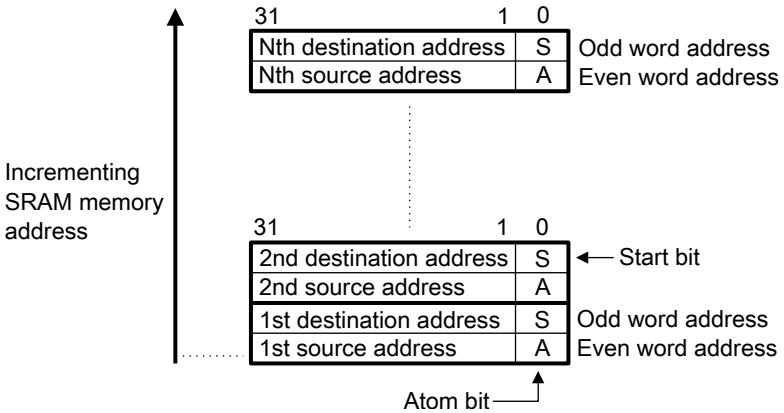



**Figure 14-2. MTB execution trace storage format**

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb instructions are at least halfword aligned. The least significant bit of the value is the A-bit. The A-bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero the branch originated from an instruction, when it is one the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, the address it branched to. The value stored only records bits[31:1] of the branch address. The least significant bit of the value is the S-bit. The S-bit indicates where the trace started. An S-bit value of 1 indicates where the first packet after the trace started and a value of 0 is used for other packets. Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S-bit set to 1. This word is always stored in the next higher word in memory, an odd word address.

When the A-bit is set to 1, the source address field contains the architecturally-preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A-bit is set to 0. In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
  - Source address field set to the address of the instruction that causes the exception return, BX or POP.

  - Destination address field set to bits[31:1] of the EXC_RETURN value. See the ARM v6-M Architecture Reference Manual.
  - The A-bit set to 0.

- The second packet has the:
  - Source address field set to bits[31:1] of the EXC_RETURN value.
  - Destination address field set to the address of the instruction where execution commences.
  - A-bit set to 1."

Given the recorded change-of-flow trace packets in system RAM and the memory image of the application, a debugger can read out the data and create an instruction-by-instruction program trace. In keeping with the low area and power implementation cost design targets, the MTB trace format is less efficient than other CoreSight trace modules, for example, the ETM (Embedded Trace Macrocell). Since each branch packet is 8 bytes in size, a 1 KB block of system RAM can contain 128 branches. Using the Dhrystone 2.1 benchmark's dynamic runtime as an example, this corresponds to about 875 instructions per KB of trace RAM, or with a zero wait state memory, this corresponds to approximately 1600 processor cycles per KB. This metric is obviously very sensitive to the runtime characteristics of the user code.

The DWT function (not shown in the core platform block diagram) monitors the processor address and data buses so that configurable watchpoints can be detected to trigger the appropriate response in the MTB recording.

### 14.1.2 Features

The key features of the MTB and DWT include:

- Memory controller for system RAM and Micro Trace Buffer for program trace packets
- Read/write capabilities for system RAM accesses, write-only for program trace packets
- Supports zero wait state response to system bus accesses when no trace data is being written
- Can buffer two AHB address phases and one data write for system RAM accesses
- Supports 64-bit program trace packets including source and destination instruction addresses
- Program trace information in RAM available to MCU's application code or external debugger
- Program trace watchpoint configuration accessible by MCU's application code or debugger
- Location and size of RAM trace buffer is configured by software

- Two DWT comparators (addresses or address + data) provide programmable start/ stop recording
- CoreSight compliant debug functionality

### 14.1.3 Modes of operation

The MTB and DWT functions do not support any special modes of operation. The MTB controller, as a memory-mapped device located on the platform's slave AHB system bus, responds strictly on the basis of memory addresses for accesses to its attached RAM array. The MTB private execution bus provides program trace packet write information to the RAM controller. Both the MTB and DWT modules are memory-mapped, so their programming models can be accessed.

All functionality associated with the MTB and DWT modules resides in the core platform's clock domain; this includes its connections with the RAM array.

## 14.2 External signal description

The MTB and DWT modules do not directly support any external interfaces.

The internal interface includes a standard AHB bus with a 32-bit datapath width from the appropriate crossbar slave port plus the private execution trace bus from the processor core. The signals in the private execution trace bus are detailed in the following table taken from the ARM CoreSight Micro Trace Buffer documentation. The signal direction is defined as viewed by the MTB controller.

**Table 14-1. Private execution trace port from the core to MTB**

| Signal | Direction | Description |
|---|---|---|
| LOCKUP | Input | Indicates the processor is in the Lockup state. This signal is driven LOW for cycles when the processor is executing normally and driven HIGH for every cycle the processor is waiting in the Lockup state. This signal is valid on every cycle. |
| IAESEQ | Input | Indicates the next instruction address in execute, IAEX, is sequential, that is non-branching. |
| IAEXEN | Input | IAEX register enable. |
| IAEX[30:0] | Input | Registered address of the instruction in the execution stage, shifted right by one bit, that is, PC >> 1. |
| ATOMIC | Input | Indicates the processor is performing non-instruction related activities. |
| EDBGRQ | Output | Request for the processor to enter the Debug state, if enabled, and halt. |

In addition, there are two signals formed by the DWT module and driven to the MTB controller: TSTART (trace start) and TSTOP (trace stop). These signals can be configured using the trace watchpoints to define programmable addresses and data values to affect the program trace recording state.

## 14.3 Memory map and register definition

The MTB and DWT modules each support a sparsely-populated 4 KB address space for their programming models. For each address space, there are a variety of control and configurable registers near the base address, followed by a large unused address space and finally a set of CoreSight registers to support dynamic determination of the debug configuration for the device.

Accesses to the programming model follow standard ARM conventions. Taken from the ARM CoreSight Micro Trace Buffer documentation, these are:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- The behavior of the MTB is UNPREDICTABLE if the registers with UNKNOWN reset values are not programmed prior to enabling trace.
- Unless otherwise stated in the accompanying text:
  - Do not modify reserved register bits
  - Ignore reserved register bits on reads
  - All register bits are reset to a logic 0 by a system or power-on reset
  - Use only word size, 32-bit, transactions to access all registers

### 14.3.1 MTB register descriptions

#### 14.3.1.1 MTB Memory map

MTB base address: F000_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | MTB Position Register (POSITION) | 32 | RW | See description. |
| 4h | MTB Master Register (MASTER) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | MTB Flow Register (FLOW) | 32 | RW | See description. |
| Ch | MTB Base Register (BASE) | 32 | RO | See description. |
| F00h | Integration Mode Control Register (MODECTRL) | 32 | RO | 0000_0000h |
| FA0h | Claim TAG Set Register (TAGSET) | 32 | RO | 0000_0000h |
| FA4h | Claim TAG Clear Register (TAGCLEAR) | 32 | RO | 0000_0000h |
| FB0h | Lock Access Register (LOCKACCESS) | 32 | RO | 0000_0000h |
| FB4h | Lock Status Register (LOCKSTAT) | 32 | RO | 0000_0000h |
| FB8h | Authentication Status Register (AUTHSTAT) | 32 | RO | 0000_0000h |
| FBCh | Device Architecture Register (DEVICEARCH) | 32 | RO | 4770_0A31h |
| FC8h | Device Configuration Register (DEVICECFG) | 32 | RO | 0000_0000h |
| FCCh | Device Type Identifier Register (DEVICETYPID) | 32 | RO | 0000_0031h |
| FD0h | Peripheral ID Register (PERIPHID4) | 32 | RO | See description. |
| FD4h | Peripheral ID Register (PERIPHID5) | 32 | RO | See description. |
| FD8h | Peripheral ID Register (PERIPHID6) | 32 | RO | See description. |
| FDCh | Peripheral ID Register (PERIPHID7) | 32 | RO | See description. |
| FE0h | Peripheral ID Register (PERIPHID0) | 32 | RO | See description. |
| FE4h | Peripheral ID Register (PERIPHID1) | 32 | RO | See description. |
| FE8h | Peripheral ID Register (PERIPHID2) | 32 | RO | See description. |
| FECh | Peripheral ID Register (PERIPHID3) | 32 | RO | See description. |
| FF0h - FFCh | Component ID Register (COMPID0 - COMPID3) | 32 | RO | See description. |

## 14.3.1.2 MTB Position Register (POSITION)

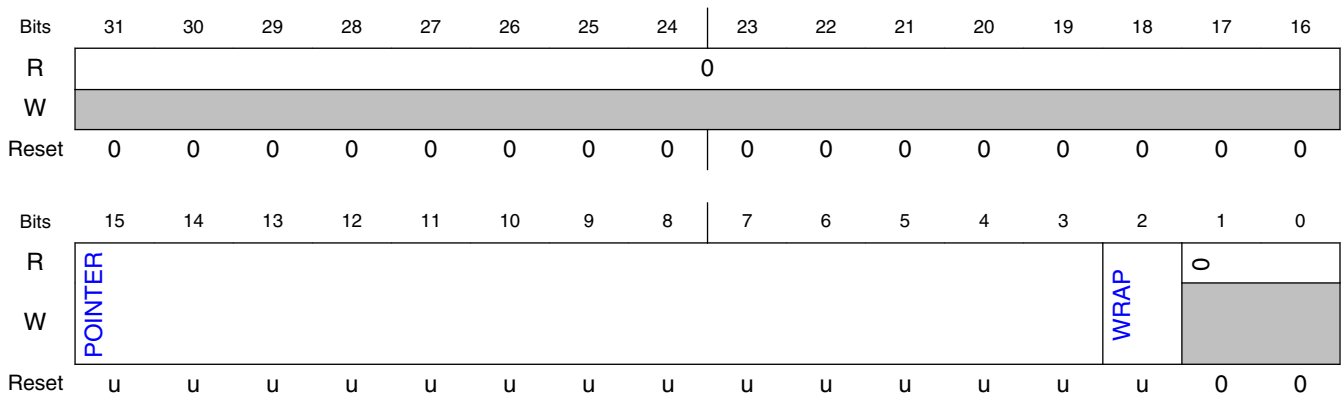
### 14.3.1.2.1 Offset

| Register | Offset |
|---|---|
| POSITION | 0h |

### 14.3.1.2.2 Function

The POSITION register contains the Trace Write Address Pointer and Wrap fields. This register can be modified by the explicit programming model writes. It is also automatically updated by the MTB hardware when trace packets are being recorded.

### 14.3.1.2.3 Diagram




### 14.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved<br>These bits must be treated as UNK/SBZP (unknown on reads, should-be-zero-or-preserved on writes). |
| 15-3<br>POINTER | Trace Packet Address Pointer<br>Because a packet consists of 2 words, the POINTER field is the address of the first word of a packet. This field contains bits[31:3] of the RAM address where the next trace packet is written. Therefore, it points to an unused location and is automatically incremented.<br>A debug agent can calculate the system memory map address for the current location in the MTB, using the following "generic" equation:<br>systemAddress = BASE+(POSITION & 0xFFFF_FFF8);<br>**NOTE:** The size of the RAM is parameterized and the most significant bits of the POINTER field are RAZ/WI.<br>For these devices, POSITION[31:16] == POSITION[POINTER[28:13] are RAZ/WI. Therefore, the active bits in this field are POSITION[15:3] == POSITION[POINTER[12:0]]. |
| 2<br>WRAP | WRAP<br>This field is set to 1 automatically when the POINTER value wraps (as determined by the MASTER[MASK] field in the MASTER Trace Control Register). A debug agent might use the WRAP field to determine whether the trace information above and below the pointer address is valid. |
| 1-0<br>— | Reserved<br>These bits must be treated as UNK/SBZP (unknown on reads, should-be-zero-or-preserved on writes). |

## 14.3.1.3 MTB Master Register (MASTER)

### 14.3.1.3.1 Offset

| Register | Offset |
|---|---|
| MASTER | 4h |

### 14.3.1.3.2 Function

The MASTER register contains the main program trace enable plus other trace controls. This register can be modified by the explicit programming model writes. MASTER[EN] and MASTER[HALTREQ] fields are also automatically updated by the MTB hardware.

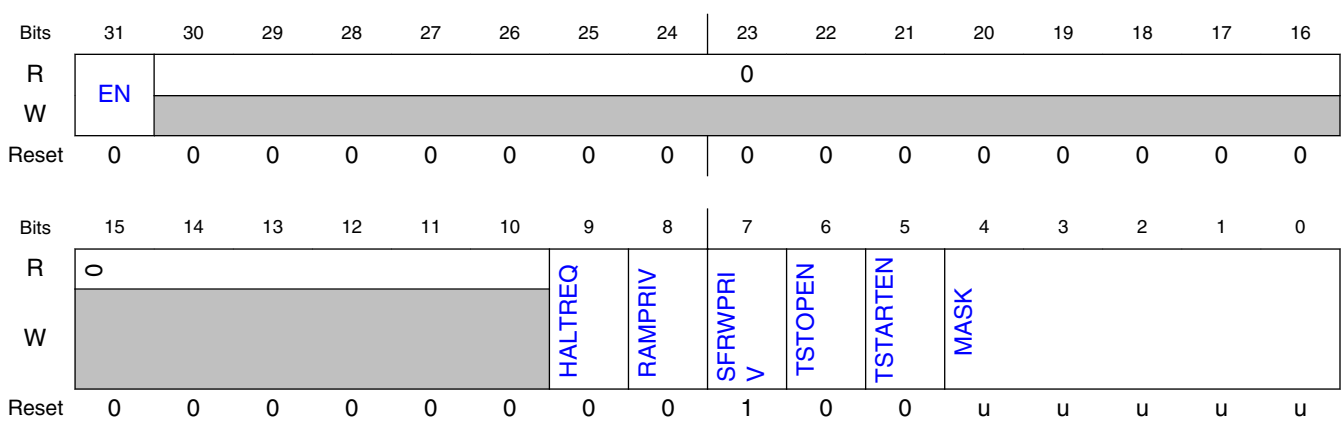
Before MASTER[EN] or MASTER[TSTARTEN] are set to 1, the software must initialize the POSITION and FLOW registers.

If FLOW[WATERMARK] is used to stop tracing or to halt the processor, MASTER[MASK] must still be set to a value that prevents POSITION[POINTER] from wrapping before it reaches the FLOW[WATERMARK] value.

**NOTE**

The format of this mask field is different than DWT_MASKn[MASK].

### 14.3.1.3.3 Diagram



#### 14.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>EN | Main Trace Enable<br>When this field is 1, trace data is written into the RAM memory location addressed by POSITION[POINTER]. The POSITION[POINTER] value auto increments after the trace data packet is written.<br>EN can be automatically set to 0 using the FLOW[WATERMARK] field and the FLOW[AUTOSTOP] bit.<br>EN is automatically set to 1 if TSTARTEN is 1 and the TSTART signal is HIGH.<br>EN is automatically set to 0 if TSTOPEN is 1 and the TSTOP signal is HIGH.<br>**NOTE:** If EN is set to 0 because FLOW[WATERMARK] is set, then it is not automatically set to 1 if TSTARTEN is 1 and the TSTART input is HIGH. In this case, tracing can only be restarted if FLOW[WATERMARK] or POSITION[POINTER] value is changed by software. |
| 30-10<br>— | Reserved<br>These bits must be treated as UNK/SBZP (unknown on reads, should-be-zero-or-preserved on writes). |
| 9<br>HALTREQ | Halt Request<br>This field is connected to the halt request signal of the trace logic, EDBGRQ. When HALTREQ is set to 1, the EDBFGRQ is asserted if DBGEN (invasive debug enable, one of the debug authentication interface signals) is also HIGH. HALTREQ can be automatically set to 1 using FLOW[WATERMARK]. |
| 8<br>RAMPRIV | RAM Privilege<br>If this field is 0, then user or privileged AHB read and write accesses to the RAM are permitted. If this field is 1, then only privileged AHB read and write accesses to the RAM are permitted and user accesses are RAZ/WI. The HPROT[1] signal determines if an access is a user or privileged mode reference. |
| 7<br>SFRWPRIV | Special Function Register Write Privilege<br>If this field is 0, then user or privileged AHB read and write accesses to the MTB Special Function Registers (programming model) are permitted. If this field is 1, then only privileged write accesses are permitted; user write accesses are ignored. The HPROT[1] signal determines if an access is user or privileged. Note MTB SFR read access are not controlled by this bit and are always permitted. |
| 6<br>TSTOPEN | Trace Stop Input Enable<br>If this field is 1 and the TSTOP signal is HIGH, then EN is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped. |
| 5<br>TSTARTEN | Trace Start Input Enable<br>If this field is 1 and the TSTART signal is HIGH, then EN is set to 1. Tracing continues until a stop condition occurs. |
| 4-0<br>MASK | Mask<br>This value determines the maximum size of the trace buffer in RAM. It specifies the most-significant bit of the POSITION[POINTER] field that can be updated by automatic increment. If the trace tries to advance past this power of 2, the POSITION[WRAP] bit is set to 1, the POSITION[MASK+3:3] == POSITION[POINTER[MASK:0]] bits are set to 0, and the POSITION[14:MASK+3] == POSITION[POINTER[11:MASK+1]] bits remain unchanged.<br>This field causes the trace packet information to be stored in a circular buffer of size 2^[MASK+4] bytes, that can be positioned in memory at multiples of this size. As detailed in the POSITION description, typical "upper limits" for the MTB size are RAM_Size/4 or RAM_Size/2. Values greater than the maximum have the same effect as the maximum. |

## 14.3.1.4 MTB Flow Register (FLOW)

### 14.3.1.4.1 Offset

| Register | Offset |
|---|---|
| FLOW | 8h |

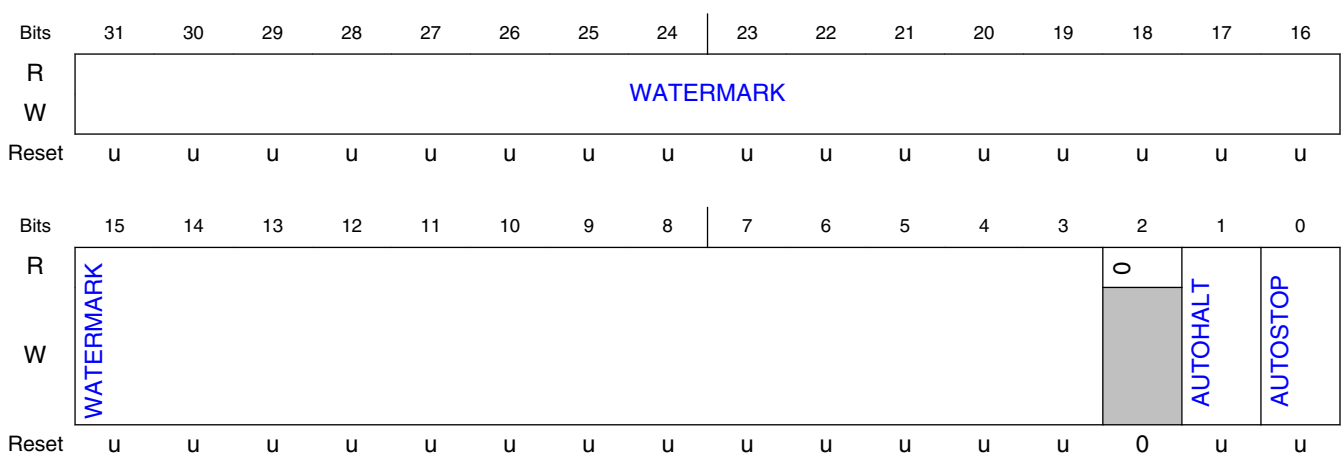
### 14.3.1.4.2 Function

The FLOW register contains the watermark address and the autostop/autohalt control bits. If tracing is stopped using the watermark autostop feature, it cannot be restarted until software clears the watermark autostop. This can be achieved in one of the following ways:

- Changing the POSITION[POINTER] field value to point to the beginning of the trace buffer, or
- Setting FLOW[AUTOSTOP] = 0.

A debug agent can use FLOW[AUTOSTOP] to fill the trace buffer once only without halting the processor.A debug agent can use FLOW[AUTOHALT] to fill the trace buffer once before causing the Cortex-M0+ processor to enter the Debug state. To enter Debug state, the Cortex-M0+ processor might have to perform additional branch type operations. Therefore, the FLOW[WATERMARK] field must be set below the final entry in the trace buffer region.

### 14.3.1.4.3 Diagram



#### 14.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-3<br>WATERMARK | WATERMARK[28:0]<br>This field contains an address in the same format as the POSITION[POINTER] field. When POSITION[POINTER] matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed. |
| 2<br>— | Reserved<br>These bits must be treated as UNK/SBZP (unknown on reads, should-be-zero-or-preserved on writes). |
| 1<br>AUTOHALT | AUTOHALT<br>If this field is 1 and WATERMARK is equal to POSITION[POINTER], then MASTER[HALTREQ] is automatically set to 1. If the DBGEN signal is HIGH, the MTB asserts this halt request to the Cortex-M0+ processor by asserting the EDBGRQ signal. |
| 0<br>AUTOSTOP | AUTOSTOP<br>If this field is 1 and WATERMARK is equal to POSITION[POINTER], then MASTER[EN] is automatically set to 0. This stops tracing. |

### 14.3.1.5 MTB Base Register (BASE)

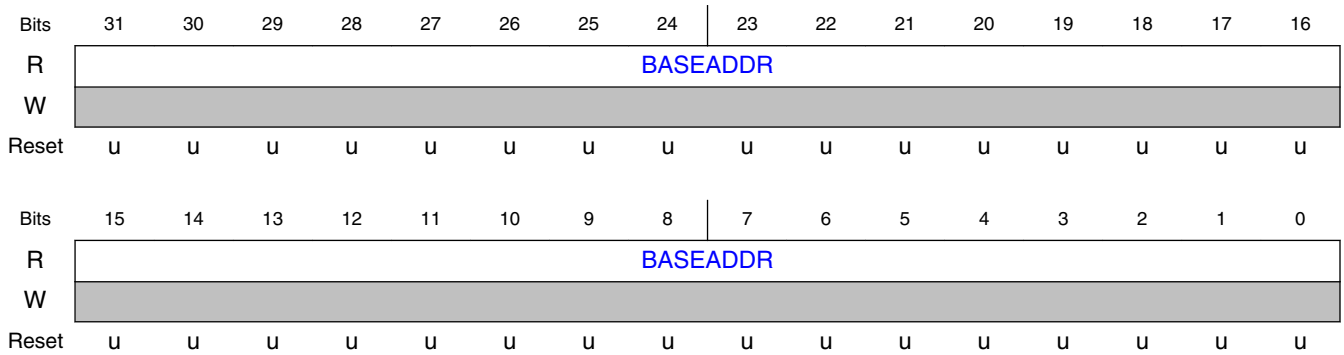
#### 14.3.1.5.1 Offset

| Register | Offset |
|---|---|
| BASE | Ch |

#### 14.3.1.5.2 Function

The read-only BASE Register indicates where the RAM is located in the system memory map. This register is provided to enable auto discovery of the MTB RAM location by a debug agent, and is defined by a hardware design parameter.

#### 14.3.1.5.3 Diagram




#### 14.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>BASEADDR | BASEADDR<br>This value is defined (hardwired): 0x09000000 |

### 14.3.1.6 Integration Mode Control Register (MODECTRL)

#### 14.3.1.6.1 Offset

| Register | Offset |
|---|---|
| MODECTRL | F00h |

#### 14.3.1.6.2 Function

This register enables the device to switch from a functional mode, or default behavior, into integration mode. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MODECTRL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MODECTRL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 14.3.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MODECTRL | MODECTRL<br>Hardwired to 0x0000_0000 |

### 14.3.1.7 Claim TAG Set Register (TAGSET)

#### 14.3.1.7.1 Offset

| Register | Offset |
|---|---|
| TAGSET | FA0h |

#### 14.3.1.7.2 Function

The Claim Tag Set Register returns the number of bits that can be set on a read, and enables individual bits to be set on a write. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TAGSET | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TAGSET | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 14.3.1.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>TAGSET | TAGSET<br>Hardwired to 0x0000_0000 |

### 14.3.1.8 Claim TAG Clear Register (TAGCLEAR)

#### 14.3.1.8.1 Offset

| Register | Offset |
|---|---|
| TAGCLEAR | FA4h |

#### 14.3.1.8.2 Function

The read/write Claim Tag Clear Register is used to read the claim status on debug resources. A read indicates the claim tag status. Writing 1 to a specific bit clears the corresponding claim tag to 0. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TAGCLEAR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TAGCLEAR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 14.3.1.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>TAGCLEAR | TAGCLEAR<br>Hardwired to 0x0000_0000 |

### 14.3.1.9 Lock Access Register (LOCKACCESS)

#### 14.3.1.9.1 Offset

| Register | Offset |
|---|---|
| LOCKACCESS | FB0h |

#### 14.3.1.9.2 Function

The Lock Access Register enables a write access to component registers. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKACCESS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKACCESS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 14.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LOCKACCESS | LOCKACCESS<br>Hardwired to 0x0000_0000 |

### 14.3.1.10 Lock Status Register (LOCKSTAT)

#### 14.3.1.10.1 Offset

| Register | Offset |
|---|---|
| LOCKSTAT | FB4h |

#### 14.3.1.10.2 Function

The Lock Status Register indicates the status of the lock control mechanism. This register is used in conjunction with the Lock Access Register. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKSTAT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKSTAT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 14.3.1.10.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LOCKSTAT | LOCKSTAT<br>Hardwired to 0x0000_0000 |

## 14.3.1.11 Authentication Status Register (AUTHSTAT)

### 14.3.1.11.1 Offset

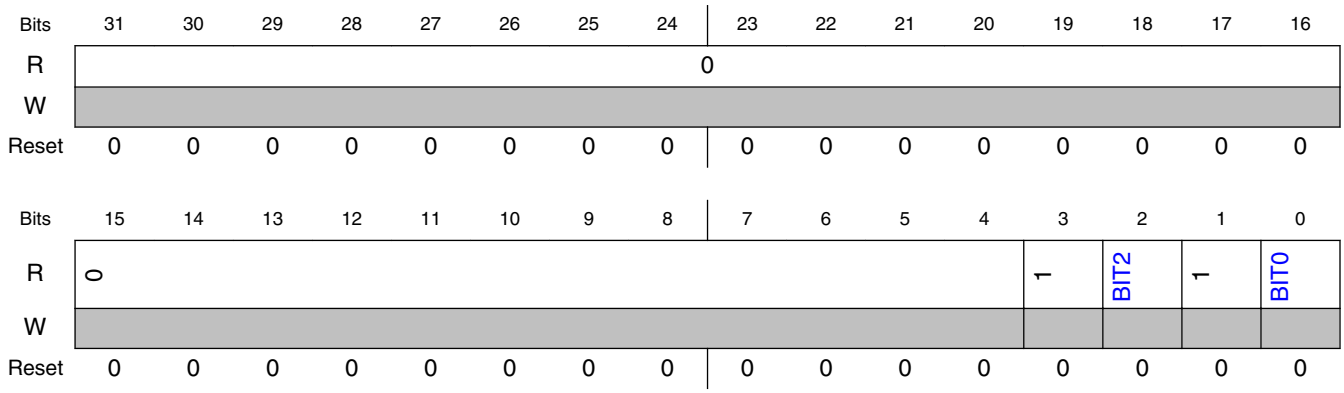
| Register | Offset |
|---|---|
| AUTHSTAT | FB8h |

### 14.3.1.11.2 Function

The Authentication Status Register reports the required security level and current status of the security enable bit pairs. Where functionality changes on a given security level, this change must be reported in this register. It is connected to specific signals used during the auto-discovery process by an external debug agent.

AUTHSTAT[3:2] indicates if nonsecure, noninvasive debug is enabled or disabled, while AUTHSTAT[1:0] indicates the enabled/disabled state of nonsecure, invasive debug. For both 2-bit fields, 0b10 indicates the functionality is disabled and 0b11 indicates it is enabled.

### 14.3.1.11.3 Diagram




### 14.3.1.11.4 Fields

| Field | Function |
|---|---|
| 31-4 | Reserved |
| — | This field is hardwired to 0x0000000. |
| 3 | BIT3 |
| BIT3 | Hardwired to 1. |
| 2 | BIT2 |
| BIT2 | Connected to NIDEN or DBGEN signal. |
| 1 | BIT1 |
| BIT1 | Hardwired to 1. |
| 0 | BIT0 |
| BIT0 | Connected to DBGEN. |

## 14.3.1.12 Device Architecture Register (DEVICEARCH)

### 14.3.1.12.1 Offset

| Register | Offset |
|---|---|
| DEVICEARCH | FBCh |

### 14.3.1.12.2 Function

This register indicates the device architecture. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICEARCH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICEARCH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

### 14.3.1.12.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DEVICEARCH | DEVICEARCH<br>Hardwired to 0x4770_0A31. |

## 14.3.1.13 Device Configuration Register (DEVICECFG)

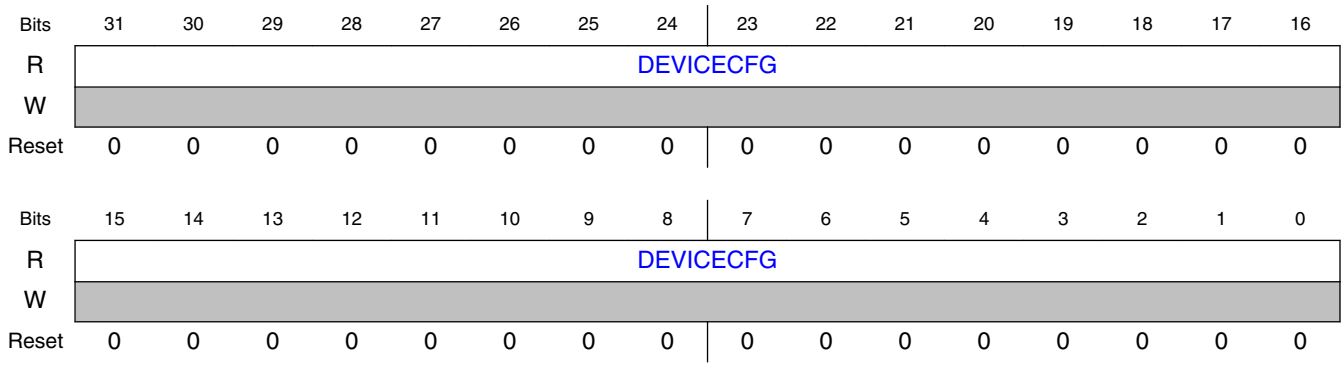
### 14.3.1.13.1 Offset

| Register | Offset |
|---|---|
| DEVICECFG | FC8h |

### 14.3.1.13.2 Function

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.1.13.3 Diagram




### 14.3.1.13.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DEVICECFG | DEVICECFG<br>Hardwired to 0x0000_0000. |

## 14.3.1.14 Device Type Identifier Register (DEVICETYPID)

### 14.3.1.14.1 Offset

| Register | Offset |
|---|---|
| DEVICETYPID | FCCh |

### 14.3.1.14.2 Function

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.1.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICETYPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICETYPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

### 14.3.1.14.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DEVICETYPID | DEVICETYPID<br>Hardwired to 0x0000_0031. |

## 14.3.1.15 Peripheral ID Register (PERIPHID4 - PERIPHID3)

### 14.3.1.15.1 Offset

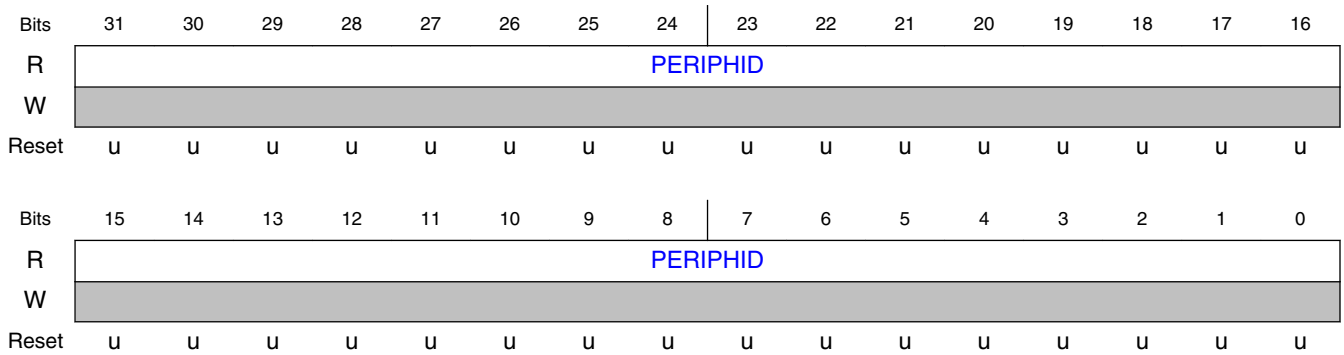
For a = 0 to 7:

| Register | Offset |
|---|---|
| PERIPHIDa | FC0h + (a × 4h) |

### 14.3.1.15.2 Function

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.1.15.3 Diagram




### 14.3.1.15.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PERIPHID | PERIPHID<br>Peripheral ID4 is hardwired to 0x0000_0004; ID0 to 0x0000_0032; ID1 to 0x0000_00B9; ID2 to 0x0000_001B; and all the others to 0x0000_0000. |

## 14.3.1.16 Component ID Register (COMPID0 - COMPID3)

### 14.3.1.16.1 Offset

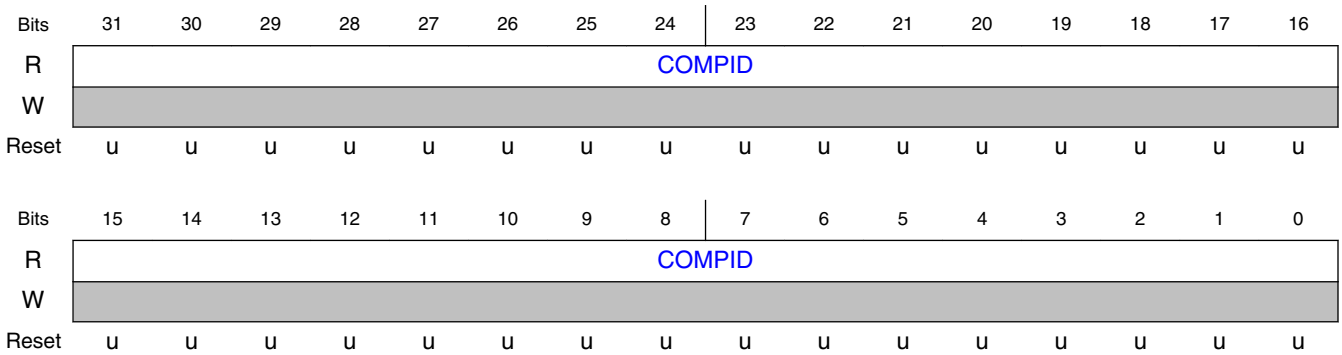
For a = 0 to 3:

| Register | Offset |
|---|---|
| COMPIDa | FF0h + (a × 4h) |

### 14.3.1.16.2 Function

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.1.16.3 Diagram




#### 14.3.1.16.4 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPID | Component ID<br>Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1. |

## 14.3.2 DWT register descriptions

### 14.3.2.1 DWT Memory map

The DWT programming model supports a very simplified subset of the v7M debug architecture and follows the standard ARM DWT definition.

MTBDWT base address: F000_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | DWT Control Register (CTRL) | 32 | RO | 2F00_0000h |
| 20h | DWT Comparator Register (COMP0) | 32 | RW | 0000_0000h |
| 24h | DWT Comparator Mask Register (MASK0) | 32 | RW | 0000_0000h |
| 28h | DWT Comparator Function Register 0 (FCT0) | 32 | RW | 0000_0000h |
| 30h | DWT Comparator Register (COMP1) | 32 | RW | 0000_0000h |
| 34h | DWT Comparator Mask Register (MASK1) | 32 | RW | 0000_0000h |
| 38h | DWT Comparator Function Register 1 (FCT1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 200h | DWT Trace Buffer Control Register (TBCTRL) | 32 | RW | 2000_0000h |
| FC8h | Device Configuration Register (DEVICECFG) | 32 | RO | 0000_0000h |
| FCCh | Device Type Identifier Register (DEVICETYPID) | 32 | RO | 0000_0004h |
| FD0h | Peripheral ID Register (PERIPHID4) | 32 | RO | See description. |
| FD4h | Peripheral ID Register (PERIPHID5) | 32 | RO | See description. |
| FD8h | Peripheral ID Register (PERIPHID6) | 32 | RO | See description. |
| FDCh | Peripheral ID Register (PERIPHID7) | 32 | RO | See description. |
| FE0h | Peripheral ID Register (PERIPHID0) | 32 | RO | See description. |
| FE4h | Peripheral ID Register (PERIPHID1) | 32 | RO | See description. |
| FE8h | Peripheral ID Register (PERIPHID2) | 32 | RO | See description. |
| FECh | Peripheral ID Register (PERIPHID3) | 32 | RO | See description. |
| FF0h - FFCh | Component ID Register (COMPID0 - COMPID3) | 32 | RO | See description. |

## 14.3.2.2 DWT Control Register (CTRL)

### 14.3.2.2.1 Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

### 14.3.2.2.2 Function

This register provides read-only information on the watchpoint configuration for the DWT.

### 14.3.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NUMCMP | | | | DWTCFGCTRL | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DWTCFGCTRL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 14.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31-28<br>NUMCMP | Number of comparators<br>The DWT implements two comparators. |
| 27-0<br>DWTCFGCTRL | DWT configuration controls<br>This field is hardwired to 0xF00_0000, disabling all the remaining DWT functionality. The specific fields and their state are:<br>CTRL[27] = NOTRCPKT = 1, trace sample and exception trace is not supported<br>CTRL[26] = NOEXTTRIG = 1, external match signals are not supported<br>CTRL[25] = NOCYCCNT = 1, cycle counter is not supported<br>CTRL[24] = NOPRFCNT = 1, profiling counters are not supported<br>CTRL[22] = CYCEBTENA = 0, no POSTCNT underflow packets generated<br>CTRL[21] = FOLDEVTENA = 0, no folded instruction counter overflow events<br>CTRL[20] = LSUEVTENA = 0, no LSU counter overflow events<br>CTRL[19] = SLEEPEVTENA = 0, no sleep counter overflow events<br>CTRL[18] = EXCEVTENA = 0, no exception overhead counter events<br>CTRL[17] = CPIEVTENA = 0, no CPI counter overflow events<br>CTRL[16] = EXCTRCENA = 0, generation of exception trace disabled<br>CTRL[12] = PCSAMPLENA = 0, no periodic PC sample packets generated<br>CTRL[11:10] = SYNCTAP = 0, no synchronization packets<br>CTRL[9] = CYCTAP = 0, cycle counter is not supported<br>CTRL[8:5] = POSTINIT = 0, cycle counter is not supported<br>CTRL[4:1] = POSTPRESET = 0, cycle counter is not supported<br>CTRL[0] = CYCCNTENA = 0, cycle counter is not supported |

## 14.3.2.3 DWT Comparator Register (COMP0 - COMP1)

### 14.3.2.3.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| COMPa | 20h + (a × 10h) |

### 14.3.2.3.2 Function

The COMPn registers provide the reference value for comparator n.

### 14.3.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 14.3.2.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>COMP | Reference value for comparison<br>If COMP0 is used for a data value comparator and the access size is byte or halfword, the data value must be replicated across all appropriate byte lanes of this register. For example, if the data is a byte-sized "x" value, then COMP[31:24] = COMP[23:16] = COMP[15:8] = COMP[7:0] = "x". Likewise, if the data is a halfword-size "y" value, then COMP[31:16] = COMP[15:0] = "y". |

## 14.3.2.4 DWT Comparator Mask Register (MASK0 - MASK1)

### 14.3.2.4.1 Offset

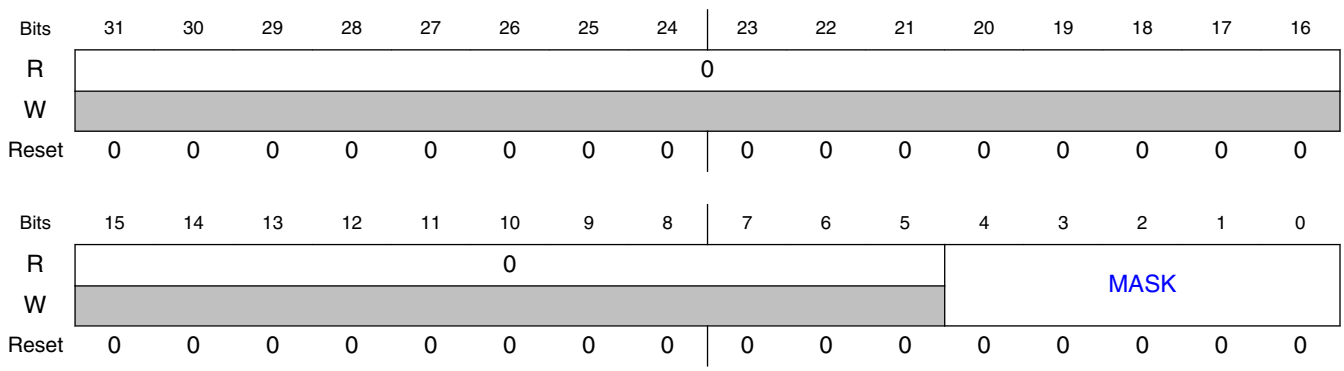
For a = 0 to 1:

| Register | Offset |
|---|---|
| MASKa | 24h + (a × 10h) |

### 14.3.2.4.2 Function

The MASKn registers define the size of the ignore mask applied to the reference address for address range matching by comparator n. Note the format of this mask field is different than the MTB_MASTER[MASK].

### 14.3.2.4.3 Diagram




### 14.3.2.4.4 Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>MASK | MASK<br>The value of the ignore mask, 0-31 bits, is applied to address range matching. MASK = 0 is used to include all bits of the address in the comparison, except if MASK = 0 and the comparator is configured to watch instruction fetch addresses, address bit [0] is ignored by the hardware since all fetches must be at least halfword aligned. For MASK != 0 and regardless of watch type, address bits [x-1:0] are ignored in the address comparison.<br>Using a mask means the comparator matches on a range of addresses, defined by the unmasked most significant bits of the address, bits [31:x]. The maximum MASK value is 24, producing a 16 Mbyte mask. An attempted write of a MASK value > 24 is limited by the DWT hardware to 24.<br>If COMP0 is used as a data value comparator, then MASK0 should be programmed to zero. |

## 14.3.2.5 DWT Comparator Function Register 0 (FCT0)

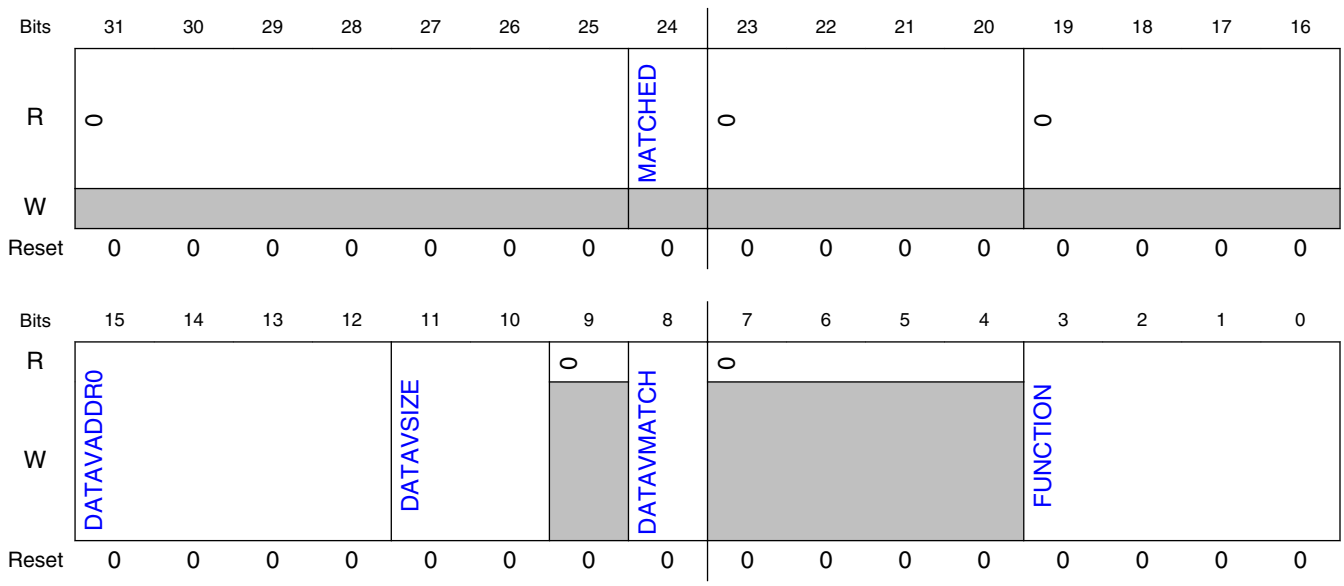
### 14.3.2.5.1 Offset

| Register | Offset |
|---|---|
| FCT0 | 28h |

### 14.3.2.5.2 Function

The FCTn registers control the operation of comparator n.

### 14.3.2.5.3 Diagram




### 14.3.2.5.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MATCHED | Comparator match<br>If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.<br>0b - No match.<br>1b - Match occurred. |
| 23-20<br>— | Reserved |
| 19-16<br>— | Reserved<br>Data Value Address 1. Reserved, RAZ/WI. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-12<br>DATAVADDR0 | Data Value Address 0<br>Since the DWT implements two comparators, the DATAVADDR0 field is restricted to values {0,1}. When the DATAVMATCH bit is asserted, this field defines the comparator number to use for linked address comparison.<br>If COMP0 is used as a data watchpoint and COMP1 as an address watchpoint, DATAVADDR0 must be set. |
| 11-10<br>DATAVSIZE | Data Value Size<br>For data value matching, this field defines the size of the required data comparison.<br>00b - Byte.<br>01b - Halfword.<br>10b - Word.<br>11b - Reserved. Any attempts to use this value results in UNPREDICTABLE behavior. |
| 9<br>— | Reserved<br>Data Value Address 1. Reserved, RAZ/WI. |
| 8<br>DATAVMATCH | Data Value Match<br>When this field is 1, it enables data value comparison. For this implementation, COMP0 supports address or data value comparisons; COMP1 only supports address comparisons.<br>0b - Perform address comparison.<br>1b - Perform data value comparison. |
| 7-4<br>— | Reserved<br>Data Value Address 1. Reserved, RAZ/WI. |
| 3-0<br>FUNCTION | Function<br>Selects the action taken on a comparator match. If COMP0 is used for a data value and COMP1 for an address value, then FCT1[FUNCTION] must be set to zero. For this configuration, MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses.<br>0000b - Disabled.<br>0001-0011b - Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.<br>0100b - Instruction fetch.<br>0101b - Data operand read.<br>0110b - Data operand write.<br>0111b - Data operand (read + write).<br>1000-1111b - Reserved. Any attempts to use this value results in UNPREDICTABLE behavior. |

## 14.3.2.6 DWT Comparator Function Register 1 (FCT1)

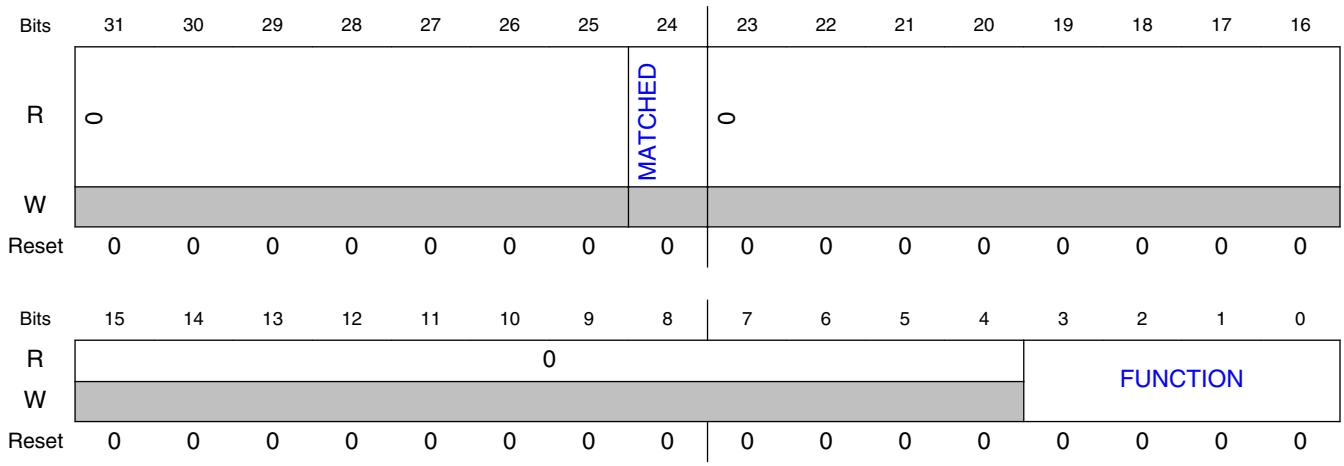
### 14.3.2.6.1 Offset

| Register | Offset |
|---|---|
| FCT1 | 38h |

### 14.3.2.6.2 Function

The FCTn registers control the operation of comparator n. Since the DWT only supports data value comparisons on comparator 0, there are several fields in the FCT1 register that are RAZ/WI (bits 12, 11:10, 8).

### 14.3.2.6.3 Diagram




### 14.3.2.6.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MATCHED | Comparator match<br>If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.<br>0b - No match.<br>1b - Match occurred. |
| 23-4<br>— | Reserved |
| 3-0<br>FUNCTION | Function<br>Selects the action taken on a comparator match. If COMP0 is used for a data value and COMP1 for an address value, then FCT1[FUNCTION] must be set to zero. For this configuration, MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses.<br>0000b - Disabled.<br>0001-0011b - Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.<br>0100b - Instruction fetch.<br>0101b - Data operand read.<br>0110b - Data operand write.<br>0111b - Data operand (read + write).<br>1000-1111b - Reserved. Any attempts to use this value results in UNPREDICTABLE behavior. |

## 14.3.2.7 DWT Trace Buffer Control Register (TBCTRL)

### 14.3.2.7.1 Offset

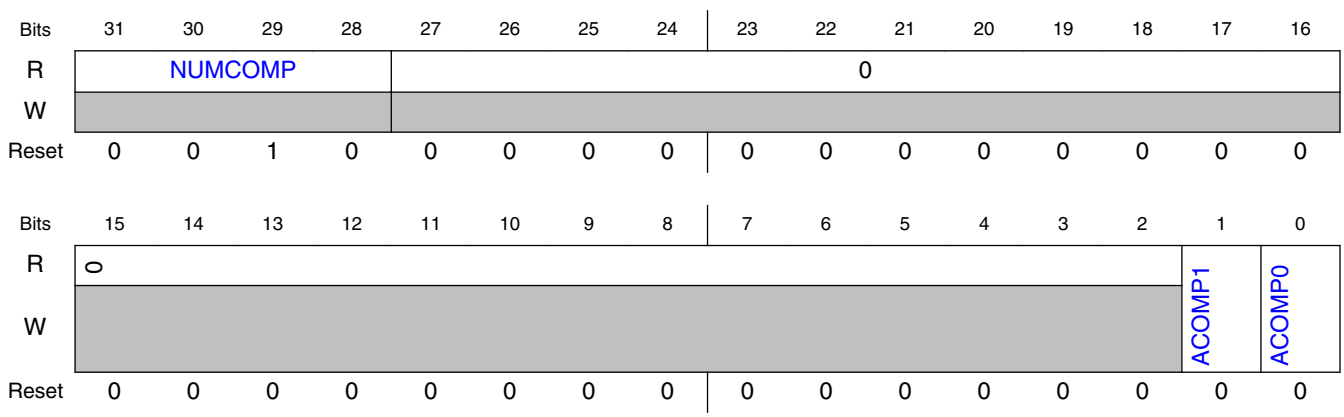
| Register | Offset |
|---|---|
| TBCTRL | 200h |

### 14.3.2.7.2 Function

The TBCTRL register defines how the watchpoint comparisons control the actual trace buffer operation.

Recall the MTB supports starting and stopping the program trace based on the watchpoint comparisons signaled via TSTART and TSTOP. The watchpoint comparison signals are enabled in the MTB's control logic by setting the appropriate enable bits, MTB_MASTER[TSTARTEN, TSTOPEN]. In the event of simultaneous assertion of both TSTART and TSTOP, TSTART takes priority.

### 14.3.2.7.3 Diagram




### 14.3.2.7.4 Fields

| Field | Function |
|---|---|
| 31-28<br>NUMCOMP | Number of Comparators<br>This read-only field specifies the number of comparators in the DWT. This implementation includes two registers. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 27-2<br>— | Reserved<br>RAZ/WI |
| 1<br>ACOMP1 | Action based on Comparator 1 match<br>When the FCT1[MATCHED] is set, it indicates COMP1 address compare has triggered and the trace buffer's recording state is changed.<br>0b - Trigger TSTOP based on the assertion of FCT1[MATCHED].<br>1b - Trigger TSTART based on the assertion of FCT1[MATCHED]. |
| 0<br>ACOMP0 | Action based on Comparator 0 match<br>When the FCT0[MATCHED] is set, it indicates COMP0 address compare has triggered and the trace buffer's recording state is changed. The assertion of FCT0[MATCHED] is caused by the following conditions:<br>• Address match in COMP0 when FCT0[DATAVMATCH] = 0<br>• Data match in COMP0 when FCT0[DATAVMATCH, DATAVADDR0] = {1,0}<br>• Data match in COMP0 and address match in COMP1 when FCT0[DATAVMATCH, DATAVADDR0] = {1,1}<br>0b - Trigger TSTOP based on the assertion of FCT0[MATCHED].<br>1b - Trigger TSTART based on the assertion of FCT0[MATCHED]. |

## 14.3.2.8 Device Configuration Register (DEVICECFG)

### 14.3.2.8.1 Offset

| Register | Offset |
|---|---|
| DEVICECFG | FC8h |

### 14.3.2.8.2 Function

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.2.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICECFG | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICECFG | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 14.3.2.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DEVICECFG | DEVICECFG<br>Hardwired to 0x0000_0000. |

### 14.3.2.9 Device Type Identifier Register (DEVICETYPID)

#### 14.3.2.9.1 Offset

| Register | Offset |
|---|---|
| DEVICETYPID | FCCh |

#### 14.3.2.9.2 Function

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.2.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICETYPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICETYPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 14.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DEVICETYPID | DEVICETYPID<br>Hardwired to 0x0000_0004. |

### 14.3.2.10 Peripheral ID Register (PERIPHID4 - PERIPHID3)

#### 14.3.2.10.1 Offset

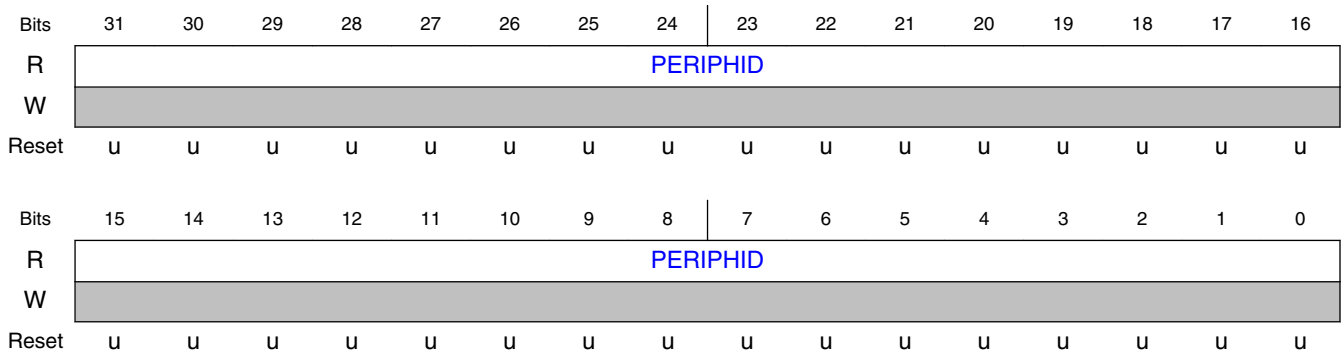
For a = 0 to 7:

| Register | Offset |
|---|---|
| PERIPHIDa | FC0h + (a × 4h) |

#### 14.3.2.10.2 Function

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.2.10.3 Diagram




### 14.3.2.10.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PERIPHID | PERIPHID<br>Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000. |

## 14.3.2.11 Component ID Register (COMPID0 - COMPID3)

### 14.3.2.11.1 Offset

For a = 0 to 3:

| Register | Offset |
|---|---|
| COMPIDa | FF0h + (a × 4h) |

### 14.3.2.11.2 Function

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.2.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COMPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COMPID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 14.3.2.11.4 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPID | Component ID<br>Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1. |

## 14.3.3 ROM register descriptions

### 14.3.3.1 ROM Memory map

The System ROM Table registers are also mapped into a sparsely-populated 4 KB address space.

For core configurations like that supported by Cortex-M0+, ARM recommends that a debugger identifies and connects to the debug components using the CoreSight debug infrastructure.

ARM recommends that a debugger follows the flow as shown in the following figure to discover the components in the CoreSight debug infrastructure. In this case, a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.

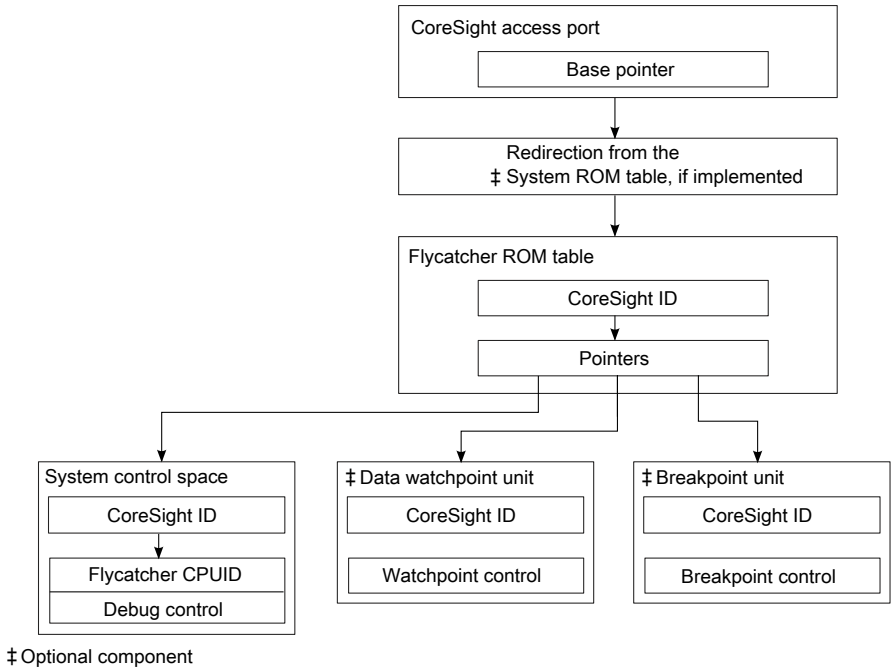



**Figure 14-3. CoreSight discovery process**

## ROM base address: F000_2000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - Ch | Entry (ENTRY0 - ENTRY3) | 32 | RO | See description. |
| 10h | End of Table Marker Register (TABLEMARK) | 32 | RO | 0000_0000h |
| FCCh | System Access Register (SYSACCESS) | 32 | RO | 0000_0001h |
| FD0h | Peripheral ID Register (PERIPHID4) | 32 | RO | See description. |
| FD4h | Peripheral ID Register (PERIPHID5) | 32 | RO | See description. |
| FD8h | Peripheral ID Register (PERIPHID6) | 32 | RO | See description. |
| FDCh | Peripheral ID Register (PERIPHID7) | 32 | RO | See description. |
| FE0h | Peripheral ID Register (PERIPHID0) | 32 | RO | See description. |
| FE4h | Peripheral ID Register (PERIPHID1) | 32 | RO | See description. |
| FE8h | Peripheral ID Register (PERIPHID2) | 32 | RO | See description. |
| FECh | Peripheral ID Register (PERIPHID3) | 32 | RO | See description. |
| FF0h - FFCh | Component ID Register (COMPID0 - COMPID3) | 32 | RO | See description. |

## 14.3.3.2 Entry (ENTRY0 - ENTRY3)

### 14.3.3.2.1 Offset

For a = 0 to 3:

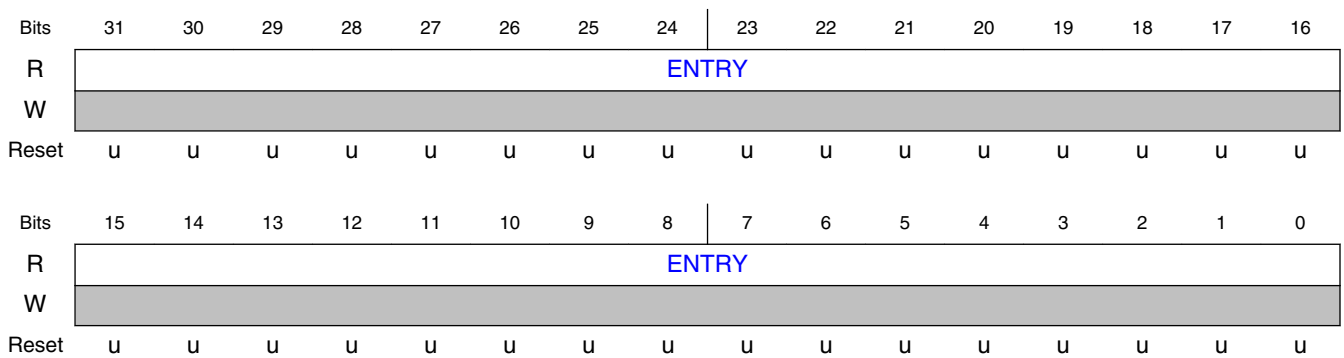
| Register | Offset |
|---|---|
| ENTRYa | 0h + (a × 4h) |

### 14.3.3.2.2 Function

The System ROM Table begins with "n" relative 32-bit addresses, one for each debug component present in the device and terminating with an all-zero value signaling the end of the table at the "n+1"-th value.

It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.3.2.3 Diagram




### 14.3.3.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ENTRY | ENTRY<br>Entry*n* values are hardwired: Entry 0 (MTB) is hardwired to 0xFFFF_E003, Entry 1 (DWT) to 0xFFFF_F003, Entry 2 (CM0PCTI) to 0x0000_4003, Entry 3 (CM0+ ROM Table) to 0xF00F_D003 |

## 14.3.3.3 End of Table Marker Register (TABLEMARK)

### 14.3.3.3.1 Offset

| Register | Offset |
|---|---|
| TABLEMARK | 10h |

### 14.3.3.3.2 Function

This register indicates end of table marker. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.3.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MARK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MARK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 14.3.3.3.4 Fields

| Field | Function |
|---|---|
| 31-0 | MARK |
| MARK | Hardwired to 0x0000_0000 |

## 14.3.3.4 System Access Register (SYSACCESS)

#### 14.3.3.4.1 Offset

| Register | Offset |
|---|---|
| SYSACCESS | FCCh |

#### 14.3.3.4.2 Function

This register indicates system access. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.3.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSACCESS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSACCESS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### 14.3.3.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>SYSACCESS | SYSACCESS<br>Hardwired to 0x0000_0001 |

### 14.3.3.5 Peripheral ID Register (PERIPHID4 - PERIPHID3)

#### 14.3.3.5.1 Offset

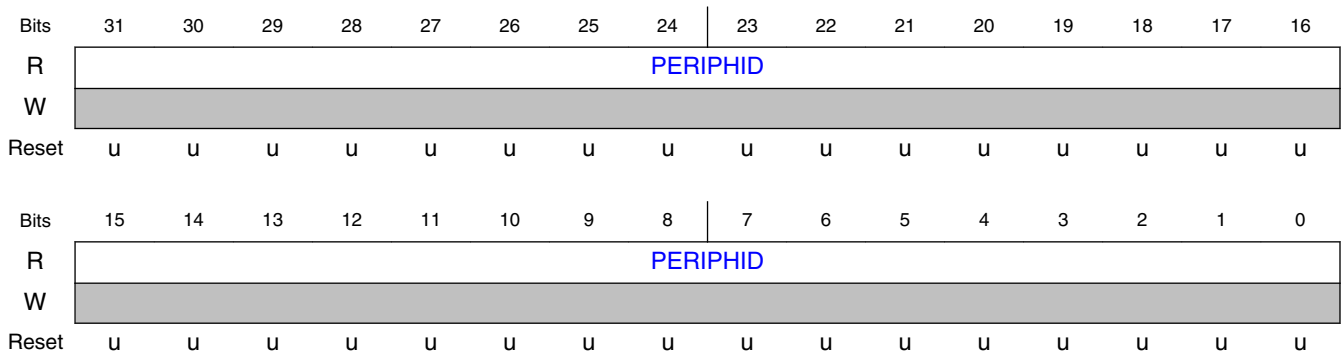
For a = 0 to 7:

| Register | Offset |
|---|---|
| PERIPHIDa | FC0h + (a × 4h) |

#### 14.3.3.5.2 Function

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

#### 14.3.3.5.3 Diagram




#### 14.3.3.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PERIPHID | PERIPHID<br>Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000. |

### 14.3.3.6 Component ID Register (COMPID0 - COMPID3)

#### 14.3.3.6.1 Offset

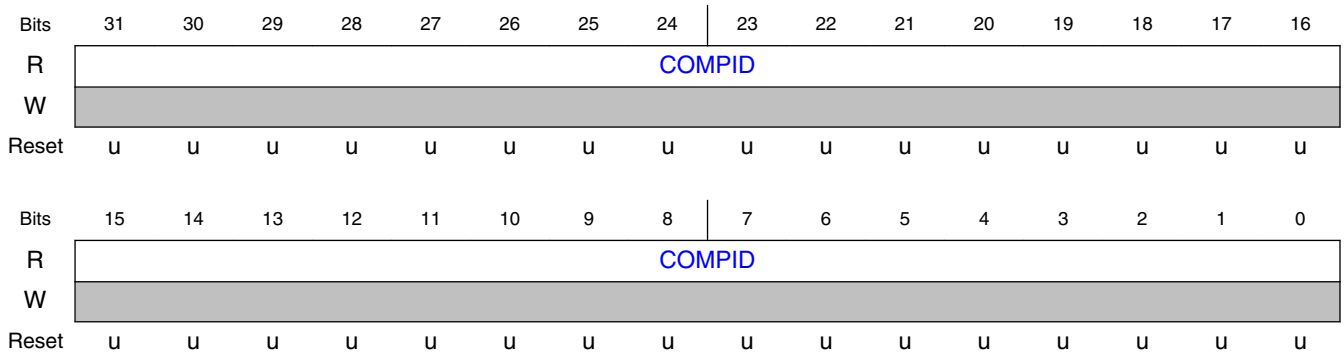
For a = 0 to 3:

| Register | Offset |
|---|---|
| COMPIDa | FF0h + (a × 4h) |

#### 14.3.3.6.2 Function

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

### 14.3.3.6.3 Diagram




### 14.3.3.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPID | Component ID<br>Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0010; ID2 to 0x0000_0005; ID3 to 0x0000_00B1. |

# Chapter 15
# JTAG Controller (JTAGC)

## 15.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

### 15.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

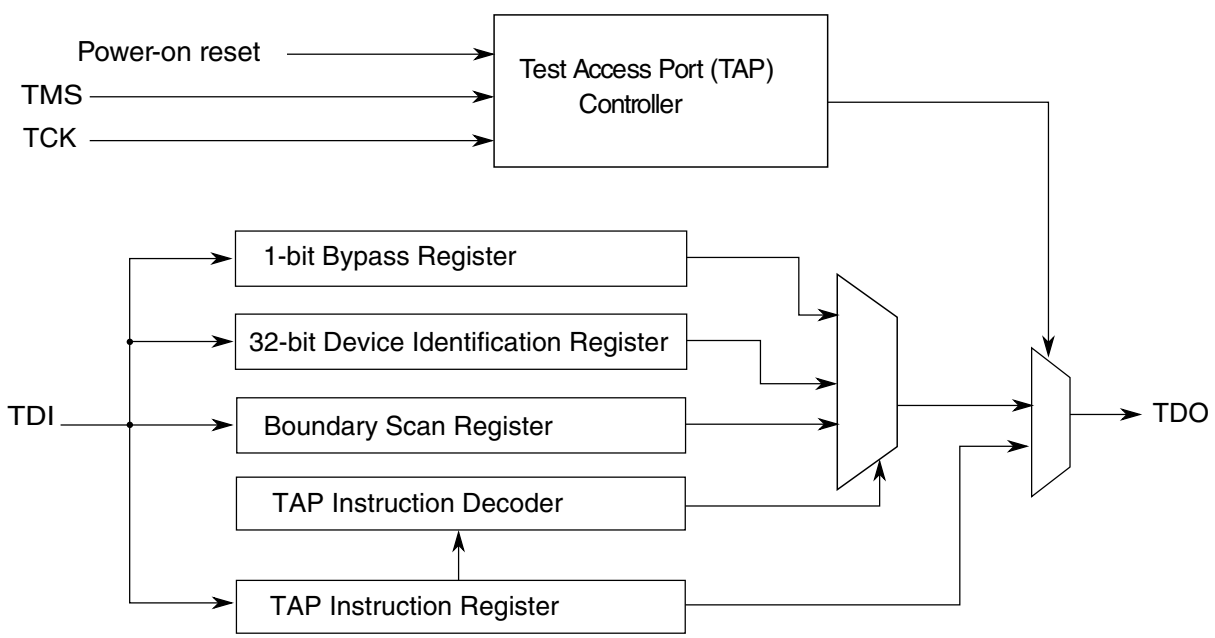



**Figure 15-1. JTAG (IEEE 1149.1) block diagram**

### 15.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to JTAGC block instructions for a list of supported instructions.
- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

### 15.1.3 Modes of operation

The JTAGC block uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

#### 15.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

#### 15.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the following instructions are active:

- BYPASS
- HIGHZ
- CLAMP

The functionality of each test mode is explained in more detail in JTAGC block instructions.

#### 15.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

## 15.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

**Table 15-1. JTAG signal properties**

| Name | I/O | Function | Reset State |
|---|---|---|---|
| TCK | Input | Test Clock | Weak pulldown |
| TDI | Input | Test Data In | Weak pullup |
| TDO | Output | Test Data Out | High Z[1] |
| TMS | Input | Test Mode Select | Weak pullup |

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

### 15.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

### 15.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

### 15.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

### 15.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

## 15.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

## 15.3.1 Instruction register

The JTAGC block uses a 4-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0b1, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

| | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| R | 0 | 0 | 0 | 1 |
| W | Instruction Code | | | |
| Reset: | 0 | 0 | 0 | 1 |

**Figure 15-2. Instruction register**

## 15.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the following instructions are active:

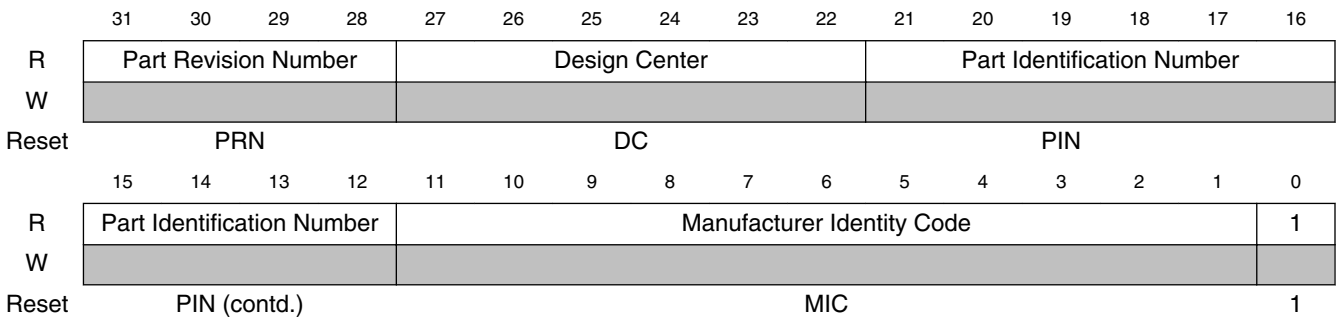
- BYPASS
- HIGHZ
- CLAMP

After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

## 15.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is

active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.




The following table describes the device identification register functions.

**Table 15-2. Device identification register field descriptions**

| Field | Description |
|---|---|
| PRN | Part Revision Number. Contains the revision number of the part. Value is cccc. |
| DC | Design Center. Indicates the design center. Value is 0x2C. |
| PIN | Part Identification Number. Contains the part number of the device. Value is 0b0000100000. |
| MIC | Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E. |
| IDCODE ID | IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1. |

### 15.3.4 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

## 15.4 Functional description

This section explains the JTAGC functional description.

### 15.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

### 15.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

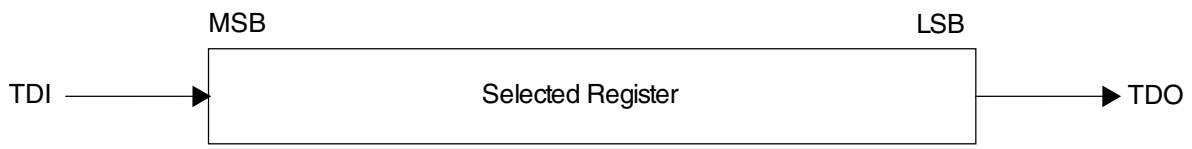
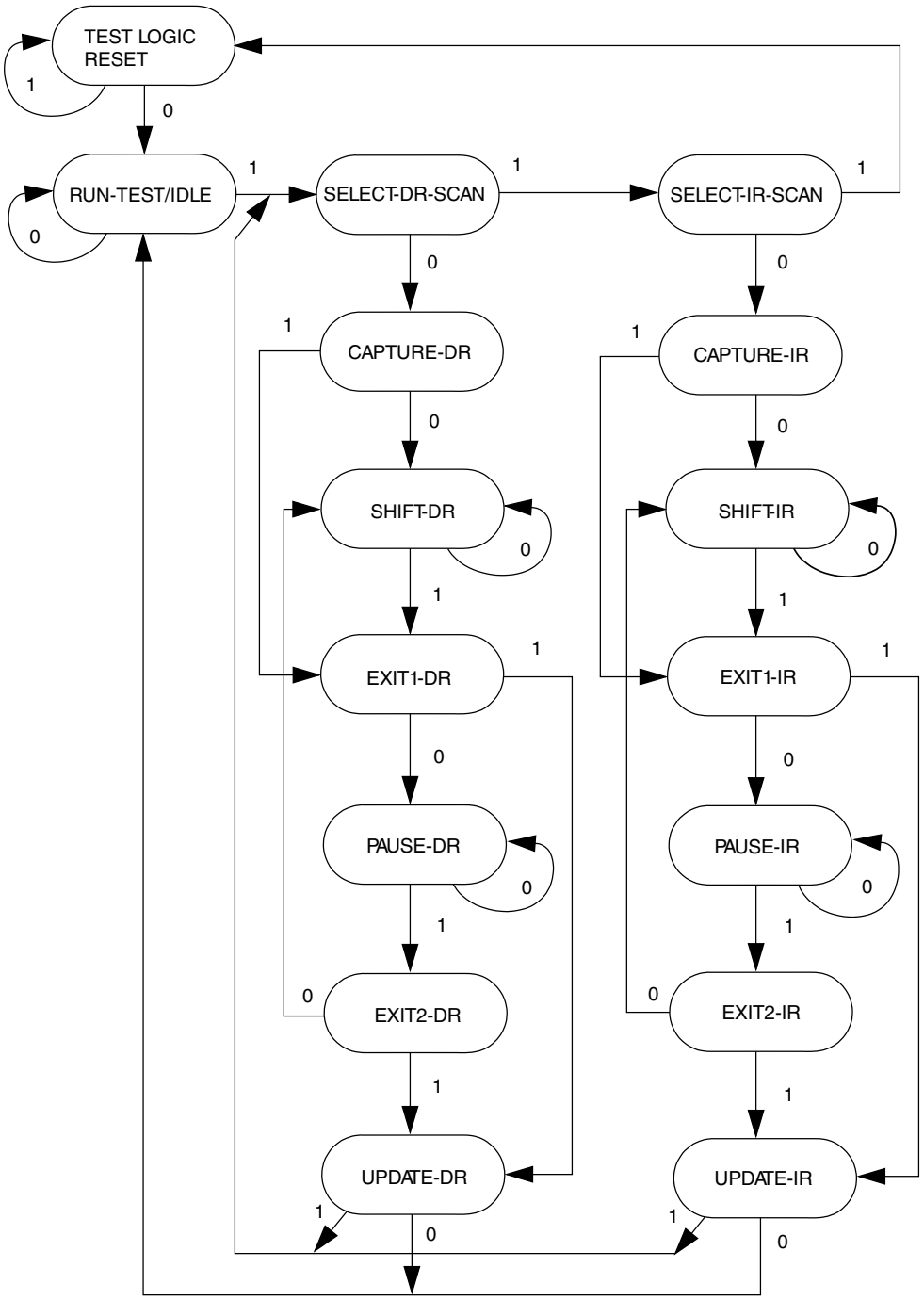



**Figure 15-3. Shifting data through a register**

### 15.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.




The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

**Figure 15-4. IEEE 1149.1-2001 TAP controller finite state machine**

## 15.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

### 15.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

## 15.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

**Table 15-3. 4-bit JTAG instructions**

| Instruction | Code[3:0] | Instruction summary |
|---|---|---|
| IDCODE | 0000 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 0010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 0011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 0100 | Selects boundary scan register and applies preloaded values to output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| Factory debug reserved | 0101 | Intended for factory debug only |
| Factory debug reserved | 0110 | Intended for factory debug only |
| Factory debug reserved | 0111 | Intended for factory debug only |
| ARM JTAG-DP Reserved | 1000 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| HIGHZ | 1001 | Selects bypass register and three-states all output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| ARM JTAG-DP Reserved | 1010 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |

*Table continues on the next page...*

**Table 15-3. 4-bit JTAG instructions (continued)**

| Instruction | Code[3:0] | Instruction summary |
|---|---|---|
| ARM JTAG-DP Reserved | 1011 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| CLAMP | 1100 | Selects bypass register and applies preloaded values to output pins.<br>**NOTE:** Execution of this instruction asserts functional reset. |
| ARM JTAG-DP Reserved | 1110 | This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information. |
| BYPASS | 1111 | Selects bypass register for data operations |

### 15.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

### 15.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

#### 15.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

#### 15.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

#### 15.4.4.5 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

#### 15.4.4.6 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

#### 15.4.4.7 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

### 15.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

## 15.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed

# Chapter 16
# Memory Map

## 16.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in one 32-bit contiguous addressable memory space. This chapter describes the memory and peripheral locations within that memory space.

## 16.2 System memory map

The following table shows the high-level device memory map. This map provides the complete architectural address space definition for the various sections. Based on the physical sizes of the memories and peripherals, the actual address regions used may be smaller.

The system memory map includes aliased address spaces that are intended for specific purposes.

- The region of memory from 0x1000_0000 to 0x1FFF_FFFF is aliased to regions of the External RAM and External Device areas of the system memory. This allows faster execution from these areas because the processor core thinks that they are in the Code area of memory. Additionally, by mapping these regions to the Code space, these devices can be cached in the I/D Caches. The FlexBus controller is the only memory peripheral assigned to this space.

**Table 16-1. System memory map**

| Start Address | Stop Address | Cortex-M4 Map | Cortex-M0+ Map | Access | Cached |
|---|---|---|---|---|---|
| 0x0000_0000 | 0x000F_FFFF | M4 - 1 MB flash (including exception vectors in first 1024 bytes) | M4 - 1 MB flash (including exception vectors in first 1024 bytes) | All Masters[1] | Yes |
| 0x0010_0000 | 0x00FF_FFFF | Reserved | Reserved | — | No |

*Table continues on the next page...*

## Table 16-1. System memory map (continued)

| Start Address | Stop Address | Cortex-M4 Map | Cortex-M0+ Map | Access | Cached |
|---|---|---|---|---|---|
| 0x0100_0000 | 0x0103_FFFF | M0+ - 256 KB flash (including exception vectors in first 1024 bytes) | M0+ - 256 KB flash (including exception vectors in first 1024 bytes) | All Masters | Yes |
| 0x0104_0000 | 0x07FF_FFFF | Reserved | Reserved | — | No |
| 0x0800_0000 | 0x0800_FFFF | Cortex M4 ITCM SRAM - 64 KB | Cortex M4 ITCM SRAM - 64 KB | All Masters | No |
| 0x0801_0000 | 0x087F_FFFF | Reserved | Reserved | — | No |
| 0x0880_0000 | 0x0880_BFFF | Boot ROM - 48 KB | Boot ROM - 48 KB | All Masters | No |
| 0x0880_C000 | 0x08FF_FFFF | Reserved | Reserved | — | No |
| 0x0900_0000 | 0x0901_FFFF | Cortex M0+ TCM SRAM - 128 KB | Cortex M0+ TCM SRAM - 128 KB | All Masters | No |
| 0x0902_0000 | 0x0FFF_FFFF | Reserved | Reserved | — | No |
| 0x1000_0000 | 0x1FFF_FFFF | FlexBus Execution Alias | FlexBus Execution Alias | All Masters | Yes |
| 0x2000_0000 | 0x2002_FFFF | Cortex M4 DTCM SRAM - 192 KB | Cortex M4 DTCM SRAM - 192 KB | All Masters | N/A |
| 0x2003_0000 | 0x3FFF_FFFF | Reserved | Reserved | — | N/A |
| 0x4000_0000 | 0x4007_FFFF | Cortex M4 AIPS0 (M4 Peripherals) | Cortex M4 AIPS0 (M4 Peripherals) | All Masters | N/A |
| 0x4008_0000 | 0x40FF_FFFF | Reserved | Reserved | — | N/A |
| 0x4100_0000 | 0x4107_FFFF | Cortex M0+ AIPS0 (M0+ Peripherals) | Cortex M0+ AIPS0 (M0+ Peripherals) | All Masters | N/A |
| 0x4108_0000 | 0x47FF_FFFF | Reserved | Reserved | — | N/A |
| 0x4800_0000 | 0x4800_0FFF | FlexRAM - 4 KB | FlexRAM - 4 KB | All Masters | N/A |
| 0x4800_1000 | 0x4800_FFFF | Reserved | Reserved | — | N/A |
| 0x4801_0000 | 0x4801_07FF | USB SRAM - 2 KB | USB SRAM - 2 KB | All Masters | N/A |
| 0x4801_0800 | 0x4801_FFFF | Reserved | Reserved | — | N/A |
| 0x4802_0000 | 0x4802_0FFF | Rapid GPIO (PTA/B/C/D) | Reserved | All Masters | N/A |
| 0x4802_1000 | 0x9FFF_FFFF | Reserved | Reserved | — | N/A |
| 0xA000_0000 | 0xAFFF_FFFF | FlexBus (External Memory) | FlexBus (External Memory) | All Masters | N/A |
| 0xB000_0000 | 0xDFFF_FFFF | Reserved | Reserved | — | N/A |
| 0xE000_0000 | 0xE0FF_FFFF | PPB - ARM SYS Modules | PPB - ARM SYS Modules | Cortex Core[2] | N/A |
| 0xE100_0000 | 0xEFFF_FFFF | Reserved | Reserved | — | N/A |
| 0xF000_0000 | 0xF0FF_FFFF | PPB - NXP SYS Modules | PPB - NXP SYS Modules | Cortex Core[2] | N/A |
| 0xF100_0000 | 0xF7FF_FFFF | Reserved | Reserved | — | N/A |
| 0xF800_0000 | 0xF8FF_FFFF | Reserved | Cortex M0+ IO Port (PTE) | Cortex M0+ | N/A |
| 0xF900_0000 | 0xFFFF_FFFF | Reserved | Reserved | — | N/A |

1. The CM0+ in the 0x0000_0000 to 0x0000_0FFF will remap to other locations based on boot sources.
2. The Private Peripheral Bus (PPB) is only accessible from the core to which it is attached.

## 16.3 Flash memory map

The flash memory and the flash registers are located at different base addresses as shown in the following figure. The base address for each is specified in System memory map .

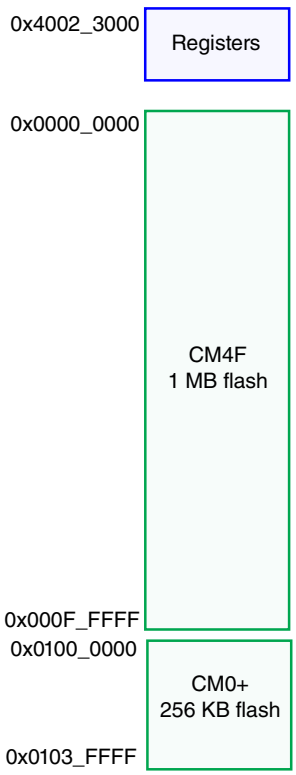



**Figure 16-1. Flash address in Cortex-M4F/M0+ map**

The on-chip Flash includes two portions. One is with 1 MB size mainly for CM4F subsystem beginning at address 0x0000_0000. Another is with 256 KB size mainly for Cortex-M0+ subsystem beginning at address 0x0100_0000.

Both Cortex-M4F/M0+ cores can access the entire amount of flash on the device with one exception – CM0+ cannot access lowest 4 KB region of CM4F flash starting from address 0x0000_0000. Each CPU accesses to the flash associated with the other CPU has lower performance.

Accesses to the flash memory ranges outside the amount of Flash on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master. Read collision events in which flash memory is accessed while a flash memory resource is being manipulated by a flash command also generates a bus error response.

## 16.4 Flash access control introduction

The Flash Access Control (FAC) is a configurable memory protection scheme optimized to allow end users to utilize software libraries while offering programmable restrictions to these libraries. The flash memory is divided into equal size segments that provide protection to proprietary software libraries. The protection of these segments is controlled as the FAC provides a cycle-by-cycle evaluation of the access rights for each transaction routed to the on-chip flash memory. Configurability allows an increasing number of protected segments while supporting two levels of vendors adding their proprietary software to a device.

Flash access control aligns to the three privilege levels supported by ARM Cortex-M family products where the most secure state - supervisor/privileged-secure - aligns to the execute-only and supervisor-only access control. The unsecure state of user-non-secure aligns to no access control states set, and the mid-level state where user-secure aligns to using the access control of execute-only.

Control for this protection scheme is implemented in Program Index NVM locations and is configurable through a Program Index flash command operations. The NVM locations controlling the FAC are affected by Erase All Blocks flash command and debug interface initiated mass erase operations.

## 16.5 SRAM memory map

See the following table for the SRAM memory map. The enabe and disable of each SRAM banks are controlled by SRAMLPR and SRAMDSR in MSMC module.

**Table 16-2. SRAM memory map**

| Start Address | Stop Address | Cortex-M4 Map | Cortex-M0+ Map | SMC0_SRAMLPR SMC0_SRAMDSR | SMC1_SRAMLPR SMC1_SRAMDSR |
|---|---|---|---|---|---|
| 0x0800_0000 | 0x0800_1FFF | CM4F ITCM SRAML | — | bit0 | — |
| 0x0800_2000 | 0x0800_3FFF | CM4F ITCM SRAML | — | bit1 | — |
| 0x0800_4000 | 0x0800_7FFF | CM4F ITCM SRAML | — | bit2 | — |
| 0x0800_8000 | 0x0800_FFFF | CM4F ITCM SRAML | — | bit3 | — |
| 0x0900_0000 | 0x0900_1FFF | — | CM0+ TCM | — | bit0 |
| 0x0900_2000 | 0x0900_3FFF | — | CM0+ TCM | — | bit1 |
| 0x0900_4000 | 0x0900_7FFF | — | CM0+ TCM | — | bit2 |

*Table continues on the next page...*

**Table 16-2. SRAM memory map (continued)**

| Start Address | Stop Address | Cortex-M4 Map | Cortex-M0+ Map | SMC0_SRAMLPR<br>SMC0_SRAMDSR | SMC1_SRAMLPR<br>SMC1_SRAMDSR |
|---|---|---|---|---|---|
| 0x0900_8000 | 0x0900_FFFF | — | CM0+ TCM | — | bit3 |
| 0x0901_0000 | 0x0901_7FFF | — | CM0+ TCM | — | bit4 |
| 0x0901_8000 | 0x0901_FFFF | — | CM0+ TCM | — | bit5 |
| 0x2000_0000 | 0x2000_1FFF | CM4F DTCM SRAMU | — | bit16 | — |
| 0x2000_2000 | 0x2000_3FFF | CM4F DTCM SRAMU | — | bit17 | — |
| 0x2000_4000 | 0x2000_7FFF | CM4F DTCM SRAMU | — | bit18 | — |
| 0x2000_8000 | 0x2000_FFFF | CM4F DTCM SRAMU | — | bit19 | — |
| 0x2001_0000 | 0x2001_7FFF | CM4F DTCM SRAMU | — | bit20 | — |
| 0x2001_8000 | 0x2001_FFFF | CM4F DTCM SRAMU | — | bit21 | — |
| 0x2002_0000 | 0x2002_7FFF | CM4F DTCM SRAMU | — | bit22 | — |
| 0x2002_8000 | 0x2002_FFFF | CM4F DTCM SRAMU | — | bit23 | — |

## 16.6 Peripheral bridge memory maps

This section covers the peripheral bridges included in this device.

### 16.6.1 ARM Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge memory maps

The peripheral memory map is accessible via one slave port on the crossbar in the 0x4100_0000 – 0x4107_FFFF region. The device implements one peripheral bridge that defines a 512 KB address space.

### 16.6.1.1 ARM Cortex-M0+/RISC-V ZERO-RISCY Peripheral Bridge 1 (AIPS-Lite 1) memory map

**Table 16-3. Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge 1 slot assignments**

| Base address | Slot | Module |
|---|---|---|
| 0x4100_0000 | 0 | — |
| 0x4100_1000 | 1 | — |
| 0x4100_2000 | 2 | — |
| 0x4100_3000 | 3 | — |
| 0x4100_4000 | 4 | — |
| 0x4100_5000 | 5 | — |
| 0x4100_6000 | 6 | — |
| 0x4100_7000 | 7 | — |
| 0x4100_8000 | 8 | Direct Memory Access 1 (DMA1) controller |
| 0x4100_9000 | 9 | Direct Memory Access 1 (DMA1) controller transfer control descriptors |
| 0x4100_A000 | 10 | — |
| 0x4100_B000 | 11 | — |
| 0x4100_C000 | 12 | — |
| 0x4100_D000 | 13 | — |
| 0x4100_E000 | 14 | — |
| 0x4100_F000 | 15 | IO Port Alias |
| 0x4101_0000 | 16 | — |
| 0x4101_1000 | 17 | — |
| 0x4101_2000 | 18 | — |
| 0x4101_3000 | 19 | — |
| 0x4101_4000 | 20 | — |
| 0x4101_5000 | 21 | — |
| 0x4101_6000 | 22 | — |
| 0x4101_7000 | 23 | — |
| 0x4101_8000 | 24 | — |
| 0x4101_9000 | 25 | — |
| 0x4101_A000 | 26 | — |
| 0x4101_B000 | 27 | Semaphore Unit (SEMA42) |
| 0x4101_C000 | 28 | — |
| 0x4101_D000 | 29 | — |
| 0x4101_E000 | 30 | — |
| 0x4101_F000 | 31 | ZERORISCY Event Unit |
| 0x4102_0000 | 32 | Multi System Mode Controller (MSMC_SMC1) |
| 0x4102_1000 | 33 | Direct Memory Access Mutiplexer 1 (DMAMUX1) |
| 0x4102_2000 | 34 | Interrupt Multiplexer 0 (INTMUX0) |
| 0x4102_3000 | 35 | Low Leakage Wake-up Unit 1 (LLWU1) |
| 0x4102_4000 | 36 | Messaging Unit - Side B (MU-B) |

*Table continues on the next page...*

## Table 16-3. Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge 1 slot assignments (continued)

| Base address | Slot | Module |
|---|---|---|
| 0x4102_5000 | 37 | Trigger Multiplexer 1 (TRGMUX1) |
| 0x4102_6000 | 38 | Watchdog 1 (WDOG1) |
| 0x4102_7000 | 39 | Peripheral Clock Controller 1(PCC1) |
| 0x4102_8000 | 40 | Cryptographic Acceleration Unit (CAU3) |
| 0x4102_9000 | 41 | True Random Number Generator (TRNG) |
| 0x4102_A000 | 42 | Low-Power Periodic Interrupt Timer 1 (LPIT1) |
| 0x4102_B000 | 43 | Low-Power Timer 2 (LPTMR2) |
| 0x4102_C000 | 44 | Time Stamp Timer B (TSTMRB) |
| 0x4102_D000 | 45 | Timer / Pulse Width Modulation Module 3 (TPM3) - 2 channel |
| 0x4102_E000 | 46 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) |
| 0x4102_F000 | 47 | 2.4GHz Radio (RF2.4G) - RSIM |
| 0x4103_0000 | 48 | 2.4GHz Radio (RF2.4G) - XCVR |
| 0x4103_1000 | 49 | — |
| 0x4103_2000 | 50 | 2.4GHz Radio (RF2.4G) - Bluetooth Link layer |
| 0x4103_3000 | 51 | 2.4GHz Radio (RF2.4G) - Generic Link layer |
| 0x4103_4000 | 52 | 2.4GHz Radio (RF2.4G) - IEEE 802.15.4 Link Layer |
| 0x4103_5000 | 53 | Low-Power Serial Peripheral Interface 3 (LPSPI3) |
| 0x4103_6000 | 54 | Low-Power Universal Asynchronous Receive / Transmit 3 (LPUART3) |
| 0x4103_7000 | 55 | PORTE |
| 0x4103_8000 | 56 | Low-Power Comparator 1 (LPCMP1) |
| 0x4103_9000 | 57 | — |
| 0x4103_A000 | 58 | — |
| 0x4103_B000 | 59 | — |
| 0x4103_C000 | 60 | — |
| 0x4103_D000 | 61 | — |
| 0x4103_E000 | 62 | — |
| 0x4103_F000 | 63 | — |
| 0x4104_0000 | 64 | — |
| 0x4104_1000 | 65 | — |
| 0x4104_2000 | 66 | — |
| 0x4104_3000 | 67 | — |
| 0x4104_4000 | 68 | — |
| 0x4104_5000 | 69 | — |
| 0x4104_6000 | 70 | — |
| 0x4104_7000 | 71 | — |
| 0x4104_8000 | 72 | — |
| 0x4104_9000 | 73 | — |
| 0x4104_A000 | 74 | — |

*Table continues on the next page...*

## Table 16-3. Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge 1 slot assignments (continued)

| Base address | Slot | Module |
|---|---|---|
| 0x4104_B000 | 75 | — |
| 0x4104_C000 | 76 | — |
| 0x4104_D000 | 77 | — |
| 0x4104_E000 | 78 | — |
| 0x4104_F000 | 79 | — |
| 0x4105_0000 | 80 | — |
| 0x4105_1000 | 81 | — |
| 0x4105_2000 | 82 | — |
| 0x4105_3000 | 83 | — |
| 0x4105_4000 | 84 | — |
| 0x4105_5000 | 85 | — |
| 0x4105_6000 | 86 | — |
| 0x4105_7000 | 87 | — |
| 0x4105_8000 | 88 | — |
| 0x4105_9000 | 89 | — |
| 0x4105_A000 | 90 | — |
| 0x4105_B000 | 91 | — |
| 0x4105_C000 | 92 | — |
| 0x4105_D000 | 93 | — |
| 0x4105_E000 | 94 | — |
| 0x4105_F000 | 95 | — |
| 0x4106_0000 | 96 | — |
| 0x4106_1000 | 97 | — |
| 0x4106_2000 | 98 | — |
| 0x4106_3000 | 99 | — |
| 0x4106_4000 | 100 | — |
| 0x4106_5000 | 101 | — |
| 0x4106_6000 | 102 | — |
| 0x4106_7000 | 103 | — |
| 0x4106_8000 | 104 | — |
| 0x4106_9000 | 105 | — |
| 0x4106_A000 | 106 | — |
| 0x4106_B000 | 107 | — |
| 0x4106_C000 | 108 | — |
| 0x4106_D000 | 109 | — |
| 0x4106_E000 | 110 | — |
| 0x4106_F000 | 111 | — |
| 0x4107_0000 | 112 | — |

*Table continues on the next page...*

**Table 16-3. Cortex-M0+/RISC-V ZERO-RISCY peripheral bridge 1 slot assignments (continued)**

| Base address | Slot | Module |
|---|---|---|
| 0x4107_1000 | 113 | — |
| 0x4107_2000 | 114 | — |
| 0x4107_3000 | 115 | — |
| 0x4107_4000 | 116 | — |
| 0x4107_5000 | 117 | — |
| 0x4107_6000 | 118 | — |
| 0x4107_7000 | 119 | — |
| 0x4107_8000 | 120 | — |
| 0x4107_9000 | 121 | — |
| 0x4107_A000 | 122 | — |
| 0x4107_B000 | 123 | — |
| 0x4107_C000 | 124 | — |
| 0x4107_D000 | 125 | — |
| 0x4107_E000 | 126 | — |
| 0x4107_F000 | 127 | — |

## 16.6.2 ARM Cortex-M4RISC-V RI5CY peripheral bridge memory maps

The peripheral memory map is accessible via one slave port on the crossbar in the 0x4000_0000 – 0x4007_FFFF region. The device implements one peripheral bridge that defines a 512 KB address space.

### 16.6.2.1 ARM Cortex M4/RISC-V RI5CY Peripheral Bridge 0 (AIPS-Lite 0) memory map

**Table 16-4. Cortex-M4/RISC-V RI5CY peripheral bridge 0 slot assignments**

| Base address | Slot | Module |
|---|---|---|
| 0x4000_0000 | 0 | — |
| 0x4000_1000 | 1 | Miscellaneous System Control Module (MSCM) |
| 0x4000_2000 | 2 | — |
| 0x4000_3000 | 3 | — |
| 0x4000_4000 | 4 | Coretex-M4 crossbar switch |
| 0x4000_5000 | 5 | — |
| 0x4000_6000 | 6 | — |

*Table continues on the next page...*

## Table 16-4. Cortex-M4/RISC-V RI5CY peripheral bridge 0 slot assignments (continued)

| Base address | Slot | Module |
|---|---|---|
| 0x4000_7000 | 7 | — |
| 0x4000_8000 | 8 | Direct Memory Access 0 (DMA0) controller |
| 0x4000_9000 | 9 | Direct Memory Access 0 (DMA0) controller transfer control descriptors |
| 0x4000_A000 | 10 | — |
| 0x4000_B000 | 11 | — |
| 0x4000_C000 | 12 | External Bus Interface (FlexBus) |
| 0x4000_D000 | 13 | — |
| 0x4000_E000 | 14 | — |
| 0x4000_F000 | 15 | — |
| 0x4001_0000 | 16 | — |
| 0x4001_1000 | 17 | — |
| 0x4001_2000 | 18 | — |
| 0x4001_3000 | 19 | — |
| 0x4001_4000 | 20 | Resource Domain Controller (RDC) MGR and MDAC |
| 0x4001_5000 | 21 | Resource Domain Controller (RDC) PAC |
| 0x4001_6000 | 22 | Resource Domain Controller (RDC) MRC |
| 0x4001_7000 | 23 | Resource Domain Controller (RDC) MRC |
| 0x4001_8000 | 24 | — |
| 0x4001_9000 | 25 | — |
| 0x4001_A000 | 26 | — |
| 0x4001_B000 | 27 | Semaphore Unit (SEMA42) |
| 0x4001_C000 | 28 | — |
| 0x4001_D000 | 29 | — |
| 0x4001_E000 | 30 | — |
| 0x4001_F000 | 31 | — |
| 0x4002_0000 | 32 | Multi System Mode Controller (MSMC_SMC0) |
| 0x4002_1000 | 33 | Direct memory Access Mutliplexer 0 (DMAMUX0) |
| 0x4002_2000 | 34 | External Watchdog Monitor (EWM) |
| 0x4002_3000 | 35 | Flash Memory Module (FTFE) |
| 0x4002_4000 | 36 | Low Leakage Wake-up Unit 0 (LLWU0) |
| 0x4002_5000 | 37 | Messaging Unit - Side A (MUA) |
| 0x4002_6000 | 38 | System Integration Module (SIM) |
| 0x4002_7000 | 39 | USB Voltage Regulator (USBVREG) |
| 0x4002_8000 | 40 | System Power Management (SPM) |
| 0x4002_9000 | 41 | Tirgger Multiplexer 0 (TRGMUX0) |
| 0x4002_A000 | 42 | Watchdog 0 (WDOG0) |
| 0x4002_B000 | 43 | Peripheral Clock Controller 0 (PCC0) |
| 0x4002_C000 | 44 | System Clock Generator (SCG) |
| 0x4002_D000 | 45 | System Register File |

*Table continues on the next page...*

## Table 16-4. Cortex-M4/RISC-V RI5CY peripheral bridge 0 slot assignments (continued)

| Base address | Slot | Module |
|---|---|---|
| 0x4002_E000 | 46 | VBAT Register File |
| 0x4002_F000 | 47 | Cyclic Redundancy Check 0 (CRC0) |
| 0x4003_0000 | 48 | Low-Power Periodic Interrupt Timer 0 (LPIT0) |
| 0x4003_1000 | 49 | Real Time Clock (RTC) |
| 0x4003_2000 | 50 | Low-Power Timer 0 (LPTMR0) |
| 0x4003_3000 | 51 | Low-Power Timer 1 (LPTMR1) |
| 0x4003_4000 | 52 | Time Stamp Timer A (TSTMRA) |
| 0x4003_5000 | 53 | Timer / Pulse Width Modulator Module 0 (TPM0) - 6 channel |
| 0x4003_6000 | 54 | Timer / Pulse Width Modulator Module 1 (TPM1) - 2 channel |
| 0x4003_7000 | 55 | Timer / Pulse Width Modulator Module 2 (TPM2) - 6 channel |
| 0x4003_8000 | 56 | Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) |
| 0x4003_9000 | 57 | Flexible Input / Output (FlexIO) |
| 0x4003_A000 | 58 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) |
| 0x4003_B000 | 59 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) |
| 0x4003_C000 | 60 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) |
| 0x4003_D000 | 61 | Inter-IC Sound 0 (I2S0) |
| 0x4003_E000 | 62 | Secure Digital Host Controller 0 (SDHC0) |
| 0x4003_F000 | 63 | Low-Power Serial Peripheral Interface 0 (LPSPI0) |
| 0x4004_0000 | 64 | Low-Power Serial Peripheral Interface 1 (LPSPI1) |
| 0x4004_1000 | 65 | Low-Power Serial Peripheral Interface 2 (LPSPI2) |
| 0x4004_2000 | 66 | Low-Power Universal Asynchronous Receive / Transmit 0 (LPUART0) |
| 0x4004_3000 | 67 | Low-Power Universal Asynchronous Receive / Transmit 1 (LPUART1) |
| 0x4004_4000 | 68 | Low-Power Universal Asynchronous Receive / Transmit 2 (LPUART2) |
| 0x4004_5000 | 69 | Universal Serial Bus 0 (USB0) - Full Speed, Device Only |
| 0x4004_6000 | 70 | PORTA |
| 0x4004_7000 | 71 | PORTB |
| 0x4004_8000 | 72 | PORTC |
| 0x4004_9000 | 73 | PORTD |
| 0x4004_A000 | 74 | Low-Power Analog-to-Digital Converter 0 (LPADC0) |
| 0x4004_B000 | 75 | Low-Power Comparator 0 (LPCMP0) |
| 0x4004_C000 | 76 | Low Power Digital-to-Analog Converter 0 (LPDAC0) |
| 0x4004_D000 | 77 | Voltage Reference (VREF) |
| 0x4004_E000 | 78 | — |
| 0x4004_F000 | 79 | INTMUX0 (for RI5CY only) |
| 0x4005_0000 | 80 | — |
| 0x4005_1000 | 81 | — |
| 0x4005_2000 | 82 | — |
| 0x4005_3000 | 83 | — |
| 0x4005_4000 | 84 | — |

*Table continues on the next page...*

## Table 16-4. Cortex-M4/RISC-V RI5CY peripheral bridge 0 slot assignments (continued)

| Base address | Slot | Module |
|---|---|---|
| 0x4005_5000 | 85 | — |
| 0x4005_6000 | 86 | — |
| 0x4005_7000 | 87 | — |
| 0x4005_8000 | 88 | — |
| 0x4005_9000 | 89 | — |
| 0x4005_A000 | 90 | — |
| 0x4005_B000 | 91 | — |
| 0x4005_C000 | 92 | — |
| 0x4005_D000 | 93 | — |
| 0x4005_E000 | 94 | — |
| 0x4005_F000 | 95 | — |
| 0x4006_0000 | 96 | — |
| 0x4006_1000 | 97 | — |
| 0x4006_2000 | 98 | — |
| 0x4006_3000 | 99 | — |
| 0x4006_4000 | 100 | — |
| 0x4006_5000 | 101 | — |
| 0x4006_6000 | 102 | — |
| 0x4006_7000 | 103 | — |
| 0x4006_8000 | 104 | — |
| 0x4006_9000 | 105 | — |
| 0x4006_A000 | 106 | — |
| 0x4006_B000 | 107 | — |
| 0x4006_C000 | 108 | — |
| 0x4006_D000 | 109 | — |
| 0x4006_E000 | 110 | — |
| 0x4006_F000 | 111 | — |
| 0x4007_0000 | 112 | — |
| 0x4007_1000 | 113 | — |
| 0x4007_2000 | 114 | — |
| 0x4007_3000 | 115 | — |
| 0x4007_4000 | 116 | — |
| 0x4007_5000 | 117 | — |
| 0x4007_6000 | 118 | — |
| 0x4007_7000 | 119 | — |
| 0x4007_8000 | 120 | — |
| 0x4007_9000 | 121 | — |
| 0x4007_A000 | 122 | — |
| 0x4007_B000 | 123 | — |

*Table continues on the next page...*

**Table 16-4. Cortex-M4/RISC-V RI5CY peripheral bridge 0 slot assignments (continued)**

| Base address | Slot | Module |
|---|---|---|
| 0x4007_C000 | 124 | — |
| 0x4007_D000 | 125 | — |
| 0x4007_E000 | 126 | — |
| 0x4007_F000 | 127 | — |

## 16.7 Private Peripheral Bus (PPB) memory map

The Private Peripheral Bus (PPB) is part of the ARM bus architecture definition and provides access to select processor-local modules. These resources are only accessible from the core; other system masters do not have access to them. The bus is split into an internal Private Peripheral Bus (PPB) and an External Private Peripheral Bus (EPPB). The PPB provides access to tightly coupled ARM system resources such as the Instrumentation Trace Module (ITM), Data Watchpoint and Trace (DWT), Flash Patch Breakpoint (FPB), Memory Protection Unit (MPU), and the Nested Vectored Interrupt Control (NVIC).

The External Private Peripheral Bus (EPPB) is accessed through the Advanced Peripheral Bus (APB) interface. The EPPB is intended for CoreSight™-compatible debug and trace components such as the Trace Port Interface Unit (TPIU) and the Embedded Trace Macrocell (ETM) as well as vendor specific modules. The EPPB has a number of irregular limitations that make it less useful for regular system peripherals.

The following sections detail the Private Peripheral Bus memory maps associated with this device.

### 16.7.1 ARM Cortex-M0+ PPB memory map

The following table shows the memory map for the Cortex M0+ processor in this device.

**Table 16-5. ARM Cortex-M0+ PPB memory map**

| Start | End | Resource |
|---|---|---|
| Internal Private Peripheral Bus | | |
| 0xE000_0000 | 0xE000_DFFF | Reserved |
| 0xE000_E000 | 0xE000_E00F | Reserved |
| 0xE000_E010 | 0xE000_E0FF | SysTick |
| 0xE000_E100 | 0xE000_ECFF | Nested Vectored Interrupt Controller (NVIC) |

*Table continues on the next page...*

**Table 16-5. ARM Cortex-M0+ PPB memory map (continued)**

| Start | End | Resource |
|---|---|---|
| 0xE000_ED00 | 0xE000_ED8F | System Control Block |
| 0xE000_ED90 | 0xE000_EDEF | ARM Memory Protection Unit (MPU-ARM) |
| 0xE000_EDF0 | 0xE000_EEFF | Debug |
| 0xE000_EF00 | 0xE000_EFFF | Reserved |
| 0xE000_F000 | 0xE00F_EFFF | Reserved |
| 0xE00F_F000 | 0xE00F_FFFF | Core ROM Space (CRS) |
| 0xE010_0000 | 0xEFFF_FFFF | Reserved |
| External Private Peripheral Bus | | |
| 0xF000_0000 | 0xF000_0FFF | Micro Trace Buffer (MTB) |
| 0xF000_1000 | 0xF000_1FFF | Debug Watchpoint and Trace (DWT) |
| 0xF000_2000 | 0xF000_2FFF | Customer ROM Table |
| 0xF000_3000 | 0xF000_3FFF | Miscellaneous Control Module (MCM) |
| 0xF000_4000 | 0xF000_4FFF | Memory Mapped Division and Square Root (MMDVSQ) |
| 0xF000_5000 | 0xF000_5FFF | Reserved |
| 0xF000_6000 | 0xF000_6FFF | Cross Trigger Interface (CTI) |
| 0xF000_7000 | 0xF000_7FFF | Reserved |
| 0xF000_9000 | 0xF7FF_FFFF | Reserved |
| 0xF800_0000 | 0xF800_0FFF | IOPort |
| 0xF800_1000 | 0xFFFF_FFFF | Reserved |

## 16.7.2 ARM Cortex-M4 PPB memory map

The following table shows the memory map for the Cortex M4 processor in this device.

**Table 16-6. ARM Cortex-M4 PPB memory map**

| Start | End | Resource |
|---|---|---|
| Internal Private Peripheral Bus | | |
| 0xE000_0000 | 0xE000_0FFF | Instrumentation Trace Macrocell (ITM) |
| 0xE000_1000 | 0xE000_1FFF | Data Watchpoint and Trace (DWT) |
| 0xE000_2000 | 0xE000_2FFF | Flash Patch and Breakpoint (FPB) |
| 0xE000_3000 | 0xE000_DFFF | Reserved |
| 0xE000_E000 | 0xE000_EFFF | System Control Space (SCS) (for NVIC, MPU and FPU) |
| 0xE000_F000 | 0xE003_FFFF | Reserved |
| External Private Peripheral Bus | | |
| 0xE004_0000 | 0xE004_0FFF | Trace Port Interface Unit (TPIU) |
| 0xE004_1000 | 0xE004_1FFF | Embedded Trace Macrocell (ETM) |
| 0xE004_2000 | 0xE004_2FFF | Reserved |

*Table continues on the next page...*

## Table 16-6. ARM Cortex-M4 PPB memory map (continued)

| Start | End | Resource |
|---|---|---|
| 0xE004_3000 | 0xE004_3FFF | Reserved |
| 0xE004_4000 | 0xE004_4FFF | Cross Trigger Interface (CTI) |
| 0xE004_5000 | 0xE007_FFFF | Reserved |
| 0xE008_0000 | 0xE008_0FFF | Miscellaneous Control Module (MCM) |
| 0xE008_1000 | 0xE008_1FFF | Reserved |
| 0xE008_2000 | 0xE008_2FFF | Reserved |
| 0xE008_3000 | 0xE00F_EFFF | Reserved |
| 0xE00F_F000 | 0xE00F_FFFF | ROM Table |
| 0xE010_0000 | 0xFFFF_FFFF | Reserved |

# Chapter 17
# Flash Memory Module (FTFE)

## 17.1 Chip-specific FTFE information

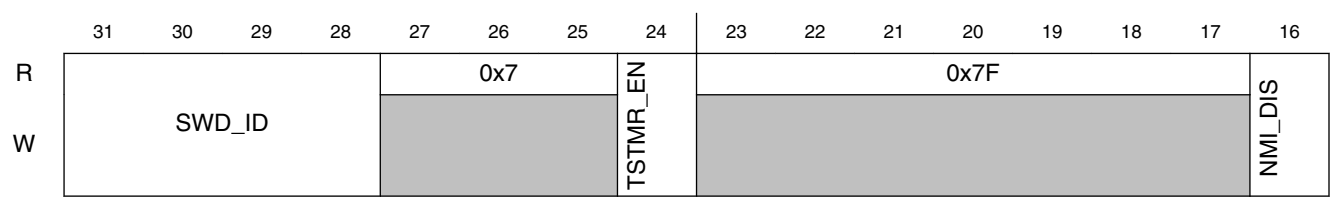
### 17.1.1 Flash Option (FOPT) customization

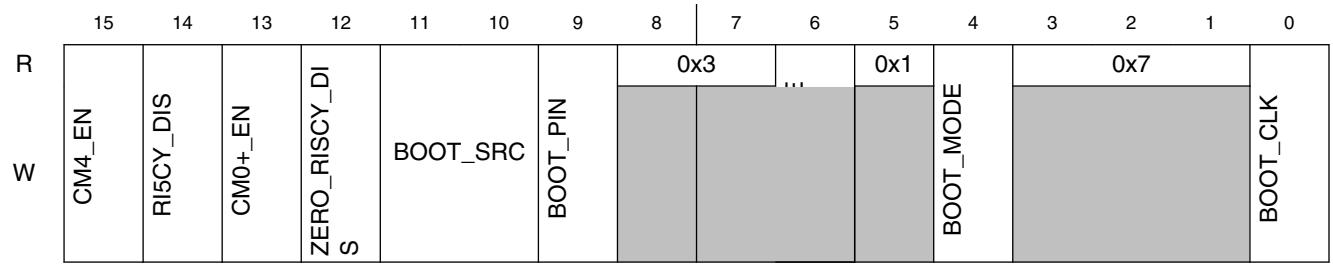
The Flash Option (FOPT) register in the Flash Memory module (FTFE_FOPT) allows the user to customize the operation of the MCU at boot time. The register contains read-only bits that are loaded from the NVM's flash configuration field. The default setting for all values in the FTFE_FOPT register is logic 1 since it is copied from the option byte residing in flash, which has all bits as logic 1 in the flash erased state. To configure for alternate settings, program the appropriate bits in the NVM option byte. The new settings will take effect on subsequent POR and any system reset. For more details on programming the option byte, see the flash memory chapter.

The MCU uses the FTFE_FOPT to configure the device at reset as shown in the following table.

**NOTE**

An FTFE_FOPT value of 0x0000_0000 is invalid and will be ignored. The FOPT register is written to 0xFFFF_FFFF if the contents of the Flash non-volatile option are 0x0000_0000.






## Table 17-3. Flash Option Register Bit Definitions

| Bit Num | Field | Definition |
|---|---|---|
| 31-28 | SWD_ID | **Serial Wire Debug Identification** - The SWD_ID sets the Instance ID of SWD modules that support multidrop. The Instance ID is used to select a specific SWD instance in systems with a single SWD interface port, but multiple devices with SWD interfaces. |
| 27-25 | Reserved | This field is reserved. |
| 24 | TSTMR_EN | **Time Stamp Timer Enable** - The TSTMR_EN bit enables the Time Stamp Timer (TSTMR) when the the Slow-Speed IRC is enabled.<br>• 0 - The TSTMR is disabled and cannot be enabled through any hardware or software operations.<br>• 1 - The TSTMR is enabled and begins counting whenever the Slow-Speed IRC is enabled. |
| 23-17 | Reserved | This field is reserved. |
| 16 | NMI_DIS | **Non-Maskable Interrupt (NMI) Disable** - The NMI_DIS bit disables the NMI signal.<br>• 0 - NMI pin and interrupts are enabled.<br>• 1 - NMI interrupts are always blocked. The associated pin continues to default to NMI pin controls with internal pullup enabled. |
| 15 | CM4_EN | **ARM CM4F core Enable** - The CM4_EN enables CM4F core.<br>• 0 - CM4F core is disabled.<br>• 1 - CM4F core is enabled. |
| 14 | RI5CY_DIS | **RI5CY core Disable** - The RI5CY_DIS bit disables RI5CY core.<br>• 0 - RI5CY core is enabled.<br>• 1 - RI5CY core is disabled. |
| 13 | CM0+_EN | **ARM CM0+ core Enable** - The CM0+_EN enables CM4F core.<br>• 0 - CM0+ core is disabled.<br>• 1 - CM0+ core is enabled. |
| 12 | ZERO_RISCY_DIS | **ZERO_RISCY core Disable** - The ZERO_RISCY_DIS bit disables ZERO_RISCY core.<br>• 0 - ZERO_RISCY core is enabled.<br>• 1 - ZERO_RISCY core is disabled. |
| 11-10 | BOOT_SRC | **Boot Source** - The BOOT_SRC field selects the boot source if the Boot Pin Configuration bit (BOOT_PIN) is set to 1.<br>• 00 - Boot from the internal flash memory.<br>• 01 - Reserved.<br>• 10 - Boot from the Boot ROM, configure communications peripheral(s), and enter boot loader mode. See Boot sequence for more information<br>• 11 - Boot from the Boot ROM, and enter boot loader mode. See Boot sequence for more information. |

*Table continues on the next page...*

**Table 17-3. Flash Option Register Bit Definitions (continued)**

| Bit Num | Field | Definition |
|---|---|---|
| 9 | BOOT_PIN | **Boot Pin Configuration** - The BOOT_PIN bit controls the behavior of the boot pin selection.<br>• 0 - Force Boot from ROM with update if BOOTCFG0 asserted, where BOOTCFG0 is the boot configuration function which is muxed with NMI pin. The RESET_b pin must be enabled when this option is selected.<br>• 1 - Boot source configured by Boot Source (BOOT_SRC) field. |
| 8-7 | Reserved | This field is reserved. |
| 6 | BOOT_CORE | **Boot Core** - The BOOT_CORE bit selects which core will start executing code at the completion of initialization. Additionally, the Non-Maskable Interrupt (NMI) is routed to this core if the NMI_DIS bit is set to 0.<br>• 0 - CPU1 (Cortex-M0+) or/and CPU3 (ZERO-RISCY), which depends on [CM0+_EN] and [ZERO_RISCY_DIS] settings, will start executing code on completion of initialization.<br>• 1 - CPU0 (Cortex-M4) or/and CPU2 (RI5CY), which depends on [CM4_EN] and [RI5CY_DIS] settings,will start executing code on completion of initialization. |
| 5 | Reserved | This field is reserved for future expansion. |
| 4 | BOOT_MODE | **Boot Mode** - The BOOT_MODE bit selects the operational mode that will be entered at the completion of initialization.<br>• 0 - The device will exit initialization in VLPR mode.<br>• 1 - The device will exit initialization in RUN mode. |
| 3-1 | Reserved | This field is reserved. |
| 0 | BOOT_CLK | **Boot Clock** - The BOOT_CLK field controls the setting of the Core Clock Divide Ratio (DIVCORE) field. Depending on the value of the BOOT_MODE bit, this field will be set in either the SCG_RCCR register or the SCG_VCCR register.<br>BOOT_MODE = 1 (Exit initialization in RUN mode.)<br>• 1 - SCG_RCCR[DIVCORE] set to 0000 - Divide-by-1<br>• 0 - SCG_RCCR[DIVCORE] set to 0001 - Divide-by-2<br>BOOT_MODE = 0 (Exit initialization in VLPR mode.)<br>• 1 - SCG_VCCR[DIVCORE] set to 0011 - Divide-by-4<br>• 0 - SCG_VCCR[DIVCORE] set to 0111 - Divide-by-8 |

## 17.2 External signal description

The flash module contains no signals that connect off-chip.

## 17.3 Memory map and registers

This section describes the memory map and registers for the flash module. Data read from unimplemented memory space in the flash module is undefined. Writes to unimplemented or reserved memory space (registers) in the flash module are ignored.

## 17.3.1 IFR descriptions

Primary Program flash block 0 contains two IFR flash memory spaces with restricted access separate from the main flash memory. Each IFR is divided into index fields with unreserved fields accessible using the Read Index command and/or programmable using the Program Index command. Once programmed, the index fields in the erasable IFR can be reprogrammed but only after a successful Erase All Blocks or Erase All Blocks Unsecure command operation which erases the entire flash memory space. There is no mechanism to erase the erasable IFR without erasing the entire flash memory. While each index field actually contains 8 bytes, the maps below indicate how many bytes are used.

### 17.3.1.1 Non-erasable IFR Map

| Record Index | Field Name | Field Size (Bytes) | RDINDEX | PGMINDEX | CCOB Used | Field Description |
|---|---|---|---|---|---|---|
| 0x00 - 0x07 | PGMONCEID | 64 | Yes | Yes | 4 - B | Program Once Identifier |
| 0x08 - 0x0F | | 64 | Yes | Yes | 4 - B | Available |
| 0x10 | VERSIONID | 8 | Yes | No | 4 - B | Version Identifier |
| 0x11 - 0x17 | | 56 | Yes | No | 4 - B | Available |
| 0x18 - 0x1F | | 64 | Yes | No | 4 - B | Available |
| 0x20 - 0x7F | | 768 | | | | Reserved |

### 17.3.1.2 Erasable IFR Map

| Record Index | Field Name | Field Size (Bytes) | RDINDEX | PGMINDEX | CCOB Used | Field Description |
|---|---|---|---|---|---|---|
| 0x80 | SEC0 | 2 | Yes | Yes | 6 - 7 | Flash security - Primary. Refer to the description of the Flash Security Register (FSEC). |
| 0x81 | FSLACC | 2 | Yes | Yes | 6 - 7 | Flash security - Factory Security Level Access. Refer to the description of the Flash |

*Table continues on the next page...*

| Record Index | Field Name | Field Size (Bytes) | RDINDEX | PGMINDEX | CCOB Used | Field Description |
|---|---|---|---|---|---|---|
| | | | | | | Security Register (FSEC). |
| 0x82 | MEEN | 2 | Yes | Yes | 6 - 7 | Flash security - Mass Erase Enable. Refer to the description of the Flash Security Register (FSEC). |
| 0x83 | KEYEN | 2 | Yes | Yes | 6 - 7 | Flash security - Backdoor Key Enable. Refer to the description of the Flash Security Register (FSEC). |
| 0x84 | FOPT | 4 | Yes | Yes | 4 - 7 | Flash nonvolatile option. Refer to the description of the Flash Option Registers (FOPT). |
| 0x85 | FPROT | 8 | Yes | Yes | 4 - B | Primary Program flash protection. Refer to the description of the Primary Program Flash Protection Registers (FPROT). |
| 0x86 | FPROTS | 2 | Yes | Yes | A - B | Secondary Program flash protection. Refer to the description of the Secondary Program Flash Protection Registers (FPROTS). |
| 0x87 | | 8 | Yes | Yes | 4 - B | Available |
| 0x88 | XACC | 8 | Yes | Yes | 4 - B | FAC Execute Protection (Primary). Refer to the description of |

*Table continues on the next page...*

| Record Index | Field Name | Field Size (Bytes) | RDINDEX | PGMINDEX | CCOB Used | Field Description |
|---|---|---|---|---|---|---|
| | | | | | | the Primary Execute-only Access Registers (FXACC). |
| 0x89 | SACC | 8 | Yes | Yes | 4 - B | FAC Supervisor Protection (Primary). Refer to the description of the Primary Supervisor Access Registers (FSACC). |
| 0x8A | XACCS | 2 | Yes | Yes | A - B | FAC Execute Protection (Secondary). Refer to the description of the Secondary Execute-only Access Registers (FXACCS). |
| 0x8B | SACCS | 2 | Yes | Yes | A - B | FAC Supervisor Protection (Secondary). Refer to the description of the Secondary Supervisor Access Registers (FSACCS). |
| 0x8C - 0x8F | | 32 | | | | Reserved |
| 0x90 - 0x93 | ROMF0 | 32 | Yes | Yes | 4 - B | ROM Field 0 (256-bit). Refer to ROM chapter for details. |
| 0x94 - 0x97 | ROMF1 | 32 | Yes | Yes | 4 - B | ROM Field 1 (256-bit). Refer to ROM chapter for details. |
| 0x98 - 0x9B | ROMF2 | 32 | Yes | Yes | 4 - B | ROM Field 2 (256-bit). Refer to ROM chapter for details. |
| 0x9C - 0x9F | ROMF3 | 32 | Yes | Yes | 4 - B | ROM Field 3 (256-bit). Refer to ROM chapter for details. |

*Table continues on the next page...*

| Record Index | Field Name | Field Size (Bytes) | RDINDEX | PGMINDEX | CCOB Used | Field Description |
|---|---|---|---|---|---|---|
| 0xA0 | SEC1 | 2 | Yes | No | 6 - 7 | Security Qualifier. Refer to the description of the Flash Security Register (FSEC). |
| 0xA1 | SWAPADDR | 2 | Yes | No | 6 - 7 | Swap Indicator Address. Refer to the description of the Swap Control command. |
| 0xA2 | SWAPEN | 2 | Yes | No | 6 - 7 | Swap System Enable. Refer to the description of the Swap Control command. |
| 0xA3 | SWAPDIS | 2 | Yes | No | 6 - 7 | Swap System Disable. Refer to the description of the Swap Control command. |
| 0xA4 - 0xAF | | 96 | | | | Reserved |
| 0xB0 | KEY | 8 | No | Yes | 4 - B | Backdoor Comparison Key (64-bit). Refer to Verify Backdoor Access Key command and Unsecuring the MCU Using Backdoor Key Access. |
| 0xB1 - 0xB4 | KEK | 32 | No | Yes | 4 - B | Key Encription Key (256-bit) |
| 0xB5 - 0xBF | | 88 | No | Yes | 4 - B | Available for factory use |
| 0xC0 - 0xFF | | 512 | | | | Reserved |

#### 17.3.1.3 Flash Configuration Field description

The erasable IFR contains flash configuration field records that store default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the flash module. All fields are programmable using the Program Index command except for the secondary security field (SEC1) which is programmed during a successful Erase All Blocks or Erase All Blocks Unsecure operation. All fields are readable using the Read Index command except for the backdoor comparison key (KEY). The following index fields make up the flash configuration field.

- SEC0 (0x80)
- FSLACC (0x81)
- MEEN (0x82)
- KEYEN (0x83)
- FOPT (0x84)
- FPROTP (0x85)
- FPROTS (0x86)
- SEC1 (0xA0)
- KEY (0xB0)

### 17.3.2 Register descriptions

The flash module contains a set of memory-mapped control and status registers.

**NOTE**

While a command is running (FSTAT[CCIF]=0), register writes are not accepted to any register except FCNFG and FSTAT. The no-write rule is relaxed during the start-up reset sequence, prior to the initial rise of CCIF. During this initialization period the user may write any register. All register writes are also disabled (except for registers FCNFG and FSTAT) whenever an erase suspend request is active (FCNFG[ERSSUSP]=1).

#### 17.3.2.1 Flash register descriptions

##### 17.3.2.1.1 Flash Memory map

FTFE base address: 4002_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Flash Status Register (FSTAT) | 8 | W1C | 00h |
| 1h | Flash Configuration Register (FCNFG) | 8 | RW | 02h |
| 2h | Flash Security Register (FSEC) | 8 | RO | See description. |
| 4h | Flash Common Command Object Registers (FCCOB3) | 8 | RW | 00h |
| 5h | Flash Common Command Object Registers (FCCOB2) | 8 | RW | 00h |
| 6h | Flash Common Command Object Registers (FCCOB1) | 8 | RW | 00h |
| 7h | Flash Common Command Object Registers (FCCOB0) | 8 | RW | 00h |
| 8h | Flash Common Command Object Registers (FCCOB7) | 8 | RW | 00h |
| 9h | Flash Common Command Object Registers (FCCOB6) | 8 | RW | 00h |
| Ah | Flash Common Command Object Registers (FCCOB5) | 8 | RW | 00h |
| Bh | Flash Common Command Object Registers (FCCOB4) | 8 | RW | 00h |
| Ch | Flash Common Command Object Registers (FCCOBB) | 8 | RW | 00h |
| Dh | Flash Common Command Object Registers (FCCOBA) | 8 | RW | 00h |
| Eh | Flash Common Command Object Registers (FCCOB9) | 8 | RW | 00h |
| Fh | Flash Common Command Object Registers (FCCOB8) | 8 | RW | 00h |
| 10h | Flash Option Registers (FOPT3) | 8 | RO | See description. |
| 11h | Flash Option Registers (FOPT2) | 8 | RO | See description. |
| 12h | Flash Option Registers (FOPT1) | 8 | RO | See description. |
| 13h | Flash Option Registers (FOPT0) | 8 | RO | See description. |
| 18h | Primary Program Flash Protection Registers (FPROTH3) | 8 | RW | See description. |
| 19h | Primary Program Flash Protection Registers (FPROTH2) | 8 | RW | See description. |
| 1Ah | Primary Program Flash Protection Registers (FPROTH1) | 8 | RW | See description. |
| 1Bh | Primary Program Flash Protection Registers (FPROTH0) | 8 | RW | See description. |
| 1Ch | Primary Program Flash Protection Registers (FPROTL3) | 8 | RW | See description. |
| 1Dh | Primary Program Flash Protection Registers (FPROTL2) | 8 | RW | See description. |
| 1Eh | Primary Program Flash Protection Registers (FPROTL1) | 8 | RW | See description. |
| 1Fh | Primary Program Flash Protection Registers (FPROTL0) | 8 | RW | See description. |
| 24h | Secondary Program Flash Protection Registers (FPROTSL) | 8 | RW | See description. |
| 25h | Secondary Program Flash Protection Registers (FPROTSH) | 8 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2Ch | Primary Flash Access Segment Size Register (FACSS) | 8 | RO | See description. |
| 2Dh | Primary Flash Access Segment Number Register (FACSN) | 8 | RO | See description. |
| 2Eh | Secondary Flash Access Segment Size Register (FACSSS) | 8 | RO | See description. |
| 2Fh | Secondary Flash Access Segment Number Register (FACSNS) | 8 | RO | See description. |
| 30h | Primary Execute-only Access Registers (XACCH3) | 8 | RO | See description. |
| 31h | Primary Execute-only Access Registers (XACCH2) | 8 | RO | See description. |
| 32h | Primary Execute-only Access Registers (XACCH1) | 8 | RO | See description. |
| 33h | Primary Execute-only Access Registers (XACCH0) | 8 | RO | See description. |
| 34h | Primary Execute-only Access Registers (XACCL3) | 8 | RO | See description. |
| 35h | Primary Execute-only Access Registers (XACCL2) | 8 | RO | See description. |
| 36h | Primary Execute-only Access Registers (XACCL1) | 8 | RO | See description. |
| 37h | Primary Execute-only Access Registers (XACCL0) | 8 | RO | See description. |
| 38h | Primary Supervisor-only Access Registers (SACCH3) | 8 | RO | See description. |
| 39h | Primary Supervisor-only Access Registers (SACCH2) | 8 | RO | See description. |
| 3Ah | Primary Supervisor-only Access Registers (SACCH1) | 8 | RO | See description. |
| 3Bh | Primary Supervisor-only Access Registers (SACCH0) | 8 | RO | See description. |
| 3Ch | Primary Supervisor-only Access Registers (SACCL3) | 8 | RO | See description. |
| 3Dh | Primary Supervisor-only Access Registers (SACCL2) | 8 | RO | See description. |
| 3Eh | Primary Supervisor-only Access Registers (SACCL1) | 8 | RO | See description. |
| 3Fh | Primary Supervisor-only Access Registers (SACCL0) | 8 | RO | See description. |
| 44h | Secondary Execute-only Access Registers (XACCSL) | 8 | RO | See description. |
| 45h | Secondary Execute-only Access Registers (XACCSH) | 8 | RO | See description. |
| 4Ch | Secondary Supervisor-only Access Registers (SACCSL) | 8 | RO | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4Dh | Secondary Supervisor-only Access Registers (SACCSH) | 8 | RO | See description. |
| 52h | Flash Standby Control Register (FSTDBYCTL) | 8 | RO | 00h |
| 53h | Flash Standby Register (FSTDBY) | 8 | RW | 00h |

## 17.3.2.1.2 Flash Status Register (FSTAT)

### 17.3.2.1.2.1 Offset

| Register | Offset |
|---|---|
| FSTAT | 0h |

### 17.3.2.1.2.2 Function

The FSTAT register reports the operational status of the flash module.

The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable. The MGSTAT0 bit is read only. The unassigned bits read 0 and are not writable.

**NOTE**

When set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL) bits in this register prevent the launch of any more commands until the flag is cleared (by writing a one to it).

### 17.3.2.1.2.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | RDCOLERR | ACCERR | FPVIOL | 0 | | | MGSTAT0 |
| W | W1C | W1C | W1C | W1C | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 17.3.2.1.2.4 Fields

| Field | Function |
|---|---|
| 7<br>CCIF | Command Complete Interrupt Flag<br>The CCIF flag indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a flash command, and CCIF stays low until the command completes or a command violation is detected.<br>The CCIF bit is reset to 0 but is set to 1 by the flash module at the end of the reset initialization sequence. Depending on how quickly the read occurs after reset release, the user may or may not see the 0 hardware reset value.<br>0b - Flash command in progress<br>1b - Flash command has completed |
| 6<br>RDCOLERR | Flash Read Collision Error Flag<br>The RDCOLERR error bit indicates that the MCU attempted a read from a flash memory resource that was being manipulated by a flash command (CCIF=0). Any simultaneous access is detected as a collision error by the block arbitration logic. The read data in this case cannot be guaranteed. The RDCOLERR bit is cleared by writing a 1 to it. Writing a 0 to RDCOLERR has no effect.<br>0b - No collision error detected<br>1b - Collision error detected |
| 5<br>ACCERR | Flash Access Error Flag<br>The ACCERR error bit indicates an illegal access has occurred to a flash memory resource caused by a violation of the command write sequence or issuing an illegal flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR while CCIF is set. Writing a 0 to the ACCERR bit has no effect.<br>0b - No access error detected<br>1b - Access error detected |
| 4<br>FPVIOL | Flash Protection Violation Flag<br>The FPVIOL error bit indicates an attempt was made to program or erase an address in a protected area of program flash memory during a command write sequence. While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to FPVIOL while CCIF is set. Writing a 0 to the FPVIOL bit has no effect.<br>0b - No protection violation detected<br>1b - Protection violation detected |
| 3-1<br>— | Reserved |
| 0<br>MGSTAT0 | Memory Controller Command Completion Status Flag<br>The MGSTAT0 status flag is set if an error is detected during execution of a flash command or during the flash reset sequence. As a status flag, this bit cannot (and need not) be cleared by the user like the other error flags in this register.<br>The value of the MGSTAT0 bit for "command-N" is valid only at the end of the "command-N" execution when CCIF=1 and before the next command has been launched. At some point during the execution of "command-N+1," the previous result is discarded and any previous error is cleared. |

### 17.3.2.1.3 Flash Configuration Register (FCNFG)

#### 17.3.2.1.3.1 Offset

| Register | Offset |
|---|---|
| FCNFG | 1h |

#### 17.3.2.1.3.2 Function

This register provides information on the current functional state of the flash module.

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. SWAP, PFLSH, and RAMRDY are read-only status bits. The reset values for the SWAP, PFLSH, and RAMRDY bits are determined during the reset sequence.

#### 17.3.2.1.3.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | RDCOLLIE | ERSAREQ | ERSSUSP | SWAP | CRCRDY | RAMRDY | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 17.3.2.1.3.4 Fields

| Field | Function |
|---|---|
| 7<br>CCIE | Command Complete Interrupt Enable<br>The CCIE bit controls interrupt generation when a flash command completes.<br>0b - Command complete interrupt disabled<br>1b - Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set. |
| 6<br>RDCOLLIE | Read Collision Error Interrupt Enable<br>The RDCOLLIE bit controls interrupt generation when a flash read collision error occurs.<br>0b - Read collision error interrupt disabled<br>1b - Read collision error interrupt enabled. An interrupt request is generated whenever a flash read collision error is detected (see the description of FSTAT[RDCOLERR]). |
| 5<br>ERSAREQ | Erase All Request<br>This bit issues a request to the memory controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the device's Chip Configuration details on how to request this command.<br>The ERSAREQ bit sets when an erase all request is triggered external to the flash module and CCIF is set (no command is currently being executed). ERSAREQ is cleared by the flash module when the operation completes. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - No request or request complete<br>1b - Request to: (1) run the Erase All Blocks command, (2) verify the erased state, (3) program the security byte in the Flash Configuration Field to the unsecure state, and (4) release MCU security by setting the FSEC[SEC] field to the unsecure state. |
| 4<br>ERSSUSP | Erase Suspend<br>The ERSSUSP bit allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing.<br>0b - No suspend requested<br>1b - Suspend the current Erase Flash Sector command execution |
| 3<br>SWAP | Swap<br>The SWAP flag indicates which half of the primary program flash space is located at relative address 0x0000. The state of the SWAP flag is set by the flash module during the reset sequence. See Swap Control command for information on swap management.<br>0b - Program flash 0 block is located at relative address 0x0000<br>1b - Program flash 1 block is located at relative address 0x0000 |
| 2<br>CRCRDY | CRC Ready<br>The CRC Ready flag indicates the programming acceleration RAM is available for CRC operations. See Generate CRC command for more information.<br>0b - Programming acceleration RAM is not available for CRC operations<br>1b - Programming acceleration RAM is available for CRC operations |
| 1<br>RAMRDY | RAM Ready<br>This flag indicates the current status of the programming acceleration RAM.<br>This bit should always be set.<br>0b - Programming acceleration RAM is not available<br>1b - Programming acceleration RAM is available |
| 0<br>— | Reserved |

## 17.3.2.1.4 Flash Security Register (FSEC)

### 17.3.2.1.4.1 Offset

| Register | Offset |
|---|---|
| FSEC | 2h |

### 17.3.2.1.4.2 Function

This read-only register holds all bits associated with the security of the MCU and flash module.

During the reset sequence, the register is loaded with the contents of the flash security byte in the Flash Configuration Field located in program flash memory.

### 17.3.2.1.4.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN | | MEEN | | FSLACC | | SEC | |
| W | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u |

### 17.3.2.1.4.4 Fields

| Field | Function |
|---|---|
| 7-6<br>KEYEN | Backdoor Key Security Enable<br>These bits enable and disable backdoor key access to the flash module.<br>00b - Backdoor key access disabled<br>01b - Backdoor key access disabled (preferred KEYEN state to disable backdoor key access)<br>10b - Backdoor key access enabled<br>11b - Backdoor key access disabled |
| 5-4<br>MEEN | Mass Erase Enable Bits<br>Enables and disables mass erase capability of the flash module. The state of the MEEN bits is only relevant when the SEC bits are set to secure. When the SEC field is set to unsecure, the MEEN setting does not matter.<br>00b - Mass erase is enabled<br>01b - Mass erase is enabled<br>10b - Mass erase is disabled<br>11b - Mass erase is enabled |
| 3-2<br>FSLACC | Factory Security Level Access Code<br>These bits enable or disable access to the flash memory contents during returned part failure analysis at NXP. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by NXP factory test must begin with a full erase to unsecure the part.<br>When access is granted (SEC is unsecure, or SEC is secure and FSLACC is granted), NXP factory testing has visibility of the current flash contents. The state of the FSLACC bits is only relevant when the SEC bits are set to secure. When the SEC field is set to unsecure, the FSLACC setting does not matter.<br>00b - Factory access granted<br>01b - Factory access denied<br>10b - Factory access denied<br>11b - Factory access granted |
| 1-0<br>SEC | Flash Security<br>These bits define the security state of the MCU. In the secure state, the MCU limits access to flash module resources. The limitations are defined per device and are detailed in the Chip Configuration details. If the flash module is unsecured using backdoor key access, the SEC bits are forced to 10b.<br>00b - MCU security status is secure<br>01b - MCU security status is secure<br>10b - MCU security status is unsecure (The standard shipping condition of the flash module is unsecure.)<br>11b - MCU security status is secure |

#### 17.3.2.1.5 Flash Common Command Object Registers (FCCOB3 - FCCO B8)

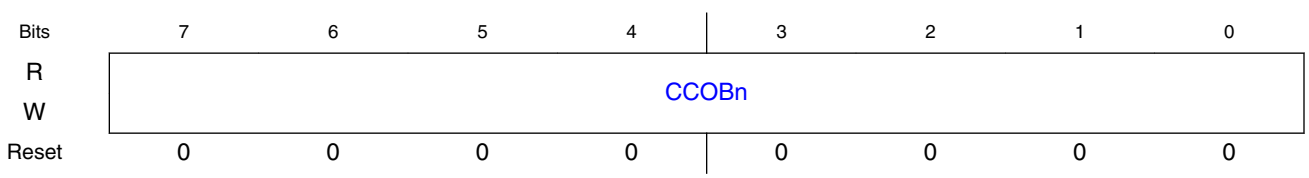
##### 17.3.2.1.5.1 Offset

| Register | Offset |
|---|---|
| FCCOB3 | 4h |
| FCCOB2 | 5h |
| FCCOB1 | 6h |
| FCCOB0 | 7h |
| FCCOB7 | 8h |
| FCCOB6 | 9h |
| FCCOB5 | Ah |
| FCCOB4 | Bh |
| FCCOBB | Ch |
| FCCOBA | Dh |
| FCCOB9 | Eh |
| FCCOB8 | Fh |

##### 17.3.2.1.5.2 Function

The FCCOB register group provides 12 bytes for command codes and parameters. The individual bytes within the set append a 0-B hex identifier to the FCCOB register name: FCCOB0, FCCOB1, ..., FCCOBB.

##### 17.3.2.1.5.3 Diagram




##### 17.3.2.1.5.4 Fields

| Field | Function |
|---|---|
| 7-0<br>CCOBn | CCOBn |

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
The FCCOB register provides a command code and relevant parameters to the memory controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command's execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes.<br><br>
Some commands return information to the FCCOB registers. Any values returned to FCCOB are available for reading after the FSTAT[CCIF] flag returns to 1 by the memory controller.<br><br>
The following table shows a generic flash command format. The first FCCOB register, FCCOB0, always contains the command code. This 8-bit value defines the command to be executed. The command code is followed by the parameters required for this specific flash command, typically an address and/or data values.<br><br>
NOTE: The command parameter table is written in terms of FCCOB Number (which is equivalent to the byte number). This number is a reference to the FCCOB register name and is not the register address.<br><br>
<table>
<tr><th>FCCOB Number</th><th>Typical Command Parameter Contents [7:0]</th></tr>
<tr><td>0</td><td>FCMD (a code that defines the flash command)</td></tr>
<tr><td>1</td><td>Flash address [23:16]</td></tr>
<tr><td>2</td><td>Flash address [15:8]</td></tr>
<tr><td>3</td><td>Flash address [7:0]</td></tr>
<tr><td>4</td><td>Data Byte 0</td></tr>
<tr><td>5</td><td>Data Byte 1</td></tr>
<tr><td>6</td><td>Data Byte 2</td></tr>
<tr><td>7</td><td>Data Byte 3</td></tr>
<tr><td>8</td><td>Data Byte 4</td></tr>
<tr><td>9</td><td>Data Byte 5</td></tr>
<tr><td>A</td><td>Data Byte 6</td></tr>
<tr><td>B</td><td>Data Byte 7</td></tr>
</table>
<br>
1. Refers to FCCOB register name, not register address<br><br>
FCCOB Endianness and Multi-Byte Access:<br><br>
The FCCOB register group uses a big endian addressing convention. For all command parameter fields larger than 1 byte, the most significant data resides in the lowest FCCOB register number. The FCCOB register group may be read and written as individual bytes, aligned words (2 bytes) or aligned longwords (4 bytes).
</td></tr>
</table>

1. Refers to FCCOB register name, not register address

## 17.3.2.1.6 Flash Option Registers (FOPT3 - FOPT0)

### 17.3.2.1.6.1 Offset

For a = 0 to 3:

| Register | Offset |
|---|---|
| FOPTa | Dh + (a × 1h) |

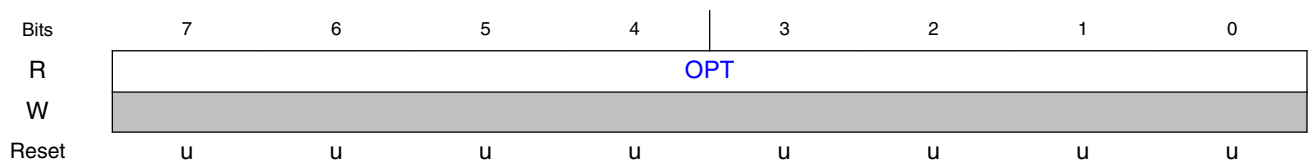
#### 17.3.2.1.6.2 Function

The flash option registers allow the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in these registers are read-only.

During the reset sequence, these registers are loaded from the flash nonvolatile option index in the Flash Configuration Field located in the program flash erasable IFR.

#### 17.3.2.1.6.3 Diagram




#### 17.3.2.1.6.4 Fields

| Field | Function |
|---|---|
| 7-0<br>OPT | Nonvolatile Option<br>These bits are loaded from flash to this register at reset. Refer to the device's Chip Configuration details for the definition and use of these bits. |

### 17.3.2.1.7 Primary Program Flash Protection Registers (FPROTH3 - FPROTL0)

#### 17.3.2.1.7.1 Offset

| Register | Offset |
|---|---|
| FPROTH3 | 18h |
| FPROTH2 | 19h |
| FPROTH1 | 1Ah |
| FPROTH0 | 1Bh |

*Table continues on the next page...*

| Register | Offset |
|---|---|
| FPROTL3 | 1Ch |
| FPROTL2 | 1Dh |
| FPROTL1 | 1Eh |
| FPROTL0 | 1Fh |

#### 17.3.2.1.7.2 Function

The FPROT registers define which primary program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by program and erase operations.

The eight FPROT registers allow up to 64 protectable regions of equal memory size. The FPROT register bits are mapped to the protectable regions as follows:

- *region_size* = total_program_flash_size / 64
- Starting from the primary program flash base address, Region 0 refers to the first *region_size* bytes, region 1 refers to the next *region_size* bytes, and so on.

In other words:

- region_*n*_start_address = primary_ program_flash_base_address + (n) * *region_size*
- region_*n*_end_address = primary_ program_flash_base_address + ((n+1) * *region_size*) -1

For example, if the total primary program flash size is 1 MByte, the *region_size* is 16 Kbytes. If the lowest primary program flash address is 0x0000_0000, the regions are defined as follows:

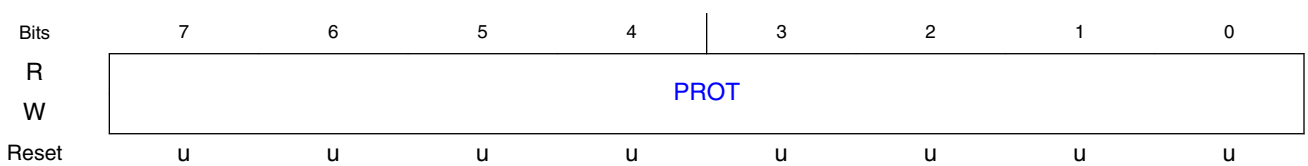
| Region | Start Address | End Address |
|---|---|---|
| 0 | 0x0000_0000 | 0x0000_3FFF |
| 1 | 0x0000_4000 | 0x0000_7FFF |
| 2 | 0x0000_8000 | 0x0000_BFFF |
| 3 | 0x0000_C000 | 0x0000_FFFF |
| . . . | . . . | . . . |
| 63 | 0x000F_C000 | 0x000F_FFFF |

- A '0' in an FPROT[PROT] bit indicates its corresponding flash protection region is protected
- A '1' in an FPROT[PROT] bit indicates its corresponding flash protection region is unprotected

| Primary Program flash protection register | Primary Program flash protection bits | CCOB used (RDINDEX/ PGMINDEX command) |
|---|---|---|
| FPROTH0 | PROT[63:56] | 4 |
| FPROTH1 | PROT[55:48] | 5 |
| FPROTH2 | PROT[47:40] | 6 |
| FPROTH3 | PROT[39:32] | 7 |
| FPROTL0 | PROT[31:24] | 8 |
| FPROTL1 | PROT[23:16] | 9 |
| FPROTL2 | PROT[15:8] | A |
| FPROTL3 | PROT[7:0] | B |

During the reset sequence, the FPROT registers are loaded with the contents of the primary program flash protection index in the Flash Configuration Field located in the program flash erasable IFR. To change the primary program flash protection that is loaded during the reset sequence, the program flash erasable IFR must be erased using the Erase All Blocks or Erase All Blocks Unsecure command. Then, the primary program flash protection index must be programmed using the Program Index command.

### 17.3.2.1.7.3 Diagram




### 17.3.2.1.7.4 Fields

| Field | Function |
|---|---|
| 7-0<br>PROT | Primary Program Flash Region Protect<br>Each primary program flash region can be protected from program and erase operations by setting the associated PROT bit to the protected state.<br>**In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected, but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FPROT bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored. |

| Field | Function |
|---|---|
| | **Restriction:** The user must never write to any FPROT register while a command is running (CCIF=0) on a primary program flash block.<br>Trying to alter data in any protected area in the primary program flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of a primary program flash block is not possible if it contains any protected region. |

#### 17.3.2.1.8 Secondary Program Flash Protection Registers (FPROTSL - FPROTSH)

##### 17.3.2.1.8.1 Offset

| Register | Offset |
|---|---|
| FPROTSL | 24h |
| FPROTSH | 25h |

##### 17.3.2.1.8.2 Function

The FPROTS registers define which secondary program flash regions are protected from program and erase operations. Protected flash regions cannot have their content changed; that is, these regions cannot be programmed and cannot be erased by any flash command. Unprotected regions can be changed by program and erase operations.

The two FPROTS registers allow up to 16 protectable regions of equal memory size. The FPROTS register bits are mapped to the protectable regions as follows:

- *region_size* = total_program_flash_size / 16
- Starting from the secondary program flash base address, Region 0 refers to the first *region_size* bytes, region 1 refers to the next *region_size* bytes, and so on.

In other words:

- region_*n*_start_address = secondary_program_flash_base_address + (n) * *region_size*
- region_*n*_end_address = secondary_program_flash_base_address + ((n+1) * *region_size*) -1

For example, if the total secondary program flash size is 256 KByte, the *region_size* is 16 Kbytes. If the lowest secondary program flash address is 0x0000_0000, the regions are defined as follows:

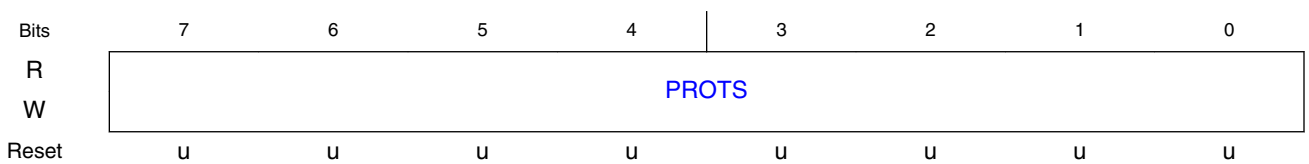
| Region | Start Address | End Address |
|---|---|---|
| 0 | 0x0000_0000 | 0x0000_3FFF |
| 1 | 0x0000_4000 | 0x0000_7FFF |
| 2 | 0x0000_8000 | 0x0000_BFFF |
| 3 | 0x0000_C000 | 0x0000_FFFF |
| . . . | . . . | . . . |
| 15 | 0x0003_C000 | 0x0003_FFFF |

- A '0' in an FPROTS[PROTS] bit indicates its corresponding flash protection region is protected
- A '1' in an FPROTS[PROTS] bit indicates its corresponding flash protection region is unprotected

| Secondary program flash protection register | Secondary program flash protection bits | CCOB used (RDINDEX/ PGMINDEX command) |
|---|---|---|
| FPROTSH | PROTS[15:8] | A |
| FPROTSL | PROTS[7:0] | B |

During the reset sequence, the FPROTS registers are loaded with the contents of the secondary program flash protection index in the Flash Configuration Field located in the program flash erasable IFR. To change the secondary program flash protection that is loaded during the reset sequence, the program flash erasable IFR must be erased using the Erase All Blocks or Erase All Blocks Unsecure command. Then, the secondary program flash protection index must be programmed using the Program Index command.

### 17.3.2.1.8.3 Diagram




### 17.3.2.1.8.4 Fields

| Field | Function |
|---|---|
| 7-0<br>PROTS | Secondary Program Flash Region Protect<br>Each secondary program flash region can be protected from program and erase operations by setting the associated PROTS bit to the protected state. |

| Field | Function |
|---|---|
| | **In NVM Normal mode:** The protection can only be increased, meaning that currently unprotected memory can be protected, but currently protected memory cannot be unprotected. Since unprotected regions are marked with a 1 and protected regions use a 0, only writes changing 1s to 0s are accepted. This 1-to-0 transition check is performed on a bit-by-bit basis. Those FPROTS bits with 1-to-0 transitions are accepted while all bits with 0-to-1 transitions are ignored.<br>**Restriction:** The user must never write to any FPROTS register while a command is running (CCIF=0) on the secondary program flash.<br>Trying to alter data in any protected area in the secondary program flash memory results in a protection violation error and sets the FSTAT[FPVIOL] bit. A full block erase of a secondary program flash block is not possible if it contains any protected region. |

## 17.3.2.1.9 Primary Flash Access Segment Size Register (FACSS)

### 17.3.2.1.9.1 Offset

| Register | Offset |
|---|---|
| FACSS | 2Ch |

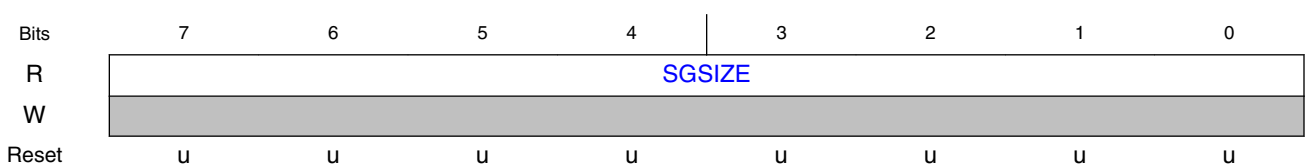
### 17.3.2.1.9.2 Function

The primary flash access segment size register determines which bits in the address are used to index into the SACC and XACC bitmaps to get the appropriate permission flags.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

### 17.3.2.1.9.3 Diagram




### 17.3.2.1.9.4 Fields

| Field | Function |
|---|---|
| 7-0<br>SGSIZE | Segment Size<br>The segment size is a fixed value based on the available primary program flash size divided by NUMSG. |

| Field | Function | | |
|---|---|---|---|
| | Primary Program Flash Size | Segment Size | Segment Size Encoding |
| | 512 KBytes | 8 KBytes | 0x5 |
| | 768 KBytes | 16 KBytes | 0x6 |
| | 1 MByte | 16 KBytes | 0x6 |
| | 1.5 MBytes | 32 KBytes | 0x7 |
| | 2 MBytes | 32 KBytes | 0x7 |

## 17.3.2.1.10 Primary Flash Access Segment Number Register (FACSN)

### 17.3.2.1.10.1 Offset

| Register | Offset |
|---|---|
| FACSN | 2Dh |

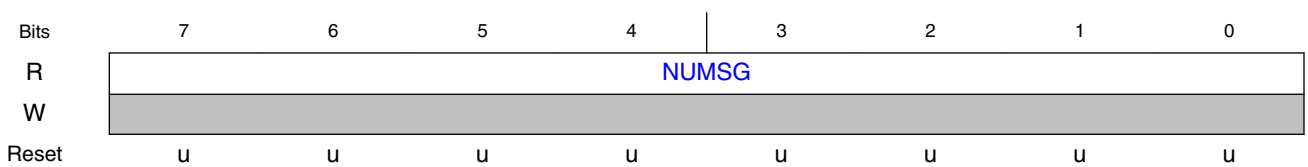
### 17.3.2.1.10.2 Function

The flash access segment number register provides the number of primary program flash segments that are available for XACC and SACC permissions.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

### 17.3.2.1.10.3 Diagram




### 17.3.2.1.10.4 Fields

| Field | Function |
|---|---|
| 7-0 | Number of Segments Indicator |
| NUMSG | The NUMSG field indicates the number of equal-sized segments in the primary program flash. |

| Field | Function |
|---|---|
| | 00110000b - Primary Program flash memory is divided into 48 segments (768 Kbytes, 1.5 Mbytes)<br>01000000b - Primary Program flash memory is divided into 64 segments (512 Kbytes, 1 Mbyte, 2 Mbytes) |

## 17.3.2.1.11 Secondary Flash Access Segment Size Register (FACSSS)

### 17.3.2.1.11.1 Offset

| Register | Offset |
|---|---|
| FACSSS | 2Eh |

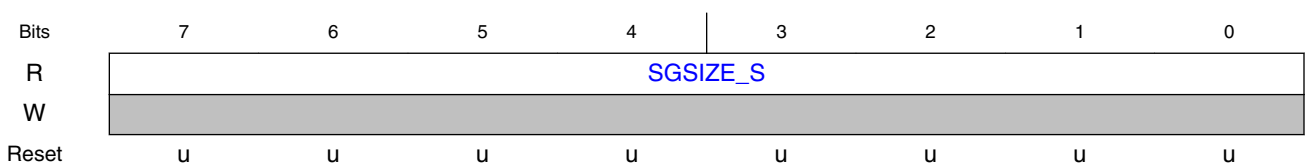
### 17.3.2.1.11.2 Function

The secondary flash access segment size register determines which bits in the address are used to index into the SACCS and XACCS bitmaps to get the appropriate permission flags.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

### 17.3.2.1.11.3 Diagram




### 17.3.2.1.11.4 Fields

| Field | Function |
|---|---|
| 7-0<br>SGSIZE_S | Segment Size<br>The segment size is a fixed value based on the available primary program flash size divided by NUMSG. |

| Secondary Program Flash Size | Segment Size | Segment Size Encoding |
|---|---|---|
| 32 KBytes | 2 KBytes | 0x3 |
| 64 KBytes | 4 KBytes | 0x4 |

| Field | Function | | |
|---|---|---|---|
| | 128 KBytes | 8 KBytes | 0x5 |
| | 256 KBytes | 16 KBytes | 0x6 |
| | 512 KBytes | 32 KBytes | 0x7 |

## 17.3.2.1.12 Secondary Flash Access Segment Number Register (FACSNS)

### 17.3.2.1.12.1 Offset

| Register | Offset |
|---|---|
| FACSNS | 2Fh |

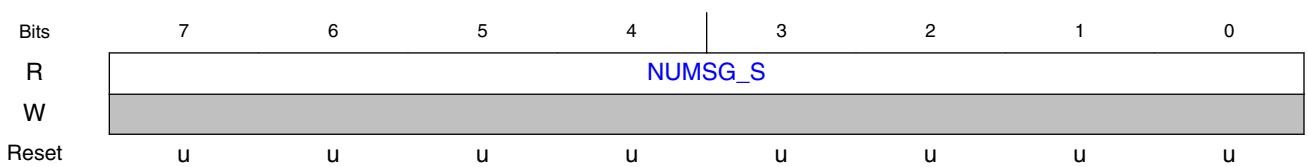
### 17.3.2.1.12.2 Function

The flash access segment number register provides the number of secondary program flash segments that are available for XACCS and SACCS permissions.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

### 17.3.2.1.12.3 Diagram




### 17.3.2.1.12.4 Fields

| Field | Function |
|---|---|
| 7-0<br>NUMSG_S | Number of Segments Indicator<br>The NUMSG_S field indicates the number of equal-sized segments in the secondary program flash.<br>00010000b - Secondary Program flash memory is divided into 16 segments |

## 17.3.2.1.13 Primary Execute-only Access Registers (XACCH3 - XACCL0)

### 17.3.2.1.13.1 Offset

| Register | Offset |
|---|---|
| XACCH3 | 30h |
| XACCH2 | 31h |
| XACCH1 | 32h |
| XACCH0 | 33h |
| XACCL3 | 34h |
| XACCL2 | 35h |
| XACCL1 | 36h |
| XACCL0 | 37h |

### 17.3.2.1.13.2 Function

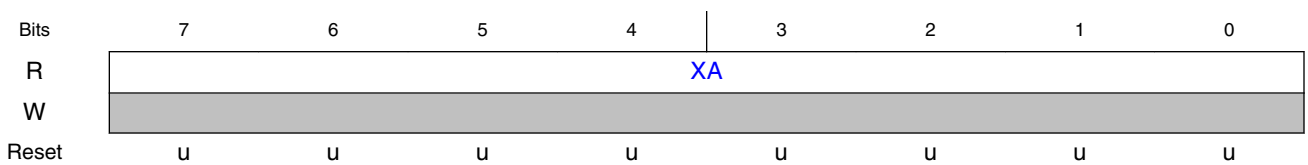
The XACC registers define which primary program flash segments are restricted to data read or execute only or both data and instruction fetches.

The eight XACC registers allow up to 64 restricted segments of equal memory size.

| Primary Execute-only access register | Primary Program flash execute-only access bits |
|---|---|
| XACCH0 | XA[63:56] |
| XACCH1 | XA[55:48] |
| XACCH2 | XA[47:40] |
| XACCH3 | XA[39:32] |
| XACCL0 | XA[31:24] |
| XACCL1 | XA[23:16] |
| XACCL2 | XA[15:8] |
| XACCL3 | XA[7:0] |

During the reset sequence, the XACC registers are loaded with the XACC index located in the program flash erasable IFR. Use the Program Index command to program the primary execute-only access control fields that are loaded during the reset sequence.

### 17.3.2.1.13.3 Diagram



#### 17.3.2.1.13.4 Fields

| Field | Function |
|---|---|
| 7-0<br>XA | Execute-only access control<br>0b - Associated segment is accessible in execute mode only (as an instruction fetch)<br>1b - Associated segment is accessible as data or in execute mode |

### 17.3.2.1.14 Primary Supervisor-only Access Registers (SACCH3 - SACC L0)

#### 17.3.2.1.14.1 Offset

| Register | Offset |
|---|---|
| SACCH3 | 38h |
| SACCH2 | 39h |
| SACCH1 | 3Ah |
| SACCH0 | 3Bh |
| SACCL3 | 3Ch |
| SACCL2 | 3Dh |
| SACCL1 | 3Eh |
| SACCL0 | 3Fh |

#### 17.3.2.1.14.2 Function

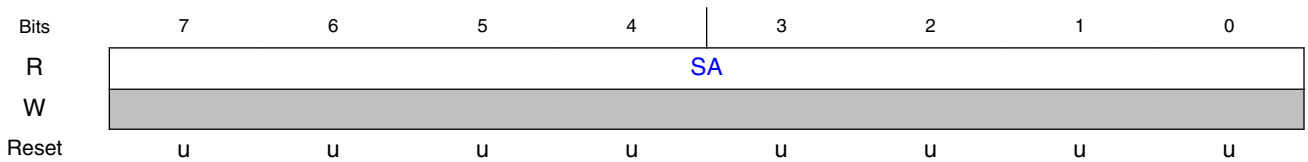
The SACC registers define which primary program flash segments are restricted to supervisor only or user and supervisor access.

The eight SACC registers allow up to 64 restricted segments of equal memory size.

| Primary Supervisor-only access register | Primary Program flash supervisor-only access bits |
|---|---|
| SACCH0 | SA[63:56] |
| SACCH1 | SA[55:48] |
| SACCH2 | SA[47:40] |
| SACCH3 | SA[39:32] |
| SACCL0 | SA[31:24] |
| SACCL1 | SA[23:16] |
| SACCL2 | SA[15:8] |
| SACCL3 | SA[7:0] |

During the reset sequence, the SACC registers are loaded with the SACC index located in the program flash erasable IFR. Use the Program Index command to program the primary supervisor access control fields that are loaded during the reset sequence.

#### 17.3.2.1.14.3 Diagram




#### 17.3.2.1.14.4 Fields

| Field | Function |
|---|---|
| 7-0<br>SA | Primary Supervisor-only access control<br>0b - Associated segment is accessible in supervisor mode only<br>1b - Associated segment is accessible in user or supervisor mode |

### 17.3.2.1.15 Secondary Execute-only Access Registers (XACCSL - XACCSH)

#### 17.3.2.1.15.1 Offset

| Register | Offset |
|---|---|
| XACCSL | 44h |
| XACCSH | 45h |

#### 17.3.2.1.15.2 Function

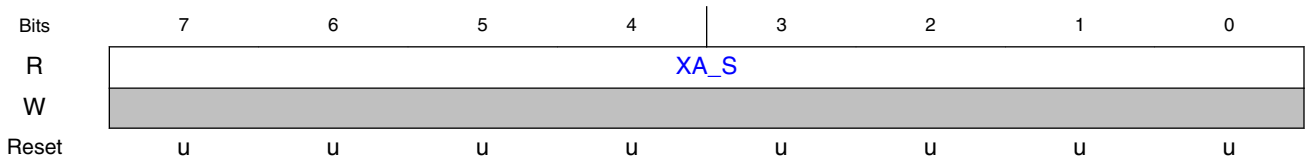
The XACCS registers define which secondary program flash segments are restricted to data read or execute only or both data and instruction fetches.

The two XACCS registers allow up to 16 restricted segments of equal memory size.

| Secondary Execute-only access register | Secondary Program flash execute-only access bits |
|---|---|
| XACCSH | XA_S[15:8] |
| XACCSL | XA_S[7:0] |

During the reset sequence, the XACCS registers are loaded with the XACCS index located in the program flash erasable IFR. Use the Program Index command to program the secondary execute-only access control fields that are loaded during the reset sequence.

#### 17.3.2.1.15.3 Diagram




#### 17.3.2.1.15.4 Fields

| Field | Function |
|---|---|
| 7-0<br>XA_S | Execute-only access control<br>0b - Associated secondary program flash segment is accessible in execute mode only (as an instruction fetch)<br>1b - Associated secondary program flash segment is accessible as data or in execute mode |

### 17.3.2.1.16 Secondary Supervisor-only Access Registers (SACCSL - SACCSH)

#### 17.3.2.1.16.1 Offset

| Register | Offset |
|---|---|
| SACCSL | 4Ch |
| SACCSH | 4Dh |

#### 17.3.2.1.16.2 Function

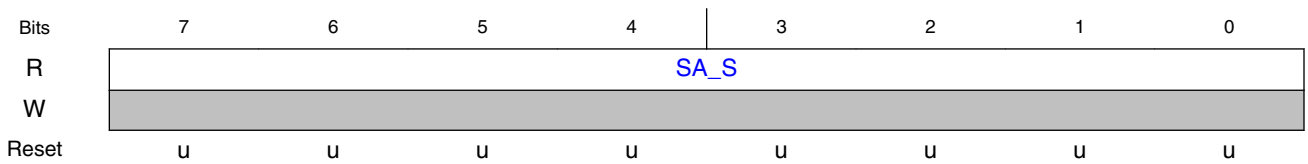
The SACCS registers define which secondary program flash segments are restricted to supervisor only or user and supervisor access.

The two SACC registers allow up to 16 restricted segments of equal memory size.

| Secondary Supervisor-only access register | Secondary Program flash supervisor-only access bits |
|---|---|
| SACCH | SA_S[15:8] |
| SACCL | SA_S[7:0] |

During the reset sequence, the SACCS registers are loaded with the SACCS index located in the program flash erasable IFR. Use the Program Index command to program the secondary supervisor access control fields that are loaded during the reset sequence.

#### 17.3.2.1.16.3 Diagram




#### 17.3.2.1.16.4 Fields

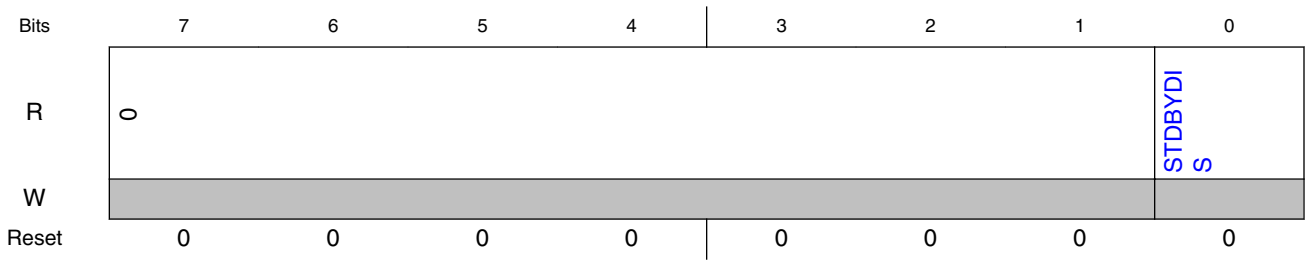
| Field | Function |
|---|---|
| 7-0<br>SA_S | Secondary Supervisor-only access control<br>0b - Associated secondary program flash segment is accessible in supervisor mode only<br>1b - Associated secondary program flash segment is accessible in user or supervisor mode |

### 17.3.2.1.17 Flash Standby Control Register (FSTDBYCTL)

#### 17.3.2.1.17.1 Offset

| Register | Offset |
|---|---|
| FSTDBYCTL | 52h |

### 17.3.2.1.17.2 Diagram




### 17.3.2.1.17.3 Fields

| Field | Function |
|---|---|
| 7-1<br>— | Reserved |
| 0<br>STDBYDIS | Standy Mode Disable<br>To prevent corruption of a read operation, the STDBYx bit must not be written high while the associated flash block is being read.<br>0b - Standby mode enabled for flash blocks selected by STDBYx<br>1b - Standby mode disabled (STDBYx ignored) |

## 17.3.2.1.18 Flash Standby Register (FSTDBY)

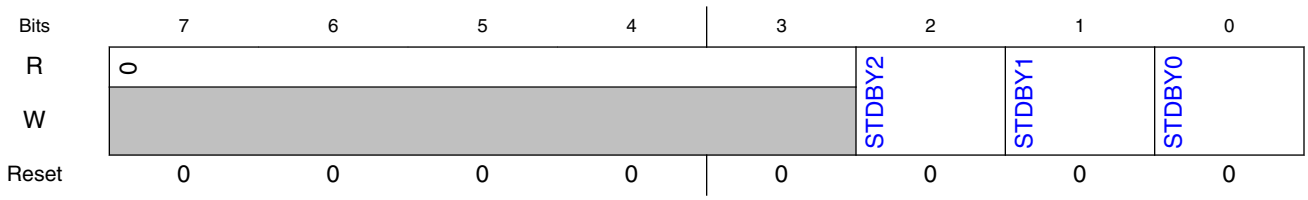
### 17.3.2.1.18.1 Offset

| Register | Offset |
|---|---|
| FSTDBY | 53h |

### 17.3.2.1.18.2 Function

This register specifies which flash blocks are placed into standby mode to reduce power consumption. All assigned bits are readable and writable when STDBYDIS=0 unless CCIF=0 where the FSTDBY register is only readable. STDBYx bits are not writable if a read is active on the associated flash block. The unassigned bits read 0 and are not writable.

#### 17.3.2.1.18.3 Diagram




#### 17.3.2.1.18.4 Fields

| Field | Function |
|---|---|
| 7-3<br>— | Reserved |
| 2<br>STDBY2 | Standy Mode for Flash Block 2<br>To prevent corruption of a read operation, the STDBYx bit must not be written high while the associated flash block is being read.<br>0b - Standby mode not enabled for flash block 2<br>1b - If STDBYDIS is clear, standby mode is enabled for flash block 2 |
| 1<br>STDBY1 | Standy Mode for Flash Block 1<br>0b - Standby mode not enabled for flash block 1<br>1b - If STDBYDIS is clear, standby mode is enabled for flash block 1 (when SWAP=0/1, flash block 1/0 is the inactive block) |
| 0<br>STDBY0 | Standy Mode for Flash Block 0<br>0b - Standby mode not enabled for flash block 0<br>1b - If STDBYDIS is clear, standby mode is enabled for flash block 0 (when SWAP=0/1, flash block 1/0 is the inactive block) |

## 17.4 Functional description

The following sections describe functional details of the flash module.

### 17.4.1 Primary Program flash memory swap

The user can configure the memory map of the program flash space such that either half of the primary program flash memory can exist at relative address 0x0000. This swap feature enables the lower half of the primary program flash space to be operational while the upper half is being updated for future use.

The Swap Control command handles swapping the two halves of primary program flash memory within the memory map. See Swap Control command for details.

## 17.4.2 Flash Program/Erase Protection

Individual regions within the flash memory can be protected from program and erase operations. Protection is controlled by the following registers:

- FPROT*n* — Eight registers protect 64 regions of the primary program flash memory as shown in the following figure

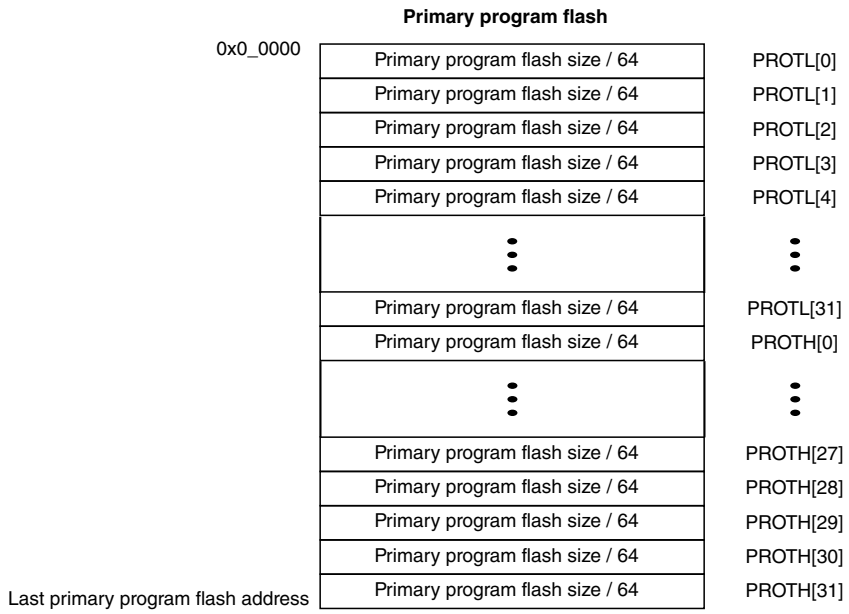



**Figure 17-1. Primary program flash protection**

- FPROTSH/FPROTSL — Two registers protect 16 regions of the secondary program flash memory as shown in the following figure

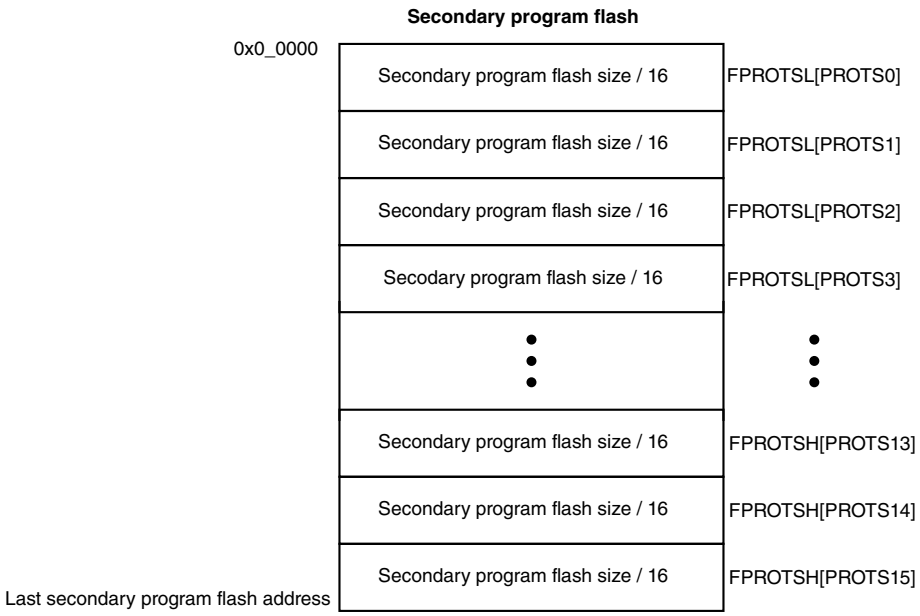



**Figure 17-2. Secondary program flash protection**

**NOTE**

Flash protection features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features. Some features described in the application note may not be available on this device.

## 17.4.3 Flash Access Protection

Individual segments within the program flash memory can be designated for restricted access. Specific flash commands (Program Check, Program Phrase, Erase Flash Block, Erase Flash Sector) monitor FXACC/FXACCS contents to protect flash memory but the FSACC/FSACCS contents do not impact flash command operation.

See AN5112: Using the Kinetis Flash Execute-Only Access Control Feature for further details.

FXACCH/FXACCL/FSACCH/FSACCL registers control access to 64 segments of the primary program flash memory as shown in the following figure:

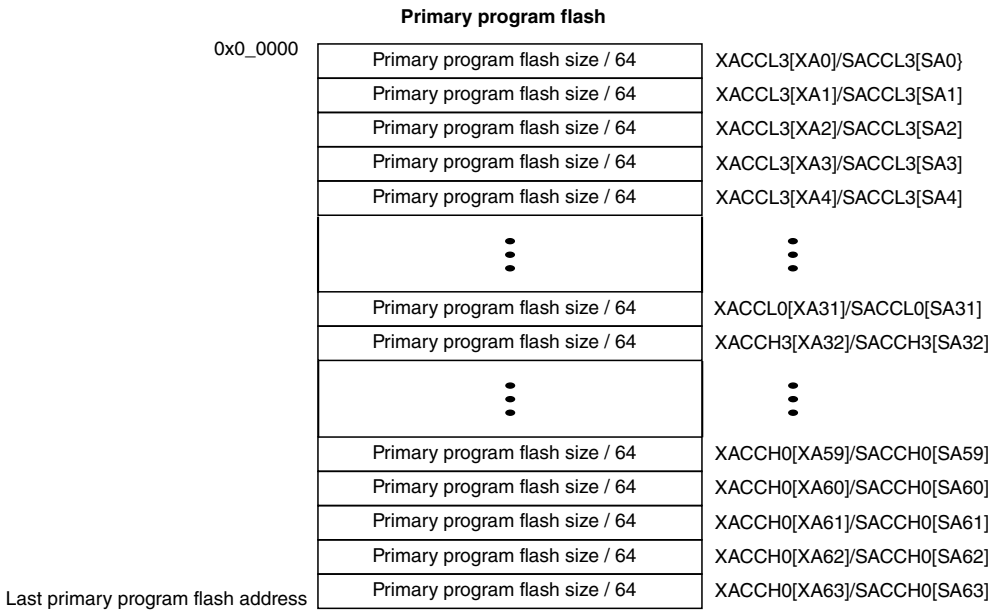



**Figure 17-3. Primary program flash access control**

FXACCSH/FXACCSL/FSACCSH/FSACCSL registers control access to 16 segments of the secondary program flash memory as shown in the following figure:

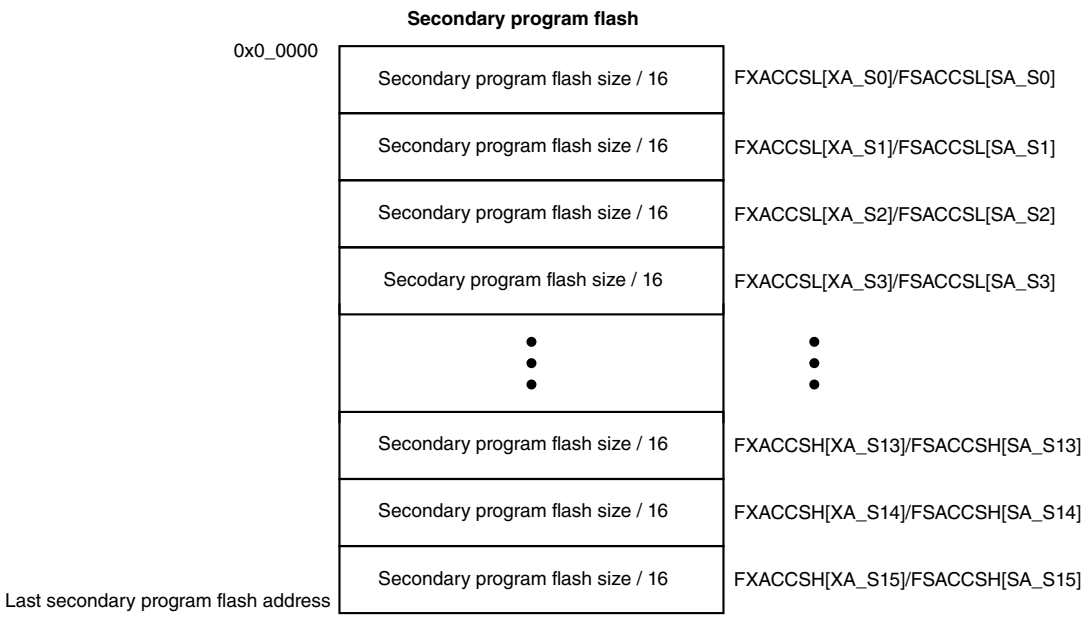



**Figure 17-4. Secondary program flash access control**

## 17.4.4 Interrupts

The flash module can generate interrupt requests to the MCU upon the occurrence of various flash module events. These interrupt events and their associated status and control bits are shown in the following table.

**Table 17-4. Flash Module Interrupt Sources**

| Flash Event | Readable Status Bit | Interrupt Enable Bit |
|---|---|---|
| Flash Command Complete | FSTAT[CCIF] | FCNFG[CCIE] |
| Flash Read Collision Error | FSTAT[RDCOLERR] | FCNFG[RDCOLLIE] |

**Note**

Vector addresses and their relative interrupt priority are determined at the MCU level.

## 17.4.5 Flash Operation in Low-Power Modes

### 17.4.5.1 Wait Mode

When the MCU enters wait mode, the flash module is not affected. The flash module can recover the MCU from wait via the command complete interrupt (see Interrupts).

### 17.4.5.2 Stop Mode

When the MCU requests stop mode, if an flash command is active (CCIF = 0) the command execution completes before the MCU is allowed to enter stop mode.

**CAUTION**

The MCU should never enter stop mode while any flash command is running (CCIF = 0).

**NOTE**

While the MCU is in very-low-power modes (VLPR, VLPW, VLPS), the flash module does not accept flash commands.

## 17.4.6 Flash memory reads and ignored writes

The flash module requires only the flash address to execute a flash memory read. MCU read access is available to all flash memory.

The MCU must not read from the flash memory while commands are running (as evidenced by CCIF=0) on that block. Read data cannot be guaranteed from a block while any command is processing within that block. The block arbitration logic detects any simultaneous access and reports this as a read collision error (see the FSTAT[RDCOLERR] bit).

### 17.4.7 Read while write (RWW)

The following simultaneous accesses are allowed:

- The user may read from one program flash memory block while commands are active in another program flash memory block.

Simultaneous operations are further discussed in Allowed simultaneous flash operations.

### 17.4.8 Flash Program and Erase

All flash functions except read require the user to setup and launch a flash command through a series of peripheral bus writes. The user cannot initiate any more flash commands until notified that the current command has completed. The flash command structure and operation are detailed in Flash Command Operations.

### 17.4.9 Flash Command Operations

Flash command operations are typically used to modify flash memory contents. The next sections describe:

- The command write sequence used to set flash command parameters and launch execution
- A description of all flash commands available

#### 17.4.9.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 17-5. The flash module performs various checks on the command (FCCOB) content and continues with command execution if all requirements are fulfilled.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be zero and the CCIF flag must read 1 to verify that any previous command has completed. If CCIF is zero, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the FCCOB registers are ignored.

Attempts to launch a flash command in VLP or HSRUN mode will be ignored. Attempts to launch a flash command while the radio is active will also be ignored.

#### 17.4.9.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired flash command. The individual registers that make up the FCCOB data set can be written in any order.

#### 17.4.9.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing the FSTAT[CCIF] bit by writing a '1' to it. The CCIF flag remains zero until the flash command completes.

The FSTAT register contains a blocking mechanism, which prevents a new flash command from launching (unable to clear CCIF) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write to clear the error flags, the second write to clear CCIF.

#### 17.4.9.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

   If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. ACCERR reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

   Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, the FSTAT[FPVIOL] (protection error) flag is set.

Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting the FSTAT[CCIF] bit.

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in the FSTAT[MGSTAT0] bit. A flash command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash module sets the FSTAT[CCIF] bit signifying that the command has completed.

The flow for a generic command write sequence is illustrated in the following figure.

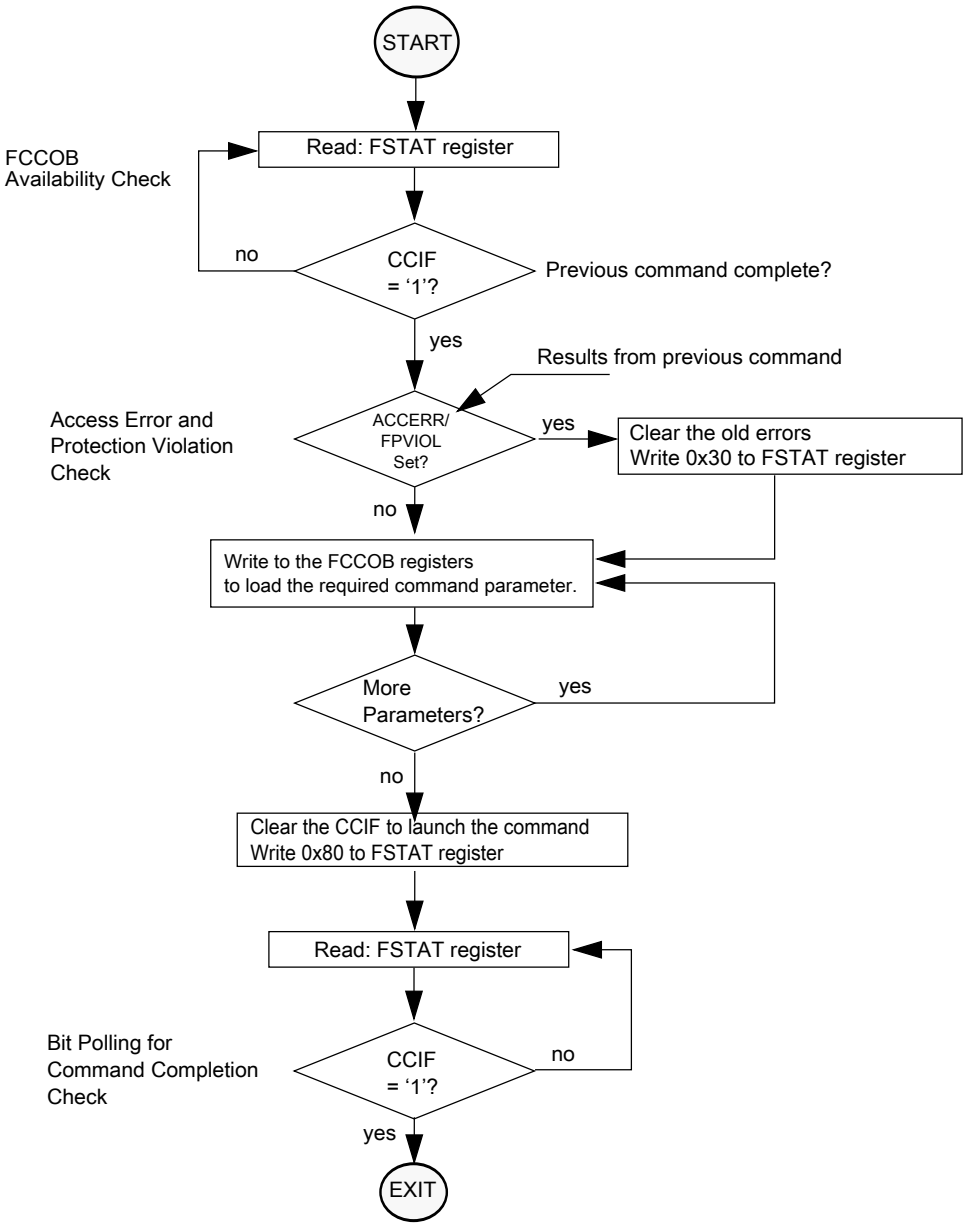



**Figure 17-5. Generic Flash Command Write Sequence Flowchart**

## 17.4.9.2 Flash commands

The following table summarizes the function of all flash commands. If any column is marked with an 'X', the flash command is relevant to that particular memory resource.

| FCMD | Command | Primary Program flash 0 | Primary Program flash 1 | Secondary Program flash 0 | Programming Acceleration RAM | Function |
|---|---|---|---|---|---|---|
| 0x00 | Read 1s Block | × | × | × | | Verify that a program flash block is erased. |
| 0x01 | Read 1s Section | × | × | × | | Verify that a given number of program flash locations from a starting address are erased. |
| 0x02 | Program Check | × | × | × | | Tests previously-programmed phrases at margin read levels. |
| 0x07 | Program Phrase | × | × | × | | Program 8 bytes in a program flash block. |
| 0x08 | Erase Flash Block | × | × | × | | Erase a program flash block. An erase of any flash block is only possible when unprotected. |
| 0x09 | Erase Flash Sector | × | × | × | | Erase all bytes in a program flash sector. |
| 0x0B | Program Section | × | × | × | × | Program data from the Section Program Buffer to a program flash block. |
| 0x0C | Generate CRC | × | × | × | × | Generate CRC signature for selected program flash sectors. |
| 0x40 | Read 1s All Blocks | × | × | × | | Verify that all program flash and the primary program flash erasable IFR are erased then release MCU security. |
| 0x41 | Read Index | IFR | | | | Read 8 bytes of a dedicated 64 byte field in |

*Table continues on the next page...*

| FCMD | Command | Primary Program flash 0 | Primary Program flash 1 | Secondary Program flash 0 | Programming Acceleration RAM | Function |
|---|---|---|---|---|---|---|
| | | | | | | theprimary program flash 0 IFR. |
| 0x43 | Program Index | IFR | | | | One-time program of 8 bytes of a dedicated 64-byte field in theprimary program flash 0 IFR. |
| 0x44 | Erase All Blocks | × | × | × | × | Erase all program flash blocks, theprimary program flash erasable IFR, and the programming acceleration RAM. Then, verify-erase and release MCU security.<br>**NOTE**: An erase is only possible when all memory locations are unprotected. |
| 0x45 | Verify Backdoor Access Key | × | | | | Release MCU security after comparing a set of user-supplied security keys to those stored in the Flash Configuration Field. |
| 0x46 | Swap Control | x | x | | | Handles swap-related activities. |
| 0x49 | Erase All Blocks Unsecure | × | × | × | × | Erase all program flash blocks, theprimary program flash erasable IFR, and the programming acceleration RAM. Then, verify-erase, |

*Table continues on the next page...*

| FCMD | Command | Primary Program flash 0 | Primary Program flash 1 | Secondary Program flash 0 | Programming Acceleration RAM | Function |
|---|---|---|---|---|---|---|
| | | | | | | program the security field to the unsecure state, and release MCU security. An erase occurs even if any flash memory location is protected. |
| 0x4A | Read 1s All Execute-only Segments | x | x | | | Verify that all program flash execute-only (XA and XA_S) segments are erased then release flash access control. |
| 0x4B | Erase All Execute-only Segments | x | x | | | Erase all program flash execute-only (XA and XA_S) segments then release flash access control. |
| 0x81 | Set RAM Function | × | | | × | Switches programming acceleration RAM function between available modes. |

### 17.4.9.3 Allowed simultaneous flash operations

Only the operations marked 'OK' in the following table are permitted to run simultaneously on the program flash memories. Some operations cannot be executed simultaneously because certain hardware resources are shared by the memories.

**Table 17-5. Allowed Simultaneous Memory Operations**

| | | Primary program flash X[1] | | | | Secondary program flash | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Read | Program Phrase | Erase Flash Sector | Erase Flash Block | Read | Program Phrase | Erase Flash Sector | Erase Flash Block |
| Primary program flash Y[1] | Read | | OK | OK | OK | OK | OK | OK | OK |
| | Program Phrase | OK | | | | OK | | | |
| | Erase Flash Sector | OK | | | | OK | | | |
| | Erase Flash Block | OK | | | | OK | | | |

1. Primary program flash X refers to any of the primary program flash blocks (0, 1) and Primary program Fflash Y refers to any of the primary program flash blocks (0, 1), but not the same block. Thus, it is possible to read from any of the blocks while programming or erasing another.

## 17.4.10 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Block, Read 1s Section, Read 1s All Execute-only Segments) and the Program Check command have a margin choice parameter that allows the user to apply non-standard read reference levels to the program flash array reads performed by these commands. Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. All simple (uncommanded) flash array reads to the MCU always use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

**CAUTION**

Factory margin levels must only be used during verify of the initial factory programming.

## 17.4.11 Flash command descriptions

This section describes all flash commands that can be launched by a command write sequence. The flash module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in a FCCOB field for the specific commands. Refer to the error handling table provided for each command.

Ensure that the ACCERR and FPVIOL bits in the FSTAT register are cleared prior to starting the command write sequence. As described in Launch the Command by Clearing CCIF, a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while a flash command is running (CCIF = 0) on that same block. The flash module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

When required by the command, address bit 23 selects between primary program flash memory (=0) and secondary program flash memory (=1).

**CAUTION**

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

### 17.4.11.1 Read 1s Block command

The Read 1s Block command checks to see if an entire flash block has been erased to the specified margin level. The FCCOB flash address bits determine which block is erase-verified.

**Table 17-6. Read 1s Block Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x00 (RD1BLK) |
| 1 | Flash address [23:16] in the flash block to be verified |
| 2 | Flash address [15:8] in the flash block to be verified |
| 3 | Flash address [7:0][1] in the flash block to be verified |
| 4 | Read-1 Margin Choice |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash, 64-bit aligned (Flash address [2:0] = 000) for secondary program flash.

After clearing CCIF to launch the Read 1s Block command, the flash module sets the read margin for 1s according to Table 17-7 and then reads all locations within the selected flash block. If the operation fails to read all 1s (i.e. the flash block is not fully erased), the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s Block operation completes.

**CAUTION**

The Read 1s Block operation will not react to the early indicator for radio activity. Therefore, the Read 1s Block command must not be launched if there is a concern about inadequate power available to support both radio activity and the Read 1s Block operation.

**Table 17-7. Margin Level Choices for Read 1s Block**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 17-8. Read 1s Block Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin choice is specified | FSTAT[ACCERR] |
| Primary program flash is selected and the address is out of primary program flash range | FSTAT[ACCERR] |
| Secondary program flash is selected and the address is out of secondary program flash range | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned for primary program flash, 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

## 17.4.11.2 Read 1s Section command

The Read 1s Section command checks if a section of flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of double-phrases to be verified for primary program flash, phrases for secondary program flash.

**Table 17-9. Read 1s Section Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x01 (RD1SEC) |
| 1 | Flash address [23:16] of the first double-phrase to be verified for primary program flash, phrase for secondary program flash |
| 2 | Flash address [15:8] of the first double-phrase to be verified for primary program flash, phrase for secondary program flash |
| 3 | Flash address [7:0] [1] of the first double-phrase to be verified for primary program flash, phrase for secondary program flash |
| 4 | Number of double-phrases to be verified for primary program flash, phrases for secondary program flash [15:8] |
| 5 | Number of double-phrases to be verified for primary program flash, phrases for secondary program flash [7:0] |
| 6 | Read-1 Margin Choice |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash, 64-bit aligned (Flash address [2:0] = 000) for secondary progra flash .

Upon clearing CCIF to launch the Read 1s Section command, the flash module sets the read margin for 1s according to Table 17-10 and then reads all locations within the specified section of flash memory. If the operation fails to read all 1s (i.e. the flash section is not erased), the FSTAT(MGSTAT0) bit is set. The CCIF flag sets after the Read 1s Section operation completes.

**CAUTION**

The Read 1s Section operation will not react to the early indicator for radio activity. Therefore, the Read 1s Section command must not be launched if there is a concern about inadequate power available to support both radio activity and the Read 1s Section operation if the number of double-phrases for primary program flash, phrases for secondary program flash to be verified is greater than 10,240.

**Table 17-10. Margin Level Choices for Read 1s Section**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 17-11. Read 1s Section Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin code is supplied | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned for primary program flash, 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| The requested section crosses a flash block boundary | FSTAT[ACCERR] |
| The requested number of double-phrases for primary program flash, phrases for secondary program flash is zero | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

### 17.4.11.3 Program Check command

The Program Check command tests a previously programmed flash longword to see if it reads correctly at the specified margin level.

**Table 17-12. Program Check Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x02 (PGMCHK) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Margin Choice |
| 8 | Byte 0 expected data |
| 9 | Byte 1 expected data |
| A | Byte 2 expected data |
| B | Byte 3 expected data |

1. Must be 32-bit aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Program Check command, the flash module sets the read margin for 1s based on the provided margin choice according to Table 17-13. The Program Check operation then reads the specified longword and compares the actual read data to the expected data provided by the FCCOB. If the comparison at margin-1 fails, the MGSTAT0 bit is set. The flash module will then set the read margin for 0s based on the provided margin choice. The Program Check operation will then read the specified longword and compare the actual read data to the expected data provided by the FCCOB. If the comparison at margin-0 fails, the MGSTAT0 bit will be set. The CCIF flag sets after the Program Check operation completes.

The starting address must be 32-bit aligned (the lowest two bits of the byte address must be 00):

- Byte 0 data is expected at the supplied 32-bit aligned address,
- Byte 1 data is expected at byte address specified + 0b01,
- Byte 2 data is expected at byte address specified + 0b10, and
- Byte 3 data is expected at byte address specified + 0b11.

**NOTE**

See the description of margin reads, Margin Read Commands

**Table 17-13. Margin Level Choices for Program Check**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x01 | Read at 'User' margin-1 and 'User' margin-0 |
| 0x02 | Read at 'Factory' margin-1 and 'Factory' margin-0 |

**Table 17-14. Program Check Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 32-bit aligned | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| An invalid margin choice is supplied | FSTAT[ACCERR] |
| Flash address is located in an XA or XA_S controlled segment and the Erase All Blocks, Erase All Blocks Unsecure, Read 1s All Execute-only Segments, Erase All Execute-only Segments or the Read 1s All Blocks command has not successfully completed since the last reset | FSTAT[FPVIOL] |
| Either of the margin reads does not match the expected data | FSTAT[MGSTAT0] |

### 17.4.11.4 Program Phrase command

The Program Phrase command programs eight previously-erased bytes in the flash memory using an embedded algorithm.

**CAUTION**

A Flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a Flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

**Table 17-15. Program Phrase Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x07 (PGM8) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Byte 0 program value |
| 5 | Byte 1 program value |
| 6 | Byte 2 program value |
| 7 | Byte 3 program value |
| 8 | Byte 4 program value |
| 9 | Byte 5 program value |
| A | Byte 6 program value |
| B | Byte 7 program value |

1. Must be 64-bit aligned (Flash address [2:0] = 000)

Upon clearing CCIF to launch the Program Phrase command, the flash module programs the data bytes into the flash memory using the supplied address. The protection status is always checked. If the swap system is enabled, the double-phrase containing the swap indicator address in each half of the primary program flash space is implicitly protected from programming. The targeted flash locations must be currently unprotected (see the description of the FPROT and FPROTS registers) to permit execution of the Program Phrase operation. The programming operation is unidirectional. It can only move NVM bits from the erased state ('1') to the programmed state ('0'). Erased bits that fail to program to the '0' state are flagged as errors in MGSTAT0. The CCIF flag is set after the Program Phrase operation completes.

The starting address must be 64-bit aligned (flash address [2:0] = 000):

- Byte 0 data is written to the starting address ('start'),
- Byte 1 data is programmed to byte address start+0b01,
- Byte 2 data is programmed to byte address start+0b10, and
- Byte 3 data is programmed to byte address start+0b11, etc.

**Table 17-16. Program Phrase Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 64-bit aligned | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| Flash address points to a protected area | FSTAT[FPVIOL] |
| Flash address is located in an XA or XA_S controlled segment and the Erase All Blocks, Erase All Blocks Unsecure, Read 1s All Execute-only Segments, Erase All Execute-only Segments or the Read 1s All Blocks command has not successfully completed since the last reset | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation. | FSTAT[MGSTAT0] |

## 17.4.11.5 Erase Flash Block Command

The Erase Flash Block operation erases all addresses in a single flash block.

**Table 17-17. Erase Flash Block Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x08 (ERSBLK) |
| 1 | Flash address [23:16] in the flash block to be erased |
| 2 | Flash address [15:8] in the flash block to be erased |
| 3 | Flash address [7:0][1] in the flash block to be verified |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash, 64-bit aligned (Flash address [2:0] = 000) for secondary program flash.

Upon clearing CCIF to launch the Erase Flash Block command, the flash module erases the main array of the selected flash block and verifies that it is erased. The Erase Flash Block command aborts and sets the FSTAT[FPVIOL] bit if any region within the block is protected (see the description of the program flash protection (FPROT and FPROTS) registers). If the swap system is enabled, the swap indicator address is implicitly protected from block erase unless the swap system is in the UPDATE or UPDATE-ERASED state and the primary program flash block being erased is the non-active block that contains the swap indicator address. If the erase verify fails, the MGSTAT0 bit in FSTAT is set. The CCIF flag sets after the Erase Flash Block operation completes.

**Table 17-18. Erase Flash Block Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Primary program flash is selected and the address is out of primary program flash range | FSTAT[ACCERR] |
| Secondary program flash is selected and the address is out of secondary program flash range | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned for primary program flash, 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| Any area of the selected flash block is protected | FSTAT[FPVIOL] |
| Flash address is located in an XA or XA_S controlled segment and the Erase All Blocks, Erase All Blocks Unsecure, Read 1s All Execute-only Segments, Erase All Execute-only Segments or the Read 1s All Blocks command has not successfully completed since the last reset | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s Block command to verify all bits are erased.

### 17.4.11.5.1 Impact of radio activity on Erase Flash Block operation

The RSIM module provides an early indicator to the flash module that the radio is going active and also indicates when the radio goes inactive. If the radio active indicator is asserted while the Erase Flash Block operation is active, the operation will stall to reduce power consumption before the radio goes active. The Erase Flash Block operation will resume after the radio active indicator negates. FSTAT[CCIF] remains clear during the stall to prevent disruption of the Erase Flash Block operation while the block being erased remains unavailable for read operations.

The following figure shows how the Erase Flash Block operation stalls and resumes based on radio activity.

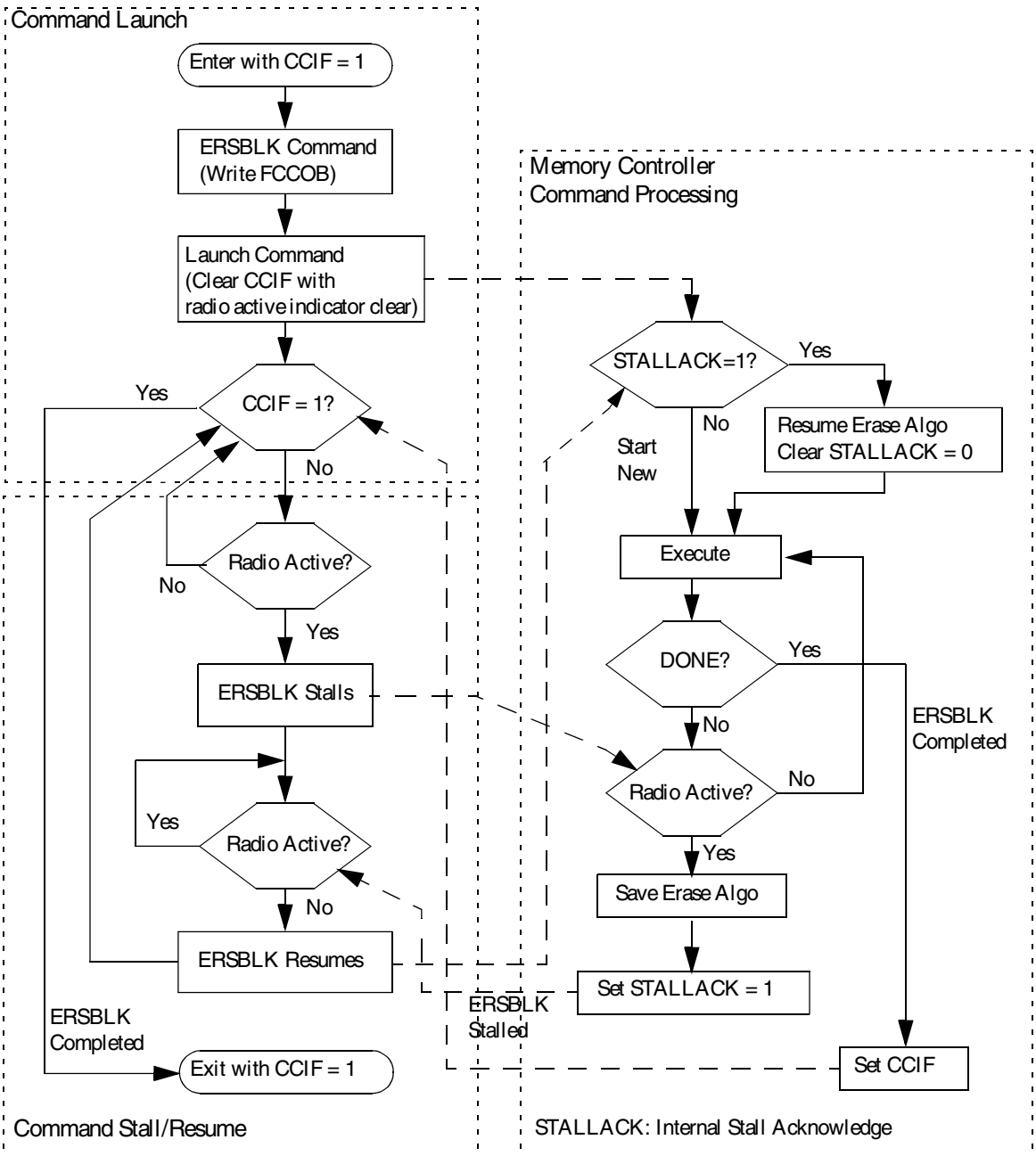



**Figure 17-6. Radio Active Impact on Erase Flash Block Operation**

## 17.4.11.6 Erase Flash Sector command

The Erase Flash Sector operation erases all addresses in a flash sector.

**Table 17-19. Erase Flash Sector Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x09 (ERSSCR) |

*Table continues on the next page...*

**Table 17-19. Erase Flash Sector Command FCCOB Requirements (continued)**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 1 | Flash address [23:16] in the flash sector to be erased |
| 2 | Flash address [15:8] in the flash sector to be erased |
| 3 | Flash address [7:0][1] in the flash block to be verified |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash, 64-bit aligned (Flash address [2:0] = 000) for secondary program flash.

Upon clearing CCIF to launch the Erase Flash Sector command, the flash module erases the selected flash sector and then verifies that it is erased. The Erase Flash Sector command aborts if the selected sector is protected (see the description of the FPROT and FPROTS registers). If the swap system is enabled, the swap indicator address in each primary program flash block is implicitly protected from sector erase unless the swap system is in the UPDATE or UPDATE-ERASED state and the primary program flash sector containing the swap indicator address being erased is in the non-active block. If the erase-verify fails the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Erase Flash Sector operation completes. The Erase Flash Sector command is suspendable (see the FCNFG[ERSSUSP] bit and Suspending an Erase Flash Sector Operation).

**Table 17-20. Erase Flash Sector Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid Flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned for primary program flash, 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| The selected flash sector is protected | FSTAT[FPVIOL] |
| Flash address is located in an XA or XA_S controlled segment and the Erase All Blocks, Erase All Blocks Unsecure, Read 1s All Execute-only Segments, Erase All Execute-only Segments or the Read 1s All Blocks command has not successfully completed since the last reset | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s Section command to verify all bits are erased.

### 17.4.11.6.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit when CCIF, ACCERR, and FPVIOL are clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see Erase Flash Sector command), the flash module samples the state of the ERSSUSP bit at convenient

points. If the flash module detects that the ERSSUSP bit is set, the Erase Flash Sector operation is suspended and sets CCIF. While ERSSUSP is set, all writes to flash registers are ignored except for writes to the FSTAT FSTDBY, and FCNFG registers.

If an Erase Flash Sector operation effectively completes before the flash module detects that a suspend request has been made, the flash module clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been successfully suspended, the flash module sets CCIF and leaves the ERSSUSP bit set. While CCIF is set, the ERSSUSP bit can only be cleared to prevent the withdrawal of a suspend request before the flash module has acknowledged it.

Suspending an Erase Flash Sector opeartion has priority over stall due to radio activity.

#### 17.4.11.6.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The flash module acknowledges the request to resume a suspended operation by clearing the ERSSUSP bit. A new suspend request can then be made by setting ERSSUSP. A single Erase Flash Sector operation can be suspended and resumed multiple times.

There is a minimum elapsed time limit of 4.3 msec between the request to resume the Erase Flash Sector operation (CCIF is cleared) and the request to suspend the operation again (ERSSUSP is set). This minimum time period is required to ensure that the Erase Flash Sector operation will eventually complete. If the minimum period is continually violated, i.e. the suspend requests come repeatedly and too quickly, no forward progress is made by the Erase Flash Sector algorithm. The resume/suspend sequence runs indefinitely without completing the erase.

#### 17.4.11.6.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the flash module starts the new command using the new FCCOB contents.

**Note**

Aborting the erase leaves the bitcells in an indeterminate, partially-erased state. Data in this sector is not reliable until a new erase command fully completes.

The following figure shows how to suspend and resume the Erase Flash Sector operation.

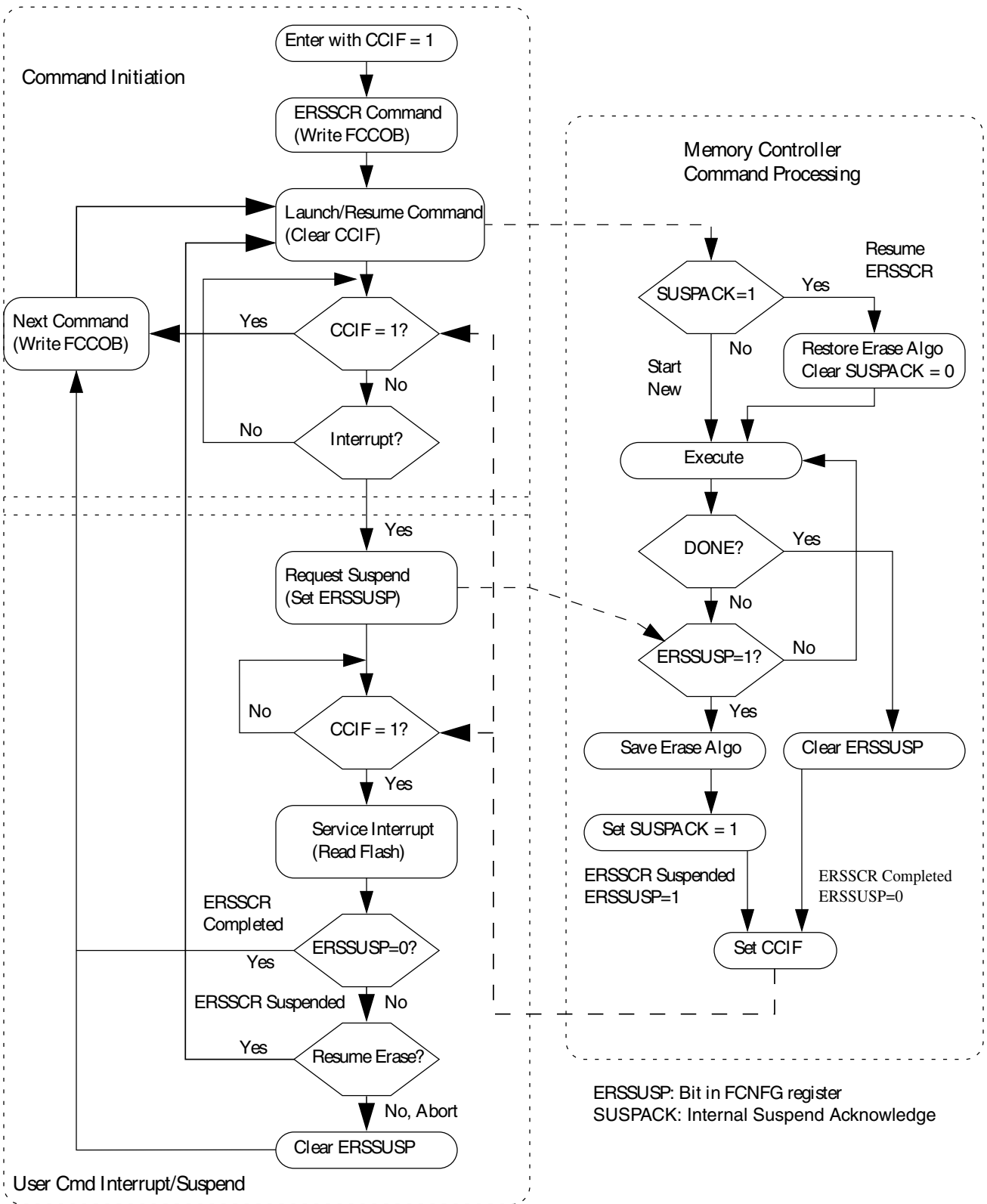



**Figure 17-7. Suspend and Resume of Erase Flash Sector Operation**

#### 17.4.11.6.4 Impact of radio activity on Erase Flash Sector operation

The RSIM module provides an early indicator to the flash module that the radio is going active and also indicates when the radio goes inactive. If the radio active indicator is asserted while the Erase Flash Sector operation is active, the operation will stall to reduce power consumption before the radio goes active. The Erase Flash Sector operation will resume after the radio active indicator negates. FSTAT[CCIF] remains clear during the stall to prevent disruption of the Erase Flash Sector operation while the block containing the sector being erased remains unavailable for read operations.

The following figure shows how the Erase Flash Sector operation stalls and resumes based on radio activity.

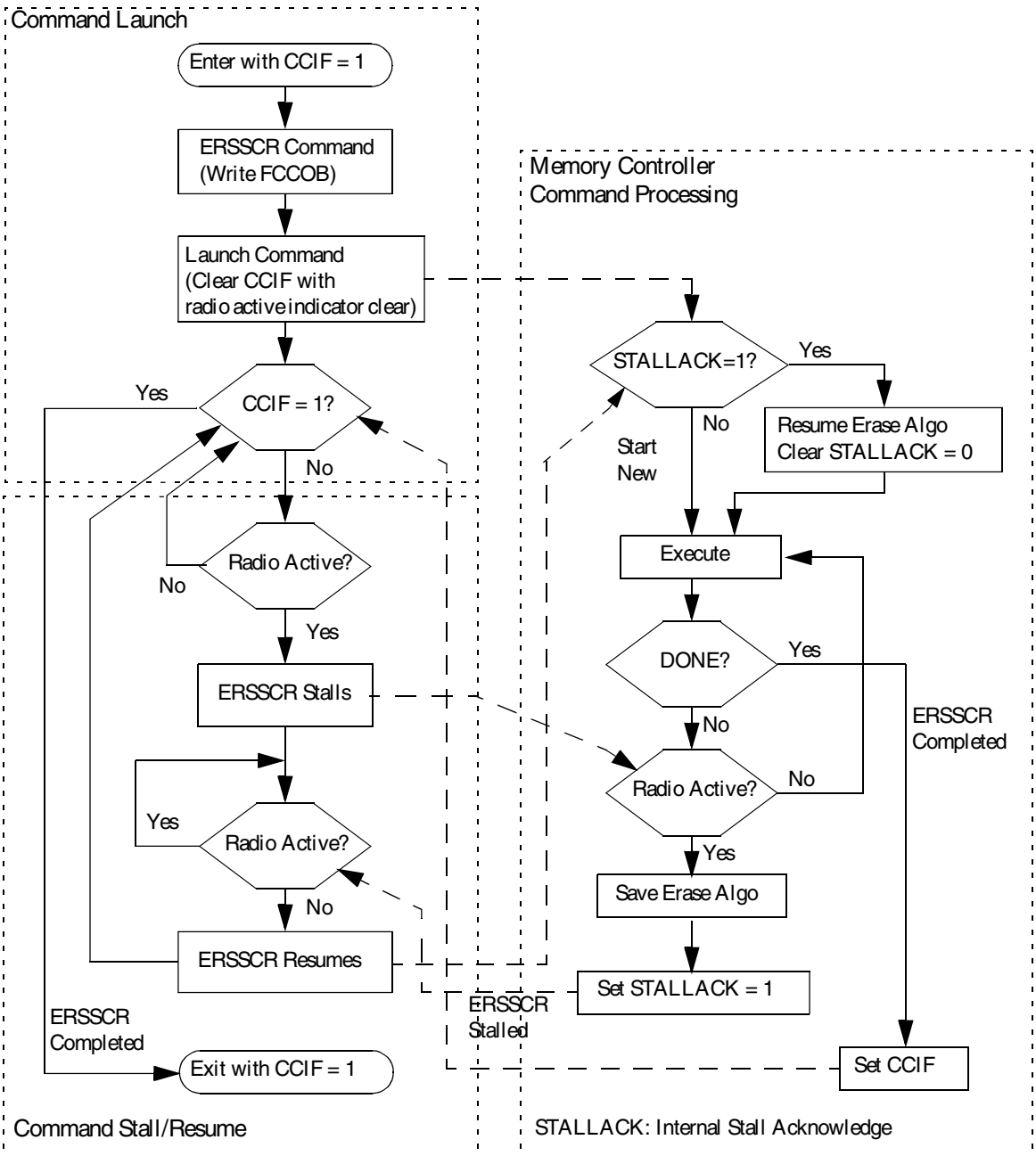



**Figure 17-8. Radio Active Impact on Erase Flash Sector Operation**

## 17.4.11.7 Program Section command

The Program Section operation programs the data found in the section program buffer to previously erased locations in the flash memory using an embedded algorithm. Data is preloaded into the section program buffer (see Flash sector programming).

The section program buffer is limited to the lower quarter of the programming acceleration RAM (relative byte addresses 0x0000-0x03FF - be sure to check your device specific memory map for the location of the programming acceleration RAM). Data written to the remainder of the programming acceleration RAM is ignored and may be overwritten by the flash module during Program Section command execution.

**CAUTION**

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

**Table 17-21. Program Section Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x0B (PGMSEC) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Number of double-phrases for primary program flash, phrases for secondary program flash to program [15:8] |
| 5 | Number of double-phrases for primary program flash, phrases for secondary program flash to program [7:0] |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash, 64-bit aligned (Flash address [2:0] = 000) for secondary program flash.

Upon clearing CCIF to launch the Program Section command, the flash module will block access to the programming acceleration RAM and program the data residing in the Section Program Buffer into the flash memory starting at the flash address provided. The starting address must be unprotected (see the description of the FPROT and FPROTS registers) to permit execution of the Program Section operation. If the swap system is enabled, the double-phrase containing the swap indicator address in each half of the primary program flash space is implicitly protected from programming. If the double-phrase containing the swap indicator address is encountered during the Program Section operation, it will be bypassed without setting FPVIOL and the contents will not be programmed. Programming, which is not allowed to cross a flash sector boundary, continues until all requested double-phrases for primary program flash, phrases for secondary program flash have been programmed. After the Program Section operation completes, the CCIF flag sets. The contents of the Section Program Buffer are not changed by the Program Section operation unless the swap system is enabled and the Program Section command is targeting the double-phrase containing the swap indicator in which case that double-phrase is changed to all ones.

**CAUTION**

The Program Section operation will not react to the early indicator for radio activity. Therefore, the Program Section command must not be launched if there is a concern about inadequate power available to support both radio activity and the Program Section operation.

**Table 17-22. Program Section Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid flash address is supplied | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned for primary program flash, 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| The requested section crosses a flash sector boundary | FSTAT[ACCERR] |
| The requested number of double-phrases for primary program flash, phrases for secondary program flash is zero | FSTAT[ACCERR] |
| The space required to store data for the requested number of double-phrases for primary program flash, phrases for secondary program flash is more than one quarter the size of the programming acceleration RAM | FSTAT[ACCERR] |
| The flash address falls in a protected area | FSTAT[FPVIOL] |
| The requested flash section is located in an XA or XA_S controlled segment and the Erase All Blocks, Erase All Blocks Unsecure, Read 1s All Execute-only Segments, Erase All Execute-only Segments or the Read 1s All Blocks command has not successfully completed since the last reset | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation | FSTAT[MGSTAT0] |

#### 17.4.11.7.1 Flash sector programming

The process of programming an entire flash sector using the Program Section command is as follows:

1. If required, execute the Set RAM Function command to make the programming acceleration RAM available and initialize the RAM to all ones.
2. Launch the Erase Flash Sector command to erase the flash sector to be programmed.
3. Beginning with the starting address of the programming acceleration RAM , sequentially write enough data to the RAM to fill an entire flash sector, or as much data is allowed due to RAM size versus flash sector size. This area of the RAM serves as the section program buffer. The section program buffer can be written to while the operation launched in step 2 is executing, i.e. while CCIF = 0.
4. Execute the Program Section command to program the contents of the section program buffer into the selected flash sector.
5. Repeat steps 3 through 4 to complete the entire flash sector, if necessary.
6. To program additional flash sectors, repeat steps 2 through 5.

### 17.4.11.8 Generate CRC command

The Generate CRC command will generate a signature based on the contents of the selected flash sectors using an embedded CRC generator compatible with the Kinetis CRC module using the following fixed parameters:

- WIDTH = 32
- POLYNOMIAL = 0x04C11DB7
- REFIN = True
- REFOUT = True
- XOROUT = 0xFFFF_FFFF

**Table 17-23. Generate CRC Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x0C (CRCGEN) |
| 1 | Flash address in the starting sector [23:16] |
| 2 | Flash address in the starting sector [15:8] |
| 3 | Flash address in the starting sector [7:0][1] |
| 4 | Number of sectors to process |
| 5 | Seed Option:<br>0 - Seed with 0xFFFF_FFFF<br>1 - Seed using the provided optional seed |
| 8 | Optional Seed [31:24] |
| 9 | Optional Seed [23:16] |
| A | Optional Seed [15:8] |
| B | Optional Seed [7:0] |
| Returned values | |
| 8 | CRC signature [31:24] |
| 9 | CRC signature [23:16] |
| A | CRC signature [15:8] |
| B | CRC signature [7:0] |

1. Must be 128-bit aligned (Flash address [3:0] = 0000) for primary program flash and 64-bit aligned (Flash address [2:0] = 000) for secondary program flash.

To allow access to the flash block that the Generate CRC command is executing on, the Set RAM Function command must be run with the 0xCC option resulting in FCNFG[CRCRDY] = 1. If no access to the flash block is required while running the Generate CRC command, the last Set RAM Function command run after reset must not be run with the 0xCC option so that FCNFG[CRCRDY] = 0.

Upon clearing CCIF to launch the Generate CRC command, the flash module will process flash data starting with the first address in the sector containing the flash address provided and will continue until all sectors requested have been processed. No protection checks are made. While the Generate CRC command executes, read access to the flash block containing the selected sector(s) is not blocked, i.e., read collisions are avoided. The CCIF flag sets after the Generate CRC operation completes.

**Table 17-24. Generate CRC Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| An invalid Flash address is supplied | FSTAT[ACCERR] |
| Flash address in not 128-bit aligned for primary program flash and 64-bit aligned for secondary program flash | FSTAT[ACCERR] |
| Flash block containing the flash address is in standby mode | FSTAT[ACCERR] |
| Requested number of sectors is 0 | FSTAT[ACCERR] |
| Requested number of sectors crosses a flash block boundary | FSTAT[ACCERR] |

### 17.4.11.9 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the flash blocks and erasable IFR have been erased to the specified read margin level, if applicable, and releases security if the readout passes, i.e. all data reads as '1'.

**Table 17-25. Read 1s All Blocks Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x40 (RD1ALL) |
| 1 | Read-1 Margin Choice |

Upon clearing CCIF to launch the Read 1s All Blocks command, the flash module:

- sets the read margin for 1s according to Table 17-26,
- checks the contents of the program flash and erasable IFR are in the erased state.

If the flash module confirms that these memory resources are erased, access control is disabled and security is released by setting the FSEC[SEC] field to the unsecure state. If the read fails, i.e. all flash memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set. The CRCRDY and RAMRDY bits are clear during the Read 1s All Blocks operation and are restored at the end of the Read 1s All Blocks operation. The CCIF flag sets after the Read 1s All Blocks operation completes.

**CAUTION**

The Read 1s All Blocks operation will not react to the early indicator for radio activity. Therefore, the Read 1s All Blocks command must not be launched if there is a concern about inadequate power available to support both radio activity and the Read 1s All Blocks operation.

**Table 17-26. Margin Level Choices for Read 1s All Blocks**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 17-27. Read 1s All Blocks Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin choice is specified | FSTAT[ACCERR] |
| Any flash block is in standby mode | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

## 17.4.11.10 Read Index Command

The Read Index command provides read access to special 8-byte records located in the primary program flash IFR (see Non-erasable IFR Map and Erasable IFR Map).

**Table 17-28. Read Index Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x41 (RDINDEX) |
| 1 | Record index (0x00 - 0x1F / 0x80 - 0xAF) |
| Returned Values | |
| 4 | Record byte 0 value |
| 5 | Record byte 1 value |
| 6 | Record byte 2 value |
| 7 | Record byte 3 value |
| 8 | Record byte 4 value |
| 9 | Record byte 5 value |
| A | Record byte 6 value |
| B | Record byte 7 value |

Upon clearing CCIF to launch the Read Index command, the selected 8-byte record is read from a primary program flash IFR and stored in the FCCOB register. The CCIF flag sets after the Read Index operation completes.

Valid record index values for the Read Index command range from 0x00 to 0x1F and 0x80 to 0xAF unless the record index is reserved. Records with an index between 0x00 - 0x0F and 0x80 - 0x9F are also programmable with restrictions using the Program Index command. Records with an index at or above 0x80 are erasable using the Erase All Blocks or Erase All Blocks Unsecure commands. During execution of the Read Index command, any attempt to read addresses within the primary program flash block containing these records will return invalid data.

**Table 17-29. Read Index Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Primary Program flash block 0 is in standby mode | FSTAT[ACCERR] |
| An invalid or reserved record index is supplied | FSTAT[ACCERR] |

### 17.4.11.11 Program Index command

The Program Index command enables programming of special 8-byte records located in the primary program flash IFR (see Non-erasable IFR Map and Erasable IFR Map).

**Table 17-30. Program Index Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x43 (PGMINDEX) |
| 1 | Record index (0x00 - 0x0F / 0x80 - 0x9F / 0xB0 - 0xFF) |
| 2 | Not Used |
| 3 | Not Used |
| 4 | Record byte 0 value |
| 5 | Record byte 1 value |
| 6 | Record byte 2 value |
| 7 | Record byte 3 value |
| 8 | Record byte 4 value |
| 9 | Record byte 5 value |
| A | Record byte 6 value |
| B | Record byte 7 value |

Upon clearing CCIF to launch the Program Index command, the operation first verifies that the selected record is erased. If erased, the selected record is programmed using the values provided. The Program Index command also verifies that the programmed values read back correctly. The CCIF flag sets after the Program Index operation completes.

Valid record index values for the Program Index command range from 0x00 to 0x0F, 0x80 to 0x9F, and 0xB0 to 0xFF unless the record index is reserved. Records with an index between 0x00 - 0x0F and 0x80 - 0x9F are also readable with restrictions using the Read Index command. Records with an index at or above 0x80 are erasable using the Erase All Blocks or Erase All Blocks Unsecure commands. During execution of the Program Index command, any attempt to read addresses within the primary program flash block containing these records will return invalid data.

Any attempt to program the SEC0 security record using the Program Index command that does not result in the ACCERR flag being set will first program the SEC1 security record to 0x0000.

**Table 17-31. Program Index Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Primary Program flash block 0 is in standby mode | FSTAT[ACCERR] |
| An invalid or reserved record index is supplied | FSTAT[ACCERR] |
| The requested record has already been programmed to a non-erased value[1] | FSTAT[ACCERR] |
| Any errors have been encountered during the verify operation | FSTAT[MGSTAT0] |

1. If a record is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Index command is allowed to execute again on that same record.

## 17.4.11.12 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory including the erasable IFR, initializes the programming acceleration RAM, and releases MCU security.

**Table 17-32. Erase All Blocks Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x44 (ERSALL) |

Upon clearing CCIF to launch the Erase All Blocks command, the flash module erases all flash memory, the erasable IFR space,and the programming acceleration RAM, then verifies that all are erased. If the flash module verifies that all flash memories and the programming acceleration RAM were properly erased, access control is disabled and security is released by setting the FSEC[SEC] field to the unsecure state and the

FCNFG[RAMRDY] bit is set. The Erase All Blocks command aborts if any flash region is protected. The swap indicator address in the primary program flash blocks are not implicitly protected from the erase operation. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Erase All Blocks operation completes.

Access control determined by the contents of the FXACC and FXACCS registers will not block execution of the Erase All Blocks command. The contents of the FXACC, FSACC, FXACCS and FSACCS registers will not be impacted by the execution of the Erase All Blocks command but the associated FXACC, FSACC, FXACCS and FSACCS index fields will be erased. After completion of the Erase All Blocks command, access control is disabled until the next reset of the flash module or the Read 1s All Blocks command is executed and fails (FSTAT[MGSTAT0] is set).

**CAUTION**

The Erase All Blocks operation will not react to the early indicator for radio activity. Therefore, the Erase All Blocks command must not be launched if there is a concern about inadequate power available to support both radio activity and the Erase All Blocks operation.

**Table 17-33. Erase All Blocks Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Any flash block is in standby mode | FSTAT[ACCERR] |
| Any region of the flash memory is protected | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation[1] | FSTAT[MGSTAT0] |

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

### 17.4.11.12.1 Triggering an erase all external to the flash module

The functionality of the Erase All Blocks/Erase All Blocks Unsecure command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the FCCOB0 register must not contain 0x44 and no flash block can be in standby mode and the flash module cannot be in HSRUN mode. When invoked, the erase all function erases all program flash memory and the program flash erasable IFR space regardless of the state of the FSTAT[ACCERR and FPVIOL] flags or the protection settings or the state of the flash swap system. If the post-erase verify passes, access control determined by the contents of the FXACC and FXACCS registers are disabled and the routine releases security by setting the

FSEC[SEC] field to the unsecure state and the FCNFG[RAMRDY] bit sets. The security field in the Flash Configuration Field is also programmed to the unsecure state. The status of the erase all request is reflected in the FCNFG[ERSAREQ] bit. The FCNFG[ERSAREQ] bit is cleared once the operation completes and the normal FSTAT error reporting, except FPVIOL, is available as described in Erase All Blocks Command/ Erase All Blocks Unsecure Command.

**CAUTION**

Since the swap related fields in the program flash erasable IFR are erased during the Erase All Blocks/Erase All Blocks Unsecure command operation, the swap system becomes uninitialized. The Swap Control command must be run with the initialization code to set the swap indicator address and initialize the swap system.

**CAUTION**

The Erase All Blocks operation will not react to the early indicator for radio activity. Therefore, the Erase All pin must not be asserted if there is a concern about inadequate power available to support both radio activity and the Erase All Blocks operation.

### 17.4.11.13 Verify Backdoor Access Key command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see Flash commands by mode). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match the backdoor keys located in the Flash Configuration Field.

**Table 17-34. Verify Backdoor Access Key Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x45 (VFYKEY) |
| 1-3 | Not Used |
| 4 | Key Byte 0 |
| 5 | Key Byte 1 |
| 6 | Key Byte 2 |
| 7 | Key Byte 3 |
| 8 | Key Byte 4 |
| 9 | Key Byte 5 |

*Table continues on the next page...*

**Table 17-34. Verify Backdoor Access Key Command FCCOB Requirements (continued)**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| A | Key Byte 6 |
| B | Key Byte 7 |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the flash module checks the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the flash module sets the FSTAT[ACCERR] bit and terminates. If the command is enabled, the flash module compares the key provided in FCCOB to the backdoor key located in the Flash Configuration Field. If the backdoor keys match, the FSEC[SEC] field is changed to the unsecure state and security is released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are immediately aborted and the FSTAT[ACCERR] bit is (again) set to 1 until a reset of the flash module occurs. If the entire 8-byte key is all zeros or all ones, the Verify Backdoor Access Key command fails with an access error. The CCIF flag sets after the Verify Backdoor Access Key operation completes.

**NOTE**

*Knowledge of the backdoor access keys supersedes the NXP lockout mechanism provided by the FSLACC bits in the FSEC register. If NXP (or anyone) knows the keys and KEYEN is enabled, they get in.*

**Table 17-35. Verify Backdoor Access Key Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Primary Program flash block 0 is in standby mode | FSTAT[ACCERR] |
| The supplied key is all-0s or all-1s | FSTAT[ACCERR] |
| An incorrect backdoor key is supplied | FSTAT[ACCERR] |
| Backdoor key access has not been enabled (see the description of the FSEC register) | FSTAT[ACCERR] |
| This command is launched and the backdoor key has mismatched since the last power down reset | FSTAT[ACCERR] |

## 17.4.11.14 Swap Control command

The Swap Control command handles specific activities associated with swapping the two halves of program flash memory within the memory map.

**Table 17-36. Swap Control Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x46 (SWAP) |
| 1 | Flash address [23:16] |
| 2 | Flash address [15:8] |
| 3 | Flash address [7:0][1] |
| 4 | Swap Control Code:<br>• 0x01 - Initialize Swap System<br>• 0x02 - Set Swap in Update State<br>• 0x04 - Set Swap in Complete State<br>• 0x08 - Report Swap Status<br>• 0x10 - Disable Swap System |
| Returned values | |
| 5 | Current Swap Mode:<br>• 0x00 - Uninitialized<br>• 0x01 - Ready<br>• 0x02 - Update<br>• 0x03 - Update-Erased<br>• 0x04 - Complete<br>• 0x05 - Disabled |
| 6 | Current Swap Block Status:<br>0x00 - primary program flash block 0 at 0x0_0000<br>0x01 - primary program flash block 1 at 0x0_0000 |
| 7 | Next Swap Block Status (after any reset):<br>0x00 - primary program flash block 0 at 0x0_0000<br>0x01 - primary program flash block 1 at 0x0_0000 |

1. Must be 128-bit aligned (Flash address [3:0] = 0000).

Upon clearing CCIF to launch the Swap Control command, the flash module will handle swap-related activities based on the Swap Control code provided in FCCOB4 as follows:

- 0x01 (Initialize Swap System to UPDATE-ERASED State) - After verifying that the current swap state is UNINITIALIZED, and that both phrases which will contain the swap indicators (located in each half of the primary program flash memory within the relative double-phrase flash address provided) are erased, and that the flash address provided is in the lower half of the primary program flash memory, the flash address provided (shifted with bits[3:0] removed) will be programmed into the SWAPADDR index field located in the primary program flash erasable IFR. After the swap indicator address has been programmed, the swap enable word (SWAPEN index field) will be programmed to 0x0000. After the swap enable word has been programmed, the swap indicator located in the lower half of the primary program flash memory will be programmed to 0xFF00.
- 0x02 (Progress Swap to UPDATE State) - After verifying that the current swap state is READY and that the aligned flash address provided matches the one stored in the

SWAPADDR index field, the swap indicator located in the currently active primary program flash block will be programmed to 0xFF00.

- 0x04 (Progress Swap to COMPLETE State) - After verifying that the current swap state is UPDATE-ERASED and that the aligned flash address provided matches the one stored in the SWAPADDR index field, the swap indicator located in the currently active primary program flash block will be programmed to 0x0000. Before executing with this Swap Control code, the user must erase the non-active swap indicator using the Erase Flash Block or Erase Flash Sector commands and update the application code or data as needed.
- 0x08 (Report Swap Status) - After verifying that the aligned flash address provided is in the lower half of the primary program flash memory, the status of the swap system will be reported as follows:
  - FCCOB5 (Current Swap State) - indicates the current swap state based on the status of the swap disable word (SWAPDIS index field), swap enable word and the swap indicators. If the MGSTAT0 flag is set after command completion, the swap state returned was not successfully transitioned from and the appropriate swap command code must be attempted again. If the current swap state is UPDATE and the non-active swap indicator is 0xFFFF, the current swap state is changed to UPDATE-ERASED.
  - FCCOB6 (Current Swap Block Status) - indicates which primary program flash block is currently located at relative flash address 0x0_0000.
  - FCCOB7 (Next Swap Block Status) - indicates which primary program flash block will be located at relative flash address 0x0_0000 after the next reset of the flash module.
- 0x10 (Disable Swap System) - After verifying that the current swap state is UNINITIALIZED , the swap disable word is programmed to 0x0000 and the swap system changed to the DISABLED state with primary program flash block 0 located at relative flash address 0x0_0000.

**NOTE**

It is recommended that the user execute the Swap Control command to report swap status (code 0x08) after any reset to determine if issues with the swap system were detected during the swap state determination procedure.

**NOTE**

It is recommended that the user write 0xFF to FCCOB5, FCCOB6, and FCCOB7 since the Swap Control command will not always return the swap state and status fields when an ACCERR is detected.

The CCIF flag sets after the Swap Control operation completes.

The swap indicators are implicitly protected from being programmed during Program Phrase or Program Section command operations and are implicitly unprotected during Swap Control command operations. The swap indicators are implicitly protected from being erased during Erase Flash Block and Erase Flash Sector command operations unless the swap indicator being erased is in the non-active primary program flash block and the swap system is in the UPDATE or UPDATE-ERASED state. The Erase All Blocks or Erase All Blocks Unsecure command or erase-all function can be used to place the swap system in the UNINITIALIZED state.

**Table 17-37. Swap Control Command Error Handling**

| Error Condition | Swap Control Code | Error Bit |
|---|---|---|
| Command not available in current mode/security[1] | All | FSTAT[ACCERR] |
| Flash address is not in the lower half of primary program flash memory | All | FSTAT[ACCERR] |
| Flash address is not 128-bit aligned | All | FSTAT[ACCERR] |
| Any primary program flash block is in standby mode | All | FSTAT[ACCERR] |
| Swap system is in the disabled state | 1, 2, 4, 10 | FSTAT[ACCERR] |
| Flash address does not match the swap indicator address in the IFR | 2, 4 | FSTAT[ACCERR] |
| Swap initialize requested when phrase containing swap indicator (in each half of primary program flash memory) is not in the erased state | 1 | FSTAT[ACCERR] |
| Swap initialize requested when swap system is not in the uninitialized state | 1 | FSTAT[ACCERR] |
| Swap update requested when swap system is not in the ready state | 2 | FSTAT[ACCERR] |
| Swap complete requested when swap system is not in the update-erased state | 4 | FSTAT[ACCERR] |
| Swap disable requested when swap system is not in the uninitialized state | 10 | FSTAT[ACCERR] |
| An undefined swap control code is provided | - | FSTAT[ACCERR] |
| Any errors have been encountered during the swap determination and program-verify operations | 1, 2, 4, 10 | FSTAT[MGSTAT0] |
| Any brownouts were detected during the swap determination procedure | 8 | FSTAT[MGSTAT0] |

1. Returned fields will not be updated, i.e. no swap state or status reporting

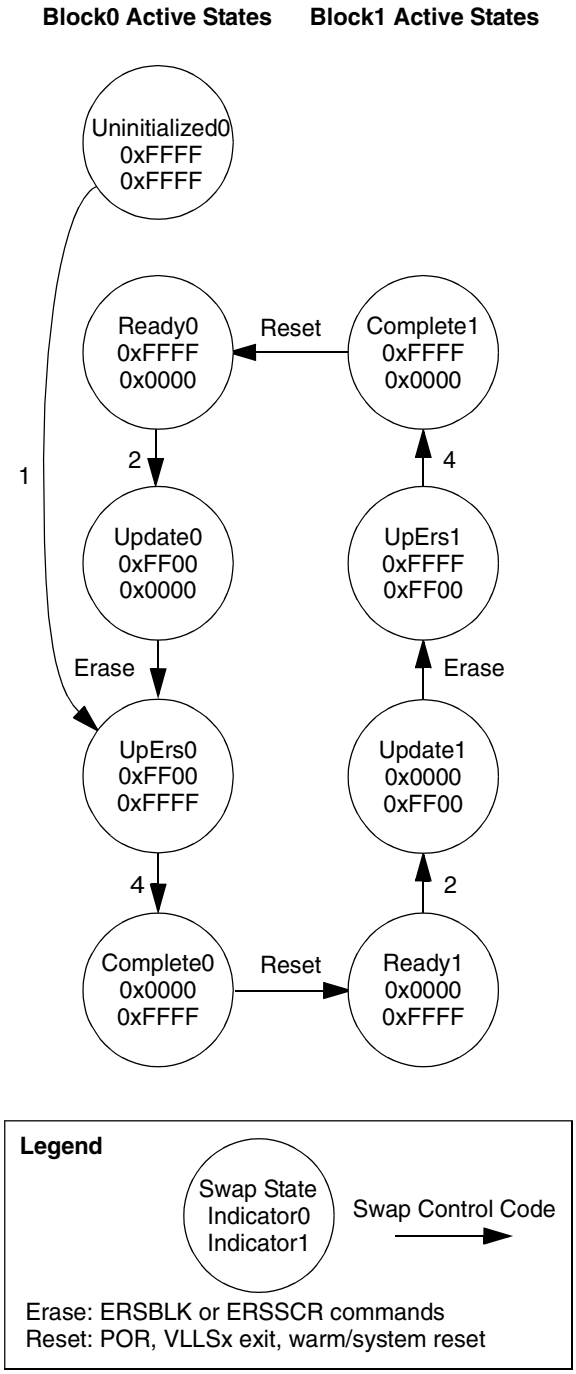



**Figure 17-9. Valid Swap State Sequencing**

## Table 17-38. Swap State Report Mapping

| Case | Swap Disable Word | Swap Enable Field[1] | Swap Indicator 0[1] | Swap Indicator 1[1] | Swap State[2] | State Code | MGSTAT 0 | Active Block |
|---|---|---|---|---|---|---|---|---|
| 1 | 0xFFFF | 0xFFFF | - | - | Uninitialized | 0 | 0 | 0 |
| 2 | 0xFFFF | 0x0000 | 0xFF00 | 0x0000 | Update | 2 | 0 | 0 |
| 3 | 0xFFFF | 0x0000 | 0xFF00- | 0xFFFF | Update-Erased | 3 | 0 | 0 |
| 4 | 0xFFFF | 0x0000 | 0x0000 | 0xFFFF[3] | Complete[4] | 4 | 0 | 0 |
| 5 | 0xFFFF | 0x0000 | 0x0000 | 0xFFFF | Ready[5] | 1 | 0 | 1 |
| 6 | 0xFFFF | 0x0000 | 0x0000 | 0xFF00 | Update | 2 | 0 | 1 |
| 7 | 0xFFFF | 0x0000 | 0xFFFF | 0xFF00 | Update-Erased | 3 | 0 | 1 |
| 8 | 0xFFFF | 0x0000 | 0xFFFF[3] | 0x0000 | Complete[4] | 4 | 0 | 1 |
| 9 | 0xFFFF | 0x0000 | 0xFFFF | 0x0000 | Ready[5] | 1 | 0 | 0 |
| 10 | 0x0000 | - | - | - | Disabled | 5 | 0 | 0 |
| 11 | 0xFFFF | 0xXXXX | - | - | Uninitialized | 0 | 1 | 0 |
| 12 | 0xFFFF | 0x0000 | 0xFFFF | 0xFFFF | Uninitialized | 0 | 1 | 0 |
| 13 | 0xFFFF | 0x0000 | 0xFFXX | 0xFFFF | Ready | 1 | 1 | 0 |
| 14 | 0xFFFF | 0x0000 | 0xFFXX | 0x0000 | Ready | 1 | 1 | 0 |
| 15[6] | 0xFFFF | 0x0000 | 0xXXXX | 0x0000 | Ready | 1 | 1 | 0 |
| 16[6] | 0xFFFF | 0x0000 | 0xFFFF | 0xFFXX | Ready | 1 | 1 | 1 |
| 17 | 0xFFFF | 0x0000 | 0x0000 | 0xFFXX | Ready | 1 | 1 | 1 |
| 18[6] | 0xFFFF | 0x0000 | 0x0000 | 0xXXXX | Ready | 1 | 1 | 1 |
| 19 | 0xFFFF | 0x0000 | 0xFF00 | 0xFFFF[7] | Update | 2 | 1 | 0 |
| 20 | 0xFFFF | 0x0000 | 0xFF00 | 0xXXXX | Update | 2 | 1 | 0 |
| 21 | 0xFFFF | 0x0000 | 0xFF(00) | 0xFFXX | Update | 2 | 1 | 0 |
| 22[6] | 0xFFFF | 0x0000 | 0x0000 | 0x0000 | Update | 2 | 1 | 0 |
| 23[6] | 0xFFFF | 0x0000 | 0xXXXX | 0xXXXX | Update | 2 | 1 | 0 |
| 24 | 0xFFFF | 0x0000 | 0xFFFF[7] | 0xFF00 | Update | 2 | 1 | 1 |
| 25 | 0xFFFF | 0x0000 | 0xXXXX | 0xFF00 | Update | 2 | 1 | 1 |
| 26 | 0xFFFF | 0x0000 | 0xFFXX | 0xFF(00) | Update | 2 | 1 | 1 |
| 27 | 0xFFFF | 0x0000 | 0xXX00 | 0xFFFF | Update-Erased | 3 | 1 | 0 |
| 28 | 0xFFFF | 0x0000 | 0xXXXX | 0xFFFF | Update-Erased | 3 | 1 | 0 |
| 29 | 0xFFFF | 0x0000 | 0xFFFF | 0xXX00 | Update-Erased | 3 | 1 | 1 |
| 30 | 0xFFFF | 0x0000 | 0xFFFF | 0xXXXX | Update-Erased | 3 | 1 | 1 |
| 31 | 0xXXXX | - | - | - | Disabled | 5 | 0 | 0 |

1. 0xXXXX, 0xFFXX, 0xXX00 indicates a non-valid value was read; 0xFF(00) indicates more 0's than other indicator (if same number of 0's, then swap system defaults to block 0 active)
2. Cases 10-29 due to brownout (abort) detected during program or erase steps related to swap
3. Must read 0xFFFF with erase verify level before transition to Complete allowed
4. No reset since successful Swap Complete execution
5. Reset after successful Swap Complete execution
6. Not a valid case
7. Fails to read 0xFFFF at erase verify level

#### 17.4.11.14.1 Swap state determination

During the reset sequence, the state of the swap system is determined by evaluating the IFR Swap Field in the program flash swap IFR and the swap indicators found within the double-phrase containing the relative swap indicator address in each half of the program flash memory.

**Table 17-39. Program Flash Swap IFR Fields**

| Address Range | Size (Bytes) | Field Description |
|---|---|---|
| 0x000 – 0x001 | 2 | Swap Indicator Address |
| 0x002 – 0x003 | 2 | Swap Enable Word |
| 0x004 – 0x009 | 6 | Reserved |
| 0x00A – 0x00B | 2 | Swap Disable Word |
| 0x00C – 0x3FF | 1012 | Reserved |

### 17.4.11.15 Erase All Blocks Unsecure Command

The Erase All Blocks Unsecure operation erases all flash memory including the erasable IFR, initializes the programming acceleration RAM, programs the security related index fields in the Flash Configuration Field to the unsecure state, and releases MCU security.

**Table 17-40. Erase All Blocks Unsecure Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x49 (ERSALLU) |

Upon clearing CCIF to launch the Erase All Blocks Unsecure command, the flash module erases all flash memory, the erasable IFR space,and the programming acceleration RAM, then verifies that all are erased. If the flash module verifies that all flash memories and the programming acceleration RAM were properly erased, access control is disabled and security is released by setting the FSEC[SEC] field to the unsecure state, the security related index fields (see Flash Configuration Field description) are programmed to the unsecure state by the Erase All Blocks Unsecure command, and the FCNFG[RAMRDY] bit is set. The swap indicator address in the primary program flash blocks are not implicitly protected from the erase operation. If the erase or program verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Erase All Blocks Unsecure operation completes.

Access control determined by the contents of the FXACC and FXACCS registers will not block execution of the Erase All Blocks Unsecure command. The contents of the FXACC, FSACC, FXACCS and FSACCS registers will not be impacted by the

execution of the Erase All Blocks Unsecure command but the associated FXACC, FSACC, FXACCS and FSACCS index fields will be erased. After completion of the Erase All Blocks Unsecure command, access control is disabled until the next reset of the flash module or the Read 1s All Blocks command is executed and fails (FSTAT[MGSTAT0] is set).

**CAUTION**

The Erase All Blocks Unsecure operation will not react to the early indicator for radio activity. Therefore, the Erase All Blocks Unsecure command must not be launched if there is a concern about inadequate power available to support both radio activity and the Erase All Blocks Unsecure operation.

**Table 17-41. Erase All Blocks Unsecure Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Any flash block is in standby mode | FSTAT[ACCERR] |
| Any errors have been encountered during erase or program verify operations | FSTAT[MGSTAT0] |

### 17.4.11.16 Read 1s All Execute-only Segments Command

The Read 1s All Execute-only Segments command checks if the program flash execute-only segments defined by the FXACC and FXACCS registers have been erased to the specified read margin level, if applicable, and releases flash access control if the readout passes, i.e. all data reads as '1'.

**Table 17-42. Read 1s All Execute-only Segments Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x4A (RD1XA) |
| 1 | Read-1 Margin Choice |

Upon clearing CCIF to launch the Read 1s All Execute-only Segments command, the flash memory module:

- sets the read margin for 1s according to Table 17-43,
- verifies the contents of the flash execute-only segments are in the erased state.

If the flash memory module confirms that these segments are erased, flash access control is disabled until the next reset or, after programming any of the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the

FSTAT[MGSTAT0] bit set. If the read fails, i.e. all segments are not in the fully erased state, the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Read 1s All Execute-only Segments operation completes.

**CAUTION**

The Read 1s All Execute-only Segments operation will not react to the early indicator for radio activity. Therefore, the Read 1s All Execute-only Segments command must not be launched if there is a concern about inadequate power available to support both radio activity and the Read 1s All Execute-only Segments operation.

**Table 17-43. Margin Level Choices for Read 1s All Execute-only Segments**

| Read Margin Choice | Margin Level Description |
|---|---|
| 0x00 | Use the 'normal' read level for 1s |
| 0x01 | Apply the 'User' margin to the normal read-1 level |
| 0x02 | Apply the 'Factory' margin to the normal read-1 level |

**Table 17-44. Read 1s All Execute-only Segments Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| An invalid margin choice is specified | FSTAT[ACCERR] |
| Any program flash block is in standby mode | FSTAT[ACCERR] |
| Read-1s fails | FSTAT[MGSTAT0] |

## 17.4.11.17 Erase All Execute-only Segments Command

The Erase All Execute-only Segments operation erases all program flash execute-only segments defined by the FXACC and FXACCS registers, verifies all segments are erased, and releases flash access control.

**Table 17-45. Erase All Execute-only Segments Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x4B (ERSXA) |

Upon clearing CCIF to launch the Erase All Execute-only Segments command, the flash memory module erases all program flash execute-only segments, then verifies that all segments are erased. If the flash memory module verifies that all segments were properly erased, flash access control is disabled until the next reset or, after programming any of

the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the FSTAT[MGSTAT0] bit set. The Erase All Execute-only Segments command aborts if any XA or XA_S controlled segment is protected. If the swap system is enabled, the swap indicator address in each primary program flash block is implicitly protected from erase unless the swap system is in the UPDATE or UPDATE-ERASED state and the flash segment containing the swap indicator address being erased is in the non-active block. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag sets after the Erase All Execute-only Segments operation completes.

Access control determined by the contents of the FXACC and FXACCS registers will not block execution of the Erase All Execute-only Segments command. The contents of the FXACC and FXACCS registers will not be impacted by the execution of the Erase All Execute-only Segments command.

**CAUTION**

The Erase All Execute-only Segments operation will not react to the early indicator for radio activity. Therefore, the Erase All Execute-only Segments command must not be launched if there is a concern about inadequate power available to support both radio activity and the Erase All Execute-only Segments operation.

**Table 17-46. Erase All Execute-only Segments Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Any program flash block is in standby mode | FSTAT[ACCERR] |
| Any XA or XA_S controlled segment is protected | FSTAT[FPVIOL] |
| Any errors have been encountered during the verify operation | FSTAT[MGSTAT0] |

## 17.4.11.18 Set RAM Function command

The Set RAM Function command sets the function of the programming acceleration RAM.

**Table 17-47. Set RAM Function Command FCCOB Requirements**

| FCCOB Number | FCCOB Contents [7:0] |
|---|---|
| 0 | 0x81 (SETRAM) |
| 1 | RAM Function Control Code<br>(see Table 17-48) |

**Table 17-48. Set RAM Function Control**

| RAM Function Control Code | Action |
|---|---|
| 0xCC | Make RAM available for CRC generation:<br>• Clear the FCNFG[RAMRDY] and FCNFG[CRCRDY] flags<br>• Set the FCNFG[CRCRDY] flag<br>• RAM available for Generate CRC command after FSTAT[CCIF] sets |
| 0xFF | Make RAM available for normal use:<br>• Clear the FCNFG[RAMRDY] and FCNFG[CRCRDY] flags<br>• Write a background of ones to all RAM locations<br>• Set the FCNFG[RAMRDY] flag |

Upon clearing CCIF to launch the Set RAM Function command, the CRCRDY and RAMRDY flags are cleared and the programming acceleration RAM is set to function based on the control code entered:

- 0xCC - When making the programming acceleration RAM available for CRC generation, the flash module blocks access to the programming acceleration RAM and sets the FCNFG[CRCRDY] flag. Once the operation has completed, the Generate CRC command can be launched to produce a CRC signature on a flash block while leaving that same flash block accessible for read access. While the CRCRDY flag is set, access to the programming acceleration RAM is blocked until the Set RAM Function command is run with a control code of 0xFF.
- 0xFF - When making the programming acceleration RAM available for normal use, the flash module overwrites the contents of the entire programming acceleration RAM with a background pattern of all ones, and sets the FCNFG[RAMRDY] flag. Normal read and write accesses to the programming acceleration RAM are now available. When large sections of flash memory need to be programmed, e.g. during factory programming, the programming acceleration RAM can be used as the Section Program Buffer for the Program Section command (see Program Section command).

The CCIF flag sets after the Set RAM Function operation completes.

**Table 17-49. Set RAM Function Command Error Handling**

| Error Condition | Error Bit |
|---|---|
| Command not available in current mode/security | FSTAT[ACCERR] |
| Primary Program flash block 0 is in standby mode | FSTAT[ACCERR] |
| Function Control Code is not defined | FSTAT[ACCERR] |

## 17.4.12 Security

The flash module provides security information to the MCU based on contents of the FSEC security register. The MCU then limits access to flash resources as defined in the device's Chip Configuration details. During the reset sequence, the flash module initializes the FSEC register using data read from the security related index fields located in the Flash Configuration Field (see Flash Configuration Field description).

The following fields are available in the FSEC register. Details of the settings are described in the FSEC register description.

**Table 17-50. FSEC fields**

| FSEC field | Description |
|---|---|
| KEYEN | Backdoor Key Access |
| MEEN | Mass Erase Capability |
| FSLACC | Factory Security Level Access |
| SEC | MCU security |

**NOTE**

Flash security features are discussed further in AN4507: Using the Kinetis Security and Flash Protection Features. Some features described in the application note may not be available on this device.

### 17.4.12.1 Changing the Security State

The security state out of reset can be permanently changed by programming the security related index fields located in the flash configuration field. This assumes that you are starting from a mode where the necessary flash erase and program commands are available. If the flash security related index fields are successfully programmed, the new FSEC value takes effect after the next MCU reset.

#### 17.4.12.1.1 Unsecuring the MCU Using Backdoor Key Access

The MCU can be unsecured by using the backdoor key access feature which requires knowledge of the contents of the 8-byte backdoor key value stored in the Flash Configuration Field (see Flash Configuration Field description). If the FSEC[KEYEN] bits are in the enabled state, the Verify Backdoor Access Key command (see Verify Backdoor Access Key command) can be run which allows the user to present prospective keys for comparison to the stored keys. If the keys match, the FSEC[SEC] bits are changed to unsecure the MCU. The entire 8-byte key cannot be all 0s or all 1s, i.e.

0x0000_0000_0000_0000 and 0xFFFF_FFFF_FFFF_FFFF are not accepted by the Verify Backdoor Access Key command as valid comparison values. While the Verify Backdoor Access Key command is active, program flash memory is not available for read access and returns invalid data.

The user code stored in the program flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN bits are in the enabled state, the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Verify Backdoor Access Key command
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the FSEC[SEC] bits are forced to the unsecure state

An illegal key provided to the Verify Backdoor Access Key command prohibits future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method to re-enable the Verify Backdoor Access Key command when a comparison fails.

After the backdoor keys have been correctly matched, the MCU is unsecured by changing the FSEC[SEC] bits. A successful execution of the Verify Backdoor Access Key command changes the security in the FSEC register only. It does not alter the security byte or the keys stored in the Flash Configuration Field (Flash Configuration Field description). After the next reset of the MCU, the security state of the flash module reverts back based on the contents of the security related index fields located in the Flash Configuration Field. The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the program flash protection registers.

If the backdoor keys successfully match, the unsecured MCU has full control of the contents of the Flash Configuration Field. The MCU may mass erase the flash memory to erase the IFR containing the Flash Configuration Field and reprogram the flash security fields including the backdoor keys.

## 17.4.13 Reset Sequence

On each system reset the flash module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, FSEC, FXACC, FSACC, FXACCS, FSACCS, and FACNFG registers and the FCNFG[SWAP, CRCRDY, RAMRDY ] bits.

CCIF is cleared throughout the reset sequence. The flash module holds off all CPU access during the reset sequence. Flash reads are possible when the hold is removed. Completion of the reset sequence is marked by setting CCIF which enables flash user commands.

If a reset occurs while any flash command is in progress, that command is immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed. Commands and operations do not automatically resume after exiting reset.

# Chapter 18
# Flash Memory Controller (FMC)

## 18.1 Chip-specific FMC information

### 18.1.1 FMC buffers control

The FMC's buffer are controlled by:

- CM4F: MSCM_OCMDR0
- CM0+: MSCM_OCMDR1

### 18.1.2 Flash Access Control (FAC) Function

The Flash Access Control (FAC) is a configurable memory protection scheme optimized to allow end users to use software libraries while offering programmable restrictions to these libraries. The flash memory is divided into *equal size segments* that provide protection to proprietary software libraries. The protection of these segments is controlled: the FAC provides a cycle-by-cycle evaluation of the access rights for each transaction routed to the on-chip flash memory. Two levels of vendors can add their proprietary software to a device; FAC protection of segments for each level are defined once, using the PGMINDEX command.

Flash access control aligns to the 3 privilege levels supported by ARM Cortex-M family products:

- Most secure state is supervisor/privileged secure: allows execute-only and provides supervisor-only access control.
- Mid-level state is execute-only.
- Unsecure state is where no access control states are set.

Features:

- Lightweight access control logic for on-chip flash memory

- Flash address space divided into (64+16) equal-sized segments (segment size is defined as 16 KB)
- Separate control bits for supervisor-only access and execute-only access per segment
- Access control evaluated on each bus cycle routed to the flash
- Access violation errors terminate the bus cycle and return zeroes for read data
- Programming model allows 2 levels of protected segments

### 18.1.2.1 FAC functional description

The Flash Management Unit (FMU) includes non-volatile configuration information that is retrieved during reset and sent to the platform to control access to the flash array during normal operation.

There are 64-bit XACCP and 64-bit SACCP for primary program flash, 16-bit XACCS and 16-bit SACCS for secondary program flash to support access control features. These NVM locations are summarized in the table below.

**Table 18-1. NVM Locations**

| NVM location | Description | |
|---|---|---|
| NVSACCP | 64-bit location is loaded during reset into the *x*_SACC register to provide access configuration for CM4F. | Segment-wise control for supervisor-only access vs. supervisor and user access |
| NVXACCP | 64-bit location is loaded during reset into the *x*_XACC register to provide access configuration for CM4F. | Segment-wise control for execute-only vs. data and execute |
| NVSACCS | 16-bit location is loaded during reset into the *x*_SACC register to provide access configuration for CM0+. | Segment-wise control for supervisor-only access vs. supervisor and user access |
| NVXACCS | 16-bit location is loaded during reset into the *x*_XACC register to provide access configuration for CM0+. | Segment-wise control for execute-only vs. data and execute |

Each of these NVM locations is programmable through a Program Index command.

The flash is released from reset early while the core continues to be held in reset. The FMU captures the NVM access control information in internal registers. The FMU ANDs the multiple execute-only fields to create a single execute-only field. This execute-only field driven to the platform is static prior to the core being released from reset. The supervisor-only field is handled in the same manner.

The FMU includes the FAC registers that provide control access to the flash address space. During the address phase of every attempted flash transfer, the supervisor access (SA*n*) and execute access (XA*n*) bits are examined to either allow or deny access. If access is denied, then the access is aborted and terminates with a bus error; the read data is also zeroed.

The next table shows segment assignments relative to the flash location.

**Table 18-2. Flash Protection Ranges**

| SA*n* and XA*n* Bit | Protected Segment Address Range | Segment Size (Fraction of total Flash) |
|---|---|---|
| **64 Segment Encodings** | | |
| 0 | 0x0_0000_0000 – (Flash_size/64)-1 | 1/64 |
| 1 | (Flash_size/64) – 2*(Flash_size/64)-1 | 1/64 |
| ..... | | |
| 63 | 62*(Flash_size/64)-1 – 63*(Flash_size/64) | 1/64 |
| **16 Segment Encodings** | | |
| 0 | 0x0_0000_0000 – (Flash_size/16)-1 | 1/16 |
| 1 | (Flash_size/16) – 2*(Flash_size/16)-1 | 1/16 |
| ..... | | |
| 15 | 14*(Flash_size/16)-1 – 15*(Flash_size/16) | 1/16 |

Individual segments within the flash memory can be independently protected from user access and data access. Protection is controlled by the individual bits within the *x*_SACC and *x*_XACC registers, as shown in the next figure.

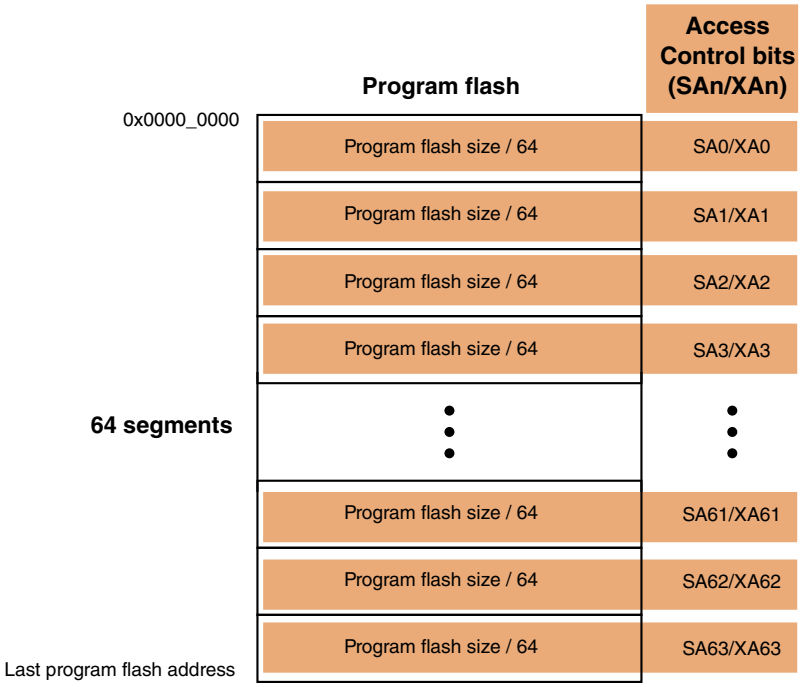



**Figure 18-1. Program flash protection (64 segments)**

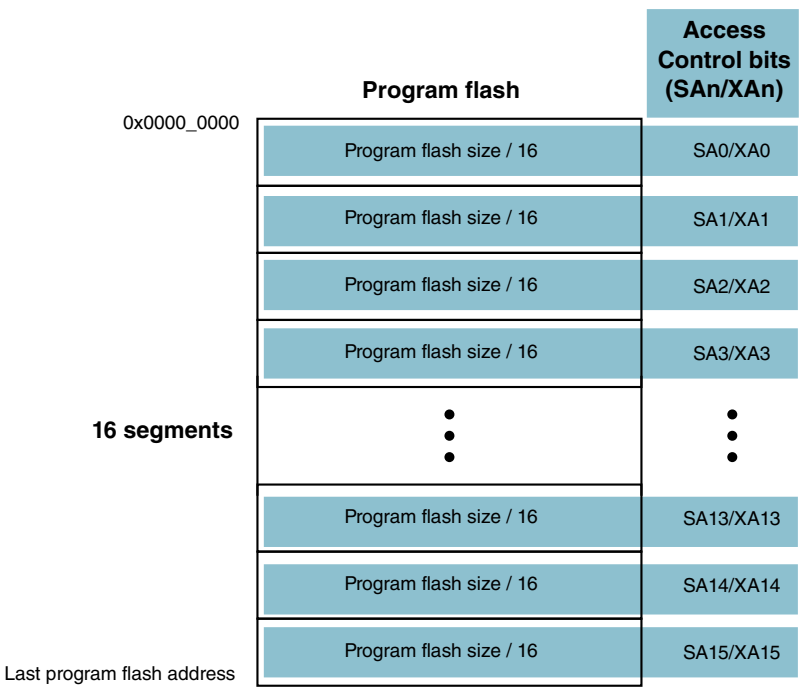



**Figure 18-2. Program flash protection (16 segments)**

#### 18.1.2.1.1 Interface Signals

**Table 18-3. Interface Signals**

| Signal | Width | From | To | Description |
|---|---|---|---|---|
| xacc0 | 64 | FMU | Platform | Direct xacc (execute-only access control) register for CM4F core |
| sacc0 | 64 | FMU | Platform | Direct sacc (supervisor access control) register for CM4F core |
| xacc1 | 16 | FMU | Platform | Direct xacc (execute-only access control) register for CM0+ core |
| sacc1 | 16 | FMU | Platform | Direct sacc (supervisor access control) register for CM0+ core |
| numsg0 | 8 | FMU | Platform | NUMSG bit field - Binary encoded number of segments<br>0x40 for 64 segments<br>0x10 for 16 segments |

#### 18.1.2.1.2 Flash Command Impact

| | |
|---|---|
| **Program Longword/Phrase/ Section** | If the targeted flash location is in an execute-only protected segment, then these program commands are not allowed unless a Read 1s All Blocks command is executed and returns with a pass code (which means the part has been fully erased). After the Read 1s All Blocks command is executed with a pass code returned, then the protected segment is open to program commands. To close off programmability to execute-only spaces once again, the device must be reset or a Read 1s All Blocks command is executed with a fail result. Attempts to program in a protected segment *when not open to program commands* causes a Protection Violation flag. |
| **PGMCHK** | The FMU will not execute the PGMCHK command on a segment that has been configured as execute-only. The Flash Protection Violation flag is set if an attempt is made to execute PGMCHK command on an execute-only address. |
| **Erase Flash Sector** | If the targeted flash sector is in an execute-only protected segment, then the Erase Flash Sector command is not allowed, and sets the Protection Violation flag. The only means of erasing protected space is by an Erase All operation. |
| **ERSALL** | The Erase All Blocks command is not affected by Access Control. An Erase All Blocks command will erase any libraries that have been programmed in any execute-only segment. |
| **SWAP** | A new control has been added to the SWAP command to disable the SWAP feature.<br>The IFR SWAP Field and the SWAP indicator are erased during the Erase All Blocks command operation, resulting in the SWAP system being uninitialized. The SWAP command must be run with the initialization code to set the SWAP indicator address and initialize the SWAP system.<br>After being disabled, SWAP cannot be enabled without doing an Erase All. An Erase All erases preloaded code and libraries in the flash array and resets the SWAP system back to uninitialized. If SWAP is intended to be disabled as part of the access control protection, then the disabled setting must be restored after an Erase All Blocks operation. |
| **ERSXA** | The target regions of the command is controlled by FAC. For ERSXA, if a flash sector is in an execute-only protected segment and it is not protected by flash protection (FPROT), it will be erased. |
| **RD1XA** | The target regions of the command is controlled by FAC. For RD1XA, if a flash sector is in an execute-only protected segment, it will be erase verified. |

#### 18.1.2.1.3 Core Platform Impact

| | |
|---|---|
| **Platform core caches (Flash and LMEM caches)** | If any segment is marked as *execute-only*, then the caches are hidden from the user. The tag is read-only and cannot be written, and the data caches cannot be read or written. Writes to the tag and data arrays are ignored, and reads of the data array return 0's. This will impact debug breakpoints. See the debug section for details. |
| **Debug** | The debugger is a non-processor bus master and cannot step, trace or break in execute-only regions. In supervisor-only mode, the debugger is restricted from changing modes. Debug accesses to any segment of flash space marked as execute-only also terminate with a bus error. |
| **PC-relative addressing** | Restrictions are placed on software for PC relative addressing, because hardware cannot determine if PC relative data references are crossing segment boundaries.<br>• If ifetch is executing in a protected segment, then data references will be allowed.<br>• Hardware cannot track speculative ifetches across boundaries. |
| **Interrupts** | If function calls are used to move into an execute-only segment, then this can be tracked by hardware when typical software controls are used (i.e., saving registers and states before executing new code). |
| **Reset Vector** | In the ARM core, the reset vector fetch is supervisor data, which poses issues if the reset vector is located in a segment marked execute-only. Additional logic has been implemented to allow supervisor data fetches to execute-only spaces, after reset until the first valid instruction fetch. After the first valid instruction fetch, the FAC logic follows normal checks. |

#### 18.1.2.1.4 Software Impact

The hardware cannot see all states of the ARM core and cannot track the software flow, and may require software restrictions to work with the hardware for a robust solution.

- **Any segment marked as execute-only can see all code in the system.** This means that one execute-only segment can read the execute-only code in another segment. Therefore, if we at the factory are sending pre-loaded code to another vendor, then that vendor will have access to our factory code. NDAs and legal agreements might help deal with this issue.
- **For single pre-loads** (for example, if we at the factory are pre-loading for a general purpose (GP) market or if a vendor with a blank part is pre-loading their proprietary code), then both levels of access control must be programmed, to protect the pre-loaded code.
- **If any portion of a protected segment is not used by pre-loaded code**, then it (the portion of a protected segment that is not used by pre-loaded code) should be programmed with NOPs, to prevent additional code from being programmed in that segment by hackers.

#### 18.1.2.1.5 Access Check Evaluation

The flash controller FAC provides a cycle-by-cycle evaluation of the access rights for each data transaction routed to the on-chip flash memory.

The entire flash storage capacity is partitioned into equal sized segments. Two registers include a supervisor-only access control indicator and a execute-only access control indicator for each segment.

The FAC logic performs the required access control evaluation using the reference address and a 2-bit attribute (or "protection" field) as inputs from the bus cycle plus the contents of the programming model registers.

The following code example illustrates C code for FAC evaluation:

```
unsigned long long sacc;  // supervisor-only map
unsigned long long xacc;  // execute-only map
unsigned int seg_size;    // 8-bit segment size
unsigned int fac_error;

fac_evaluation (addr, prot)
  unsigned int addr;  // access address
  unsigned int hprot; // encoded 2-bit "protection" field {supv, data}
{
  unsigned int sacc_flag; // sacc flag for this segment
  unsigned int xacc_flag; // xacc flag for this segment
  unsigned int i;         // segment index

  i         = (addr >> (8 + seg_size & 0x0f)) & 0x3f; // form 6-bit segment index
  sacc_flag = (sacc >> i) & 1;              // extract sacc bit for this segment
  xacc_flag = (xacc >> i) & 1;              // extract xacc bit for this segment

  // create a 4-tuple concatenating the 2-bit protection field + {sacc, xacc} flags

  switch ((hprot & 3) << 2 | (sacc_flag << 1) | xacc_flag) {
  // all these combinations are allowed accesses
    case 0x2: // {user, ifetch} && {supv+user, ifetch-only}
    case 0x3: // {user, ifetch} && {supv+user, ifetch+data}
    case 0x7: // {user, data}   && {supv+user, ifetch+data}
    case 0x8: // {supv, ifetch} && {supv-only, ifetch-only}
    case 0x9: // {supv, ifetch} && {supv-only, ifetch+data}
    case 0xa: // {supv, ifetch} && {supv+user, ifetch-only}
    case 0xb: // {supv, ifetch} && {supv+user, ifetch+data}
    case 0xd: // {supv, data}   && {supv-only, ifetch+data}
    case 0xf: // {supv, data}   && {supv+user, ifetch+data}
      fac_error = 00;
      break;

  // all these combinations are unallowed, that is, errored accesses
    case 0x0: // {user, ifetch} && {supv-only, ifetch-only}
    case 0x1: // {user, ifetch} && {supv-only, ifetch+data}
    case 0x4: // {user, data}   && {supv-only, ifetch-only}
    case 0x5: // {user, data}   && {supv-only, ifetch+data}
    case 0x6: // {user, data}   && {supv+user, ifetch-only}
    case 0xc: // {supv, data}   && {supv-only, ifetch-only}
    case 0xe: // {supv, data}   && {supv+user, ifetch-only}
      fac_error = 1;
      break;

   } // switch()

} // fac_evaluation()
```

## 18.2 Introduction

The Flash Memory Controller (FMC) is a memory interface and acceleration unit, which provides:

- an interface between the device and the nonvolatile memory.
- buffers that can accelerate flash memory transfers.

### 18.2.1 Overview

The Flash Memory Controller manages the interface between the device and the flash memory. The FMC receives status information describing the configuration of the memory and uses this information to ensure a proper interface. The next table shows the supported read/write operations.

| Flash memory type | Read | Write |
|---|---|---|
| Program flash memory | 8-bit, 16-bit, and 32-bit reads | —[1] |

1. A write operation to program flash memory results in a bus error.

- The FMC's input bus can actually operate faster than the flash memory.
- The FMC-to-flash interface has flow control, and can add wait states as needed, for input bus reads that need flash accesses.
- The FMC also contains various configurable buffers that hold recent flash accesses. If an input bus read hits a valid buffer, that access will finish with no wait states.

### 18.2.2 Features

- Interface between the device and the flash memory:
  - The FMC's input bus supports 8-bit, 16-bit, and 32-bit read operations to flash memory.
  - The FMC's flash memory interface fetches a {32, 64, 128, 256}-bit flash page.
  - For input read requests, the FMC fetches a flash page with the desired read data from flash memory.
- Acceleration of data transfer from flash memory to the device:
  - A flash-page-sized buffer that holds the current flash page fetched due to a FMC read request. Subsequent FMC read requests *that hit in the current buffer* return data with no wait states.

- A flash-page-sized prefetch speculation buffer with controls for prefetching on instructions and/or data reads. When prefetching is enabled, idle FMC-to-flash interface cycles are used to fetch the next sequential flash page and hold it in the prefetch buffer. Subsequent FMC read requests *that hit in the speculation buffer* will return data with no wait states.
- Input controls:
  - to disable data type speculation
  - to disable all speculation
  - to invalidate the current and speculation buffers

## 18.3 Modes of operation

The FMC only operates when a bus master accesses the flash memory.

For any device power mode where the flash memory cannot be accessed, the FMC is disabled.

## 18.4 External signal description

The FMC has no external signals.

## 18.5 Functional description

The FMC is a flash interface and acceleration unit, with flexible buffers for user configuration.

- The FMC's input bus can operate faster than the flash memory.
- The FMC-to-flash interface has flow control to add wait states as needed (for input bus reads that need flash accesses).
- The FMC also contains various configurable buffers that hold recent flash accesses. If an input bus read hits a valid buffer, then that access will complete with no wait states.

### 18.5.1 Default configuration

After system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory:

- For all banks:

- The current and speculation buffers are cleared by reset.
- Prefetch support for data and instructions is enabled.

## 18.5.2 Configuration options

The default configuration provides a high degree of flash acceleration, however, advanced users may want to customize the FMC buffer configurations, to maximize throughput for their use cases. When reconfiguring the FMC for custom use cases, do not program the FMC's control registers while the flash memory is being accessed. Instead, change the control registers with a routine executing from RAM in supervisor mode.

The FMC's buffering controls allow the tuning of resources to suit specific application requirements. The buffer are each controlled individually. The controls enable buffering and prefetching per access type (instruction fetch or data reference).

As an application example: if both instruction fetches and data references are accessing flash memory, then control is available to send instruction fetches, data references, or both to the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access.

## 18.5.3 Wait states

Because the core, crossbar switch, and bus masters can be clocked at a higher frequency than the flash clock, flash memory accesses that do not hit in the speculation buffer usually require wait states.

All wait states and synchronization delays are handled automatically by the Flash Memory Controller. No direct user configuration is required (or even allowed) to set up the flash wait states.

## 18.5.4 Speculative reads

The FMC has a single buffer that reads ahead to the next page in the flash memory if there is an idle cycle. Speculative prefetching is programmable for instruction and/or data accesses. Because many code accesses are sequential, using the speculative prefetch buffer improves performance in most cases.

When speculative reads are enabled, the FMC immediately requests the next sequential page address after a read completes. By requesting the next page immediately, speculative reads can help to reduce or even eliminate wait states when accessing sequential code and/or data.

## 18.6 Initialization and application information

The FMC does not require user initialization. Flash acceleration features are enabled by default.

The FMC has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly. As a result, if an application is executing flash memory commands, the FMC's current buffer and speculation buffer might need to be disabled and/or flushed to prevent the possibility of returning stale data.

# Chapter 19
# Flash security

## 19.1 Introduction

This device implements security based on the mode selected from the flash module. The following sections provide an overview of flash security and details the effects of security on non-flash modules.

## 19.2 Flash Security

The flash module provides security information to the MCU based on the state held by the FTFE_FSEC[SEC] bits. The MCU, in turn, confirms the security request and limits access to flash resources. During reset, the flash module initializes the FSEC register using data read from the security related index fields located in IFR space.

**NOTE**

The security features apply only to external accesses via debug. CPU accesses to the flash are not affected by the status of FSEC.

In the unsecured state all flash commands are available to the programming interfaces (JTAG), as well as user code execution of Flash Controller commands. When the flash is secured (FSEC[SEC] = 00, 01, or 11), programmer interfaces are only allowed to launch mass erase operations and have no access to memory locations.

Further information regarding the flash security options and enabling/disabling flash security is available in the Flash Memory Module.

## 19.3 Security Interactions with other Modules

The flash security settings are used by the SoC to determine what resources are available. The following sections describe the interactions between modules and the flash security settings or the impact that the flash security has on non-flash modules.

### 19.3.1 Security Interactions with Debug

When flash security is active the JTAG port cannot access the memory resources of the MCU. Boundary scan chain operations work, but debugging capabilities are disabled so that the debug port cannot read flash contents.

Although most debug functions are disabled, the debugger can write to the System Reset Request bit and Flash Mass Erase in Progress bit in the MDM-AP Control register to trigger a mass erase (Erase All Blocks Unsecure) command. A mass erase via the debugger is allowed even when some memory locations are protected.

When mass erase is disabled, mass erase via the debugger is blocked.

# Chapter 20
# External Bus Interface (FlexBus)

## 20.1 Chip-specific FlexBus information

### 20.1.1 FlexBus clocking

The system provides a dedicated clock source (SCG CLKOUT) to the FlexBus module's external CLKOUT. The SCG CLKOUT is output from SCG_CLKOUTCNFG mux. In order to be used as FB_CLKOUT, SCG_CLKOUTCFNG[CLKOUTSEL] must be set to 0. And under this configuation, the clock is derived from divider SCG RCCR[DIVEXT] of PLAT_CLK. See Clock Architecture and System Clock Generator (SCG) for more details.

### 20.1.2 FlexBus signal multiplexing

The multiplexing of the FlexBus address and data signals is controlled by the port control module. However, the multiplexing of some of the FlexBus control signals are controlled by the port control and FlexBus modules. The port control module registers control whether the FlexBus or another module signals are available on the external pin, while the FlexBus's CSPMCR register configures which FlexBus signals are available from the modules. The control signals are grouped as illustrated:

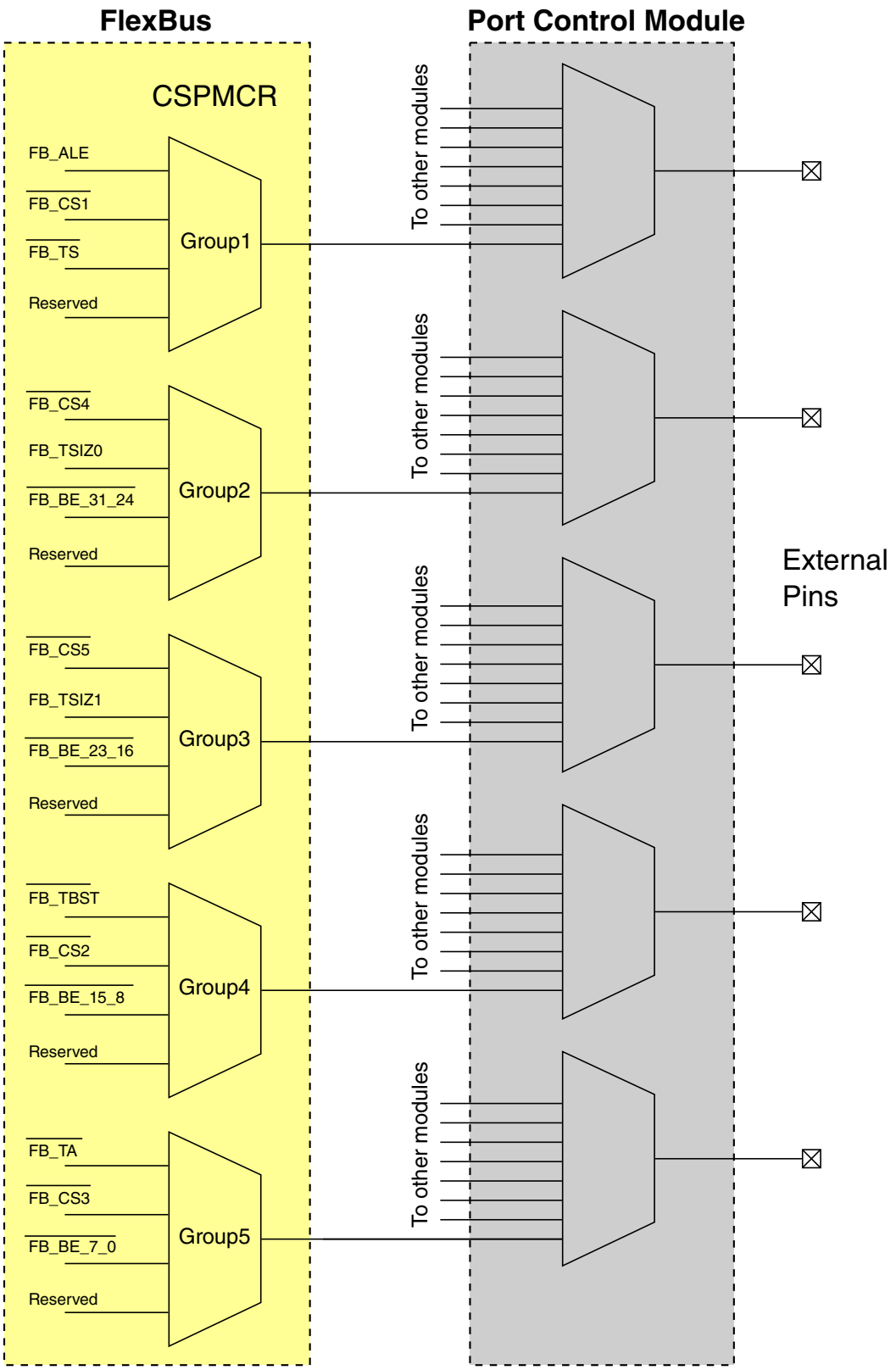



**Figure 20-1. FlexBus control signal multiplexing**

Therefore, use the CSPMCR and port control registers to configure which control signal is available on the external pin. All control signals, except for $\overline{\text{FB_TA}}$, are assigned to the ALT5 function in the port control module. The $\overline{\text{FB_TA}}$ is an input signal, and is assigned to the ALT6 function.

### 20.1.3 FlexBus CSCR0 reset value

On this device the CSCR0 resets to 0x003F_FC00. Configure this register as needed before performing any FlexBus access.

### 20.1.4 FlexBus Security

When security is enabled on the device, FlexBus accesses may be restricted by configuring SIM_CHIPCTRL[FBSL]. See System Integration Module (SIM) for details.

### 20.1.5 FlexBus line transfers

Line transfers are not possible from the ARM Cortex-M4 core. Ignore any references to line transfers in the FlexBus chapter.

## 20.2 Introduction

The FlexBus multifunction external bus interface controller:

- Provides memory expansion and provides connection to external peripherals with a parallel bus
- Provides connections to external peripherals with a parallel bus, and is often used to expand memory
- Can be directly connected to asynchronous or synchronous slave-only devices, using little or no additional circuitry:
  - External ROMs
  - Flash memories
  - Programmable logic devices
  - Other simple target (slave) devices

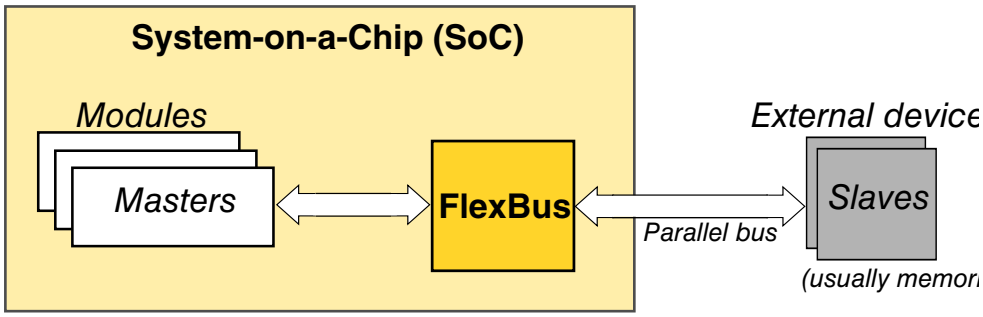



**Figure 20-2. Flexbus module in a System-on-a-Chip (SoC)**

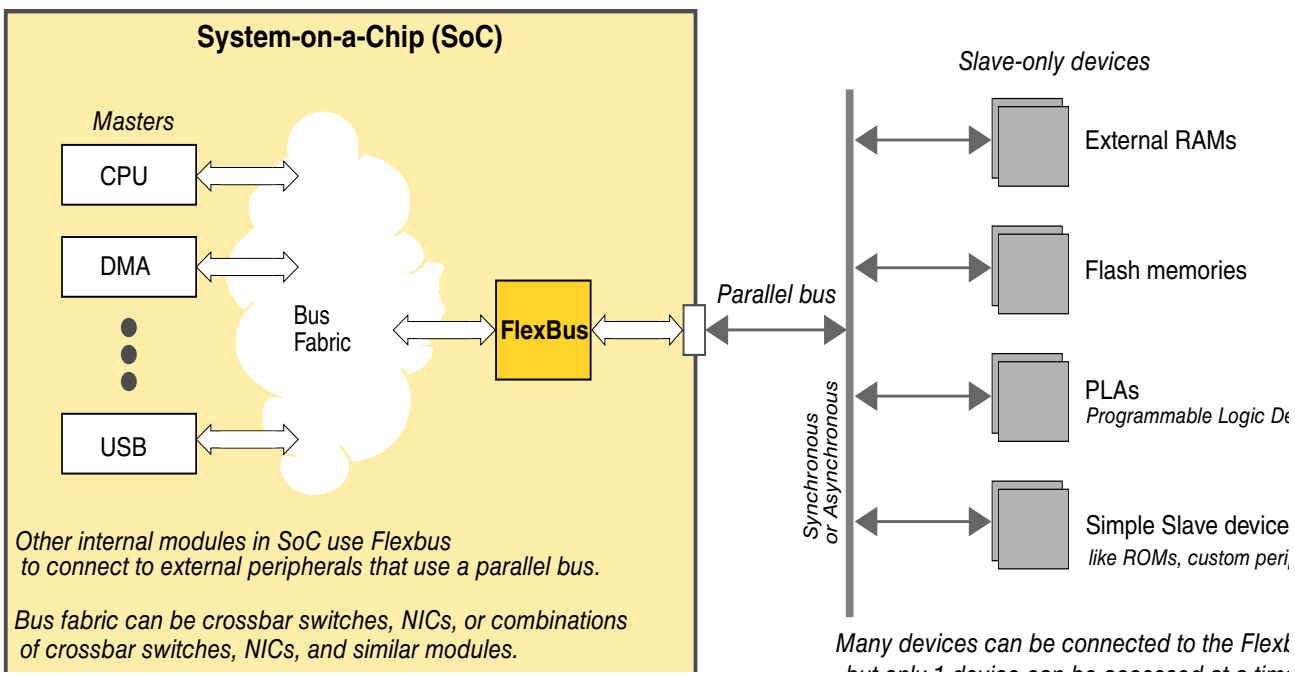



**Figure 20-3. Flexbus module in SoC details**

FlexBus features:

- 6 independent, user-programmable chip-select signals (active low) (FB_CS 5 – FB_CS0)
- 8-bit, 16-bit, and 32-bit transfers
- Programmable burst and burst-inhibited transfers selectable for each chip-select and transfer direction
- Programmable address-setup time with reference to the assertion of a chip-select
- Programmable address-hold time with reference to the deassertion of a chip-select and transfer direction
- Extended transfer start/address latch enable option to assist with glueless connections to synchronous and asynchronous memory devices

## 20.3 Signal descriptions

This table describes the external signals involved in data-transfer operations.

**NOTE**

Not all of the following signals may be available on a particular device. See the Chip Configuration details for information on which signals are available.

**NOTE**

Throughout this document, active low signals are indicated by one of 3 methods:

- "_b" appended to the end of a signal name
- a black line (called an "overbar") over the signal name
- a phrase "active low" next to the signal name

**Table 20-1. FlexBus signal descriptions**

| Signal | I/O | Function |
|---|---|---|
| FB_A31 - FB_A0 (nonmultiplexed address bus) | O | Address Bus<br>When FlexBus is used in a nonmultiplexed configuration, this is the address bus.<br>When FlexBus is used in a multiplexed configuration, this bus is not used. |
| FB_D31 – FB_D0 (nonmultiplexed data bus)<br>FB_AD31 - FB_AD0 (multiplexed address/ data bus) | I/O | When FlexBus is used in a nonmultiplexed configuration, this is the data bus, FB_D.<br>When FlexBus is used in a multiplexed configuration, this is the address and data bus, FB_AD.<br>The number of byte lanes carrying the data is determined by the port size associated with the matching chip-select.<br>The full 32-bit address is driven on the first clock of a bus cycle (address phase). After the first clock, the data is driven on the bus (data phase). During the data phase, the address is driven on the pins not used for data. For example, in 16-bit mode, the lower address is driven on FB_AD15–FB_AD0, and in 8-bit mode, the lower address is driven on FB_AD23–FB_AD0. |
| FB_CS5 – FB_CS0 | O | General Purpose Chip-Selects (active low)—Indicate which external memory or peripheral is selected. A particular chip-select is asserted when the transfer address is within the external memory's or peripheral's address space, as defined in CSAR[BA] and CSMR[BAM]. |
| FB_BE_31_24<br>FB_BE_23_16<br>FB_BE_15_8<br>FB_BE_7_0 | O | Byte Enables (active low)—Indicate that data is to be latched or driven onto a specific byte lane of the data bus. CSCR[BEM] determines if these signals are asserted on reads and writes or on writes only.<br>For external SRAM or flash devices, the active low FB_BE outputs should be connected to individual byte strobe signals. |
| FB_OE_b | O | Output Enable (active low)—Sent to the external memory or peripheral to enable a read transfer. This signal is asserted during read accesses only when a chip-select matches the current address decode. |

*Table continues on the next page...*

## Table 20-1. FlexBus signal descriptions (continued)

| Signal | I/O | Function |
|---|---|---|
| FB_R/W_b | O | Read/Write—Indicates whether the current bus operation is a read operation (FB_R/W_b high) or a write operation (FB_R/W_b low). |
| FB_TS_b | O | Transfer Start—Indicates that the chip has begun a bus transaction and that the address and attributes are valid.<br>An inverted FB_TS_b is available as an address latch enable (FB_ALE), which indicates when the address is being driven on the FB_AD bus.<br>FB_TS_b /FB_ALE is asserted for one bus clock cycle.<br>The chip can extend this signal until the first positive clock edge after active low FB_CS asserts. See CSCR[EXTS] and Extended Transfer Start/Address Latch Enable. |
| FB_ALE | O | Address Latch Enable—Indicates when the address is being driven on the FB_A bus (inverse of FB_TS_b). |
| FB_TSIZ1–FB_TSIZ0 | O | Transfer Size—Indicates (along with FB_TBST_b) the data transfer size of the current bus operation. The interface supports 8-, 16-, and 32-bit operand transfers and allows accesses to 8-, 16-, and 32-bit data ports.<br>• 00b = 4 bytes<br>• 01b = 1 byte<br>• 10b = 2 bytes<br>• 11b = 16 bytes (line)<br>For misaligned transfers, FB_TSIZ1–FB_TSIZ0 indicate the size of each transfer. For example, if a 32-bit access through a 32-bit port device occurs at a misaligned offset of 1h, 8 bits are transferred first (FB_TSIZ1–FB_TSIZ0 = 01b), 16 bits are transferred next at offset 2h (FB_TSIZ1–FB_TSIZ0 = 10b), and the final 8 bits are transferred at offset 4h (FB_TSIZ1–FB_TSIZ0 = 01b).<br>For aligned transfers larger than the port size, FB_TSIZ1–FB_TSIZ0 behave as follows:<br>• If bursting is used, FB_TSIZ1–FB_TSIZ0 are driven to the transfer size.<br>• If bursting is inhibited, FB_TSIZ1–FB_TSIZ0 first show the entire transfer size and then show the port size.<br>For burst-inhibited transfers, FB_TSIZ1–FB_TSIZ0 change with each FB_TS_b assertion to reflect the next transfer size.<br>For transfers to port sizes smaller than the transfer size, FB_TSIZ1–FB_TSIZ0 indicate the size of the entire transfer on the first access and the size of the current port transfer on subsequent transfers. For example, for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are 00b for the first transaction and 01b for the next three transactions. If bursting is used for a 32-bit write to an 8-bit port, FB_TSIZ1–FB_TSIZ0 are driven to 00b for the entire transfer. |
| FB_TBST_b | O | Transfer Burst—Indicates that a burst transfer is in progress as driven by the chip. A burst transfer can be 2 to 16 beats depending on FB_TSIZ1–FB_TSIZ0 and the port size.<br>**Note:** When a burst transfer is in progress (FB_TBST_b = 0b), the transfer size is 16 bytes (FB_TSIZ1–FB_TSIZ0 = 11b), and the address is misaligned within the 16-byte boundary, the external memory or peripheral must be able to wrap around the address. |

*Table continues on the next page...*

**Table 20-1. FlexBus signal descriptions (continued)**

| Signal | I/O | Function |
|---|---|---|
| FB_TA_b | I | Transfer Acknowledge—Indicates that the external data transfer is complete. When FB_TA_b is asserted during a read transfer, FlexBus latches the data and then terminates the transfer. When FB_TA_b is asserted during a write transfer, the transfer is terminated.<br><br>If auto-acknowledge is disabled (CSCR[AA] = 0), the external memory or peripheral drives FB_TA_b to terminate the transfer. If auto-acknowledge is enabled (CSCR[AA] = 1), FB_TA_b is generated internally after a specified number of wait states, or the external memory or peripheral may assert external FB_TA_b before the wait-state countdown to terminate the transfer early. The chip deasserts active low FB_CS one cycle after the last FB_TA_b is asserted. During read transfers, the external memory or peripheral must continue to drive data until FB_TA_b is recognized. For write transfers, the chip continues driving data one clock cycle after active low FB_CS is deasserted.<br><br>The number of wait states is determined by CSCR or the external FB_TA_b input. If the external FB_TA_b is used, the external memory or peripheral has complete control of the number of wait states.<br><br>**Note:** External memory or peripherals should assert FB_TA_b only while the active low FB_CS signal to the external memory or peripheral is asserted.<br><br>The CSPMCR register controls muxing of FB_TA_b with other signals. When the CSPMCR register does not allow FB_TA_b control, auto-acknowledge must be used (CSCR[AA] =1'b1); otherwise the bus may hang. |
| FB_CLK | O | FlexBus Clock Output |

## 20.4 Memory Map/Register Definition

The following tables describe the registers and bit meanings for configuring chip-select operation.

The actual number of chip selects available depends upon the device and its pin configuration. If the device does not support certain chip select signals or the pin is not configured for a chip-select function, then that corresponding set of chip-select registers has no effect on an external pin.

### Note

You must set CSMR0[V] before the chip select registers take effect.

A bus error occurs when writing to reserved register locations (for example, when writing to the region between the CS registers and the pin mux). Addresses *base address* + (0x48 - 0x5C) are reserved.

## 20.4.1 FB register descriptions

### 20.4.1.1 FB Memory map

FB base address: 4000_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Chip Select Address Register (CSAR0) | 32 | RW | 0000_0000h |
| 4h | Chip Select Mask Register (CSMR0) | 32 | RW | 0000_0000h |
| 8h | Chip Select Control Register (CSCR0) | 32 | RW | See description. |
| Ch | Chip Select Address Register (CSAR1) | 32 | RW | 0000_0000h |
| 10h | Chip Select Mask Register (CSMR1) | 32 | RW | 0000_0000h |
| 14h | Chip Select Control Register (CSCR1) | 32 | RW | See description. |
| 18h | Chip Select Address Register (CSAR2) | 32 | RW | 0000_0000h |
| 1Ch | Chip Select Mask Register (CSMR2) | 32 | RW | 0000_0000h |
| 20h | Chip Select Control Register (CSCR2) | 32 | RW | See description. |
| 24h | Chip Select Address Register (CSAR3) | 32 | RW | 0000_0000h |
| 28h | Chip Select Mask Register (CSMR3) | 32 | RW | 0000_0000h |
| 2Ch | Chip Select Control Register (CSCR3) | 32 | RW | See description. |
| 30h | Chip Select Address Register (CSAR4) | 32 | RW | 0000_0000h |
| 34h | Chip Select Mask Register (CSMR4) | 32 | RW | 0000_0000h |
| 38h | Chip Select Control Register (CSCR4) | 32 | RW | See description. |
| 3Ch | Chip Select Address Register (CSAR5) | 32 | RW | 0000_0000h |
| 40h | Chip Select Mask Register (CSMR5) | 32 | RW | 0000_0000h |
| 44h | Chip Select Control Register (CSCR5) | 32 | RW | See description. |
| 60h | Chip Select Port Multiplexing Control Register (CSPMCR) | 32 | RW | 0000_0000h |

### 20.4.1.2 Chip Select Address Register (CSAR0 - CSAR5)

#### 20.4.1.2.1 Offset

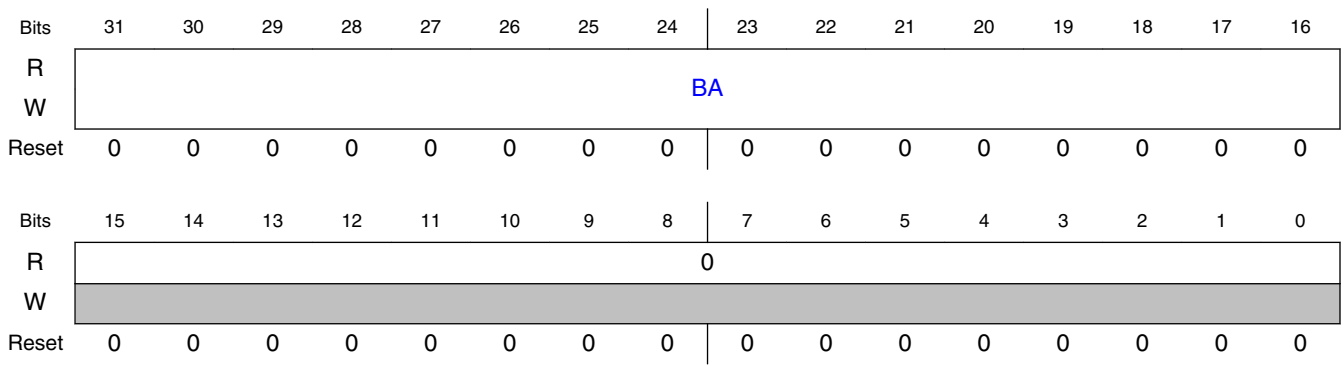
For a = 0 to 5:

| Register | Offset |
|---|---|
| CSARa | 0h + (a × Ch) |

#### 20.4.1.2.2 Function

Specifies the associated chip-select's base address.

#### 20.4.1.2.3 Diagram




#### 20.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31-16<br>BA | Base Address<br>Defines the base address for memory dedicated to the associated chip-select. BA is compared to bits 31–16 on the internal address bus to determine if the associated chip-select's memory is being accessed.<br>**NOTE:** Because the FlexBus module is one of the slaves connected to the crossbar switch, it is only accessible within a certain memory range. See the chip memory map for the applicable FlexBus "expansion" address range for which the chip-selects can be active. Set the CSARn and CSMRn registers appropriately before accessing this region. |
| 15-0<br>— | Reserved |

### 20.4.1.3 Chip Select Mask Register (CSMR0 - CSMR5)

#### 20.4.1.3.1 Offset

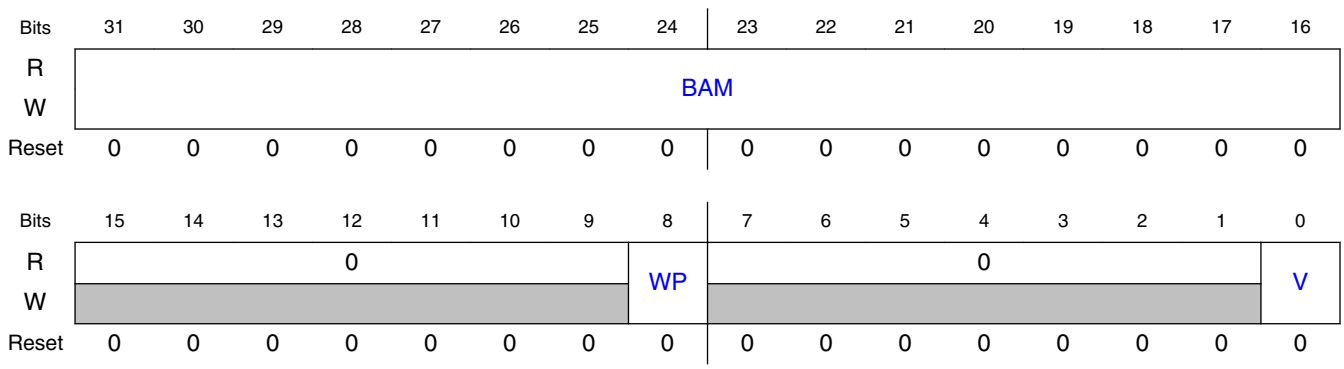
For a = 0 to 5:

| Register | Offset |
|---|---|
| CSMRa | 4h + (a × Ch) |

#### 20.4.1.3.2 Function

Specifies the address mask and allowable access types for the associated chip-select.

#### 20.4.1.3.3 Diagram




#### 20.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>BAM | Base Address Mask<br>Defines the associated chip-select's block size by masking address bits.<br>0000000000000000b - The corresponding address bit in CSAR is used in the chip-select decode.<br>0000000000000001b - The corresponding address bit in CSAR is a don't care in the chip-select decode. |
| 15-9<br>— | Reserved |
| 8<br>WP | Write Protect<br>Controls write accesses to the address range in the corresponding CSAR.<br>0b - Write accesses are allowed.<br>1b - Write accesses are not allowed. Attempting to write to the range of addresses for which the WP bit is set results in a bus error termination of the internal cycle and no external cycle. |
| 7-1<br>— | Reserved |
| 0 | Valid |

| Field | Function |
|---|---|
| V | Specifies whether the corresponding CSAR, CSMR, and CSCR contents are valid. Programmed chip-selects do not assert until the V bit is 1b (except for FB_CS0_B, which acts as the global chip-select).<br>**NOTE:** At reset, FB_CS0_B will fire for any access to the FlexBus memory region. CSMR0[V] must be set as part of the chip select initialization sequence to allow other chip selects to function as programmed.<br>0b - Chip-select is invalid.<br>1b - Chip-select is valid. |

## 20.4.1.4 Chip Select Control Register (CSCR0 - CSCR5)

### 20.4.1.4.1 Offset

For a = 0 to 5:

| Register | Offset |
|---|---|
| CSCRa | 8h + (a × Ch) |

### 20.4.1.4.2 Function

Controls the auto-acknowledge, address setup and hold times, port size, burst capability, and number of wait states for the associated chip select. FB_CS0_B allows address decoding for an external device to serve as the boot memory before system initialization and configuration are completed.

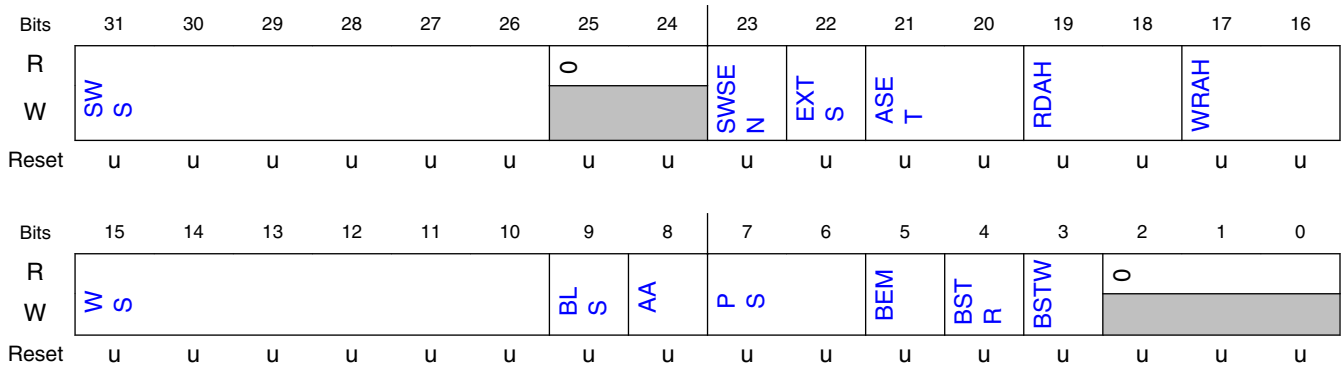
**NOTE**

To support the global chip-select (FB_CS0_B), the CSCR0 reset values differ from the other CSCRs (CSCR1 - CSCR5 all reset to 0). The reset value of CSCR0 is as follows:

- Bits 31 – 24 are 0b
- Bit 23 – 3 are chip-dependent
- Bits 2 – 0 are 0b

See the chip configuration details for your particular chip for information on the exact CSCR0 reset value.

### 20.4.1.4.3 Diagram




### 20.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-26<br>SWS | Secondary Wait States<br>Used only when the SWSEN bit is 1b. Specifies the number of wait states inserted before an internal transfer acknowledge is generated for a burst transfer (except for the first termination, which is controlled by WS). |
| 25-24<br>— | Reserved |
| 23<br>SWSEN | Secondary Wait State Enable<br>0b - Disabled. A number of wait states (specified by WS) are inserted before an internal transfer acknowledge is generated for all transfers.<br>1b - Enabled. A number of wait states (specified by SWS) are inserted before an internal transfer acknowledge is generated for burst transfer secondary terminations. |
| 22<br>EXTS | EXTS<br>Extended Transfer Start/Extended Address Latch Enable<br>Controls how long FB_TS_B/FB_ALE is asserted.<br>0b - Disabled. FB_TS_B/FB_ALE asserts for one bus clock cycle.<br>1b - Enabled. FB_TS_B/FB_ALE remains asserted until the first positive clock edge after FB_CSn_B asserts. |
| 21-20<br>ASET | Address Setup<br>Controls when the chip-select is asserted with respect to assertion of a valid address and attributes.<br>00b - Assert FB_CSn_B on the first rising clock edge after the address is asserted (default for all but FB_CS0_B).<br>01b - Assert FB_CSn_B on the second rising clock edge after the address is asserted.<br>10b - Assert FB_CSn_B on the third rising clock edge after the address is asserted.<br>11b - Assert FB_CSn_B on the fourth rising clock edge after the address is asserted (default for FB_CS0_B ). |
| 19-18<br>RDAH | Read Address Hold or Deselect<br>Controls the address and attribute hold time after the termination during a read cycle that hits in the associated chip-select's address space. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE:** • The hold time applies only at the end of a transfer. Therefore, during a burst transfer or a transfer to a port size smaller than the transfer size, the hold time is only added after the last bus cycle.<br>• The number of cycles the address and attributes are held after FB_CSn_B deassertion depends on the value of the AA bit.<br>00b - When AA is 1b, 1 cycle. When AA is 0b, 0 cycles.<br>01b - When AA is 1b, 2 cycles. When AA is 0b, 1 cycle.<br>10b - When AA is 1b, 3 cycles. When AA is 0b, 2 cycles.<br>11b - When AA is 1b, 4 cycles. When AA is 0b, 3 cycles. |
| 17-16<br>WRAH | Write Address Hold or Deselect<br>Controls the address, data, and attribute hold time after the termination of a write cycle that hits in the associated chip-select's address space.<br>**NOTE:** The hold time applies only at the end of a transfer. Therefore, during a burst transfer or a transfer to a port size smaller than the transfer size, the hold time is only added after the last bus cycle.<br>00b - 1 cycle (default for all but FB_CS0_B)<br>01b - 2 cycles<br>10b - 3 cycles<br>11b - 4 cycles (default for FB_CS0_B) |
| 15-10<br>WS | Wait States<br>Specifies the number of wait states inserted after FlexBus asserts the associated chip-select and before an internal transfer acknowledge is generated (WS = 00h inserts 0 wait states, ..., WS = 3Fh inserts 63 wait states). |
| 9<br>BLS | Byte-Lane Shift<br>Specifies if data on FB_AD appears left-aligned or right-aligned during the data phase of a FlexBus access.<br>0b - Not shifted. Data is left-aligned on FB_AD.<br>1b - Shifted. Data is right-aligned on FB_AD. |
| 8<br>AA | Auto-Acknowledge Enable<br>Asserts the internal transfer acknowledge for accesses specified by the chip-select address.<br>**NOTE:** If AA is 1b for a corresponding FB_CSn_B and the external system asserts an external FB_TA_B before the wait-state countdown asserts the internal FB_TA_B, the cycle is terminated. Burst cycles increment the address bus between each internal termination.<br>**NOTE:** This field must be 1b if CSPMCR disables FB_TA_B.<br>0b - Disabled. No internal transfer acknowledge is asserted and the cycle is terminated externally.<br>1b - Enabled. Internal transfer acknowledge is asserted as specified by WS. |
| 7-6<br>PS | Port Size<br>Specifies the data port width of the associated chip-select, and determines where data is driven during write cycles and where data is sampled during read cycles.<br>00b - 32-bit port size. Valid data is sampled and driven on FB_D[31:0].<br>01b - 8-bit port size. Valid data is sampled and driven on FB_D[31:24] when BLS is 0b, or FB_D[7:0] when BLS is 1b.<br>1xb - 16-bit port size. Valid data is sampled and driven on FB_D[31:16] when BLS is 0b, or FB_D[15:0] when BLS is 1b. |
| 5<br>BEM | Byte-Enable Mode<br>Specifies whether the corresponding FB_BE_B is asserted for read accesses. Certain memories have byte enables that must be asserted during reads and writes. Write 1b to the BEM bit in the relevant CSCR to provide the appropriate mode of byte enable support for these SRAMs.<br>0b - FB_BE_B is asserted for data write only.<br>1b - FB_BE_B is asserted for data read and write accesses. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 4<br>BSTR | Burst-Read Enable<br>Specifies whether burst reads are enabled for memory associated with each chip select.<br>0b - Disabled. Data exceeding the specified port size is broken into individual, port-sized, non-burst reads. For example, a 32-bit read from an 8-bit port is broken into four 8-bit reads.<br>1b - Enabled. Enables data burst reads larger than the specified port size, including 32-bit reads from 8- and 16-bit ports, 16-bit reads from 8-bit ports, and line reads from 8-, 16-, and 32-bit ports. |
| 3<br>BSTW | Burst-Write Enable<br>Specifies whether burst writes are enabled for memory associated with each chip select.<br>0b - Disabled. Data exceeding the specified port size is broken into individual, port-sized, non-burst writes. For example, a 32-bit write to an 8-bit port takes four byte writes.<br>1b - Enabled. Enables burst write of data larger than the specified port size, including 32-bit writes to 8- and 16-bit ports, 16-bit writes to 8-bit ports, and line writes to 8-, 16-, and 32-bit ports. |
| 2-0<br>— | Reserved |

## 20.4.1.5 Chip Select Port Multiplexing Control Register (CSPMCR)

### 20.4.1.5.1 Offset

| Register | Offset |
|---|---|
| CSPMCR | 60h |

### 20.4.1.5.2 Function

Controls the multiplexing of the FlexBus signals. Use the CSPMCR register (Chip Select Port Multiplexing Control) and port control registers to configure which control signals are available on the external pins.

The multiplexing of the FlexBus address and data signals is controlled by the port control module. However, the multiplexing of some of the FlexBus control signals are controlled by **both** the FlexBus and port control modules. The FlexBus's CSPMCR register configures which FlexBus signals are available from the FlexBus module, and the port control module registers control whether the FlexBus or other module signals are available on the external pin.

**NOTE**

A bus error occurs when you do any of the following:

- Write to a reserved address
- Write to a reserved field in this register, or
- Access this register using a size other than 32 bits.

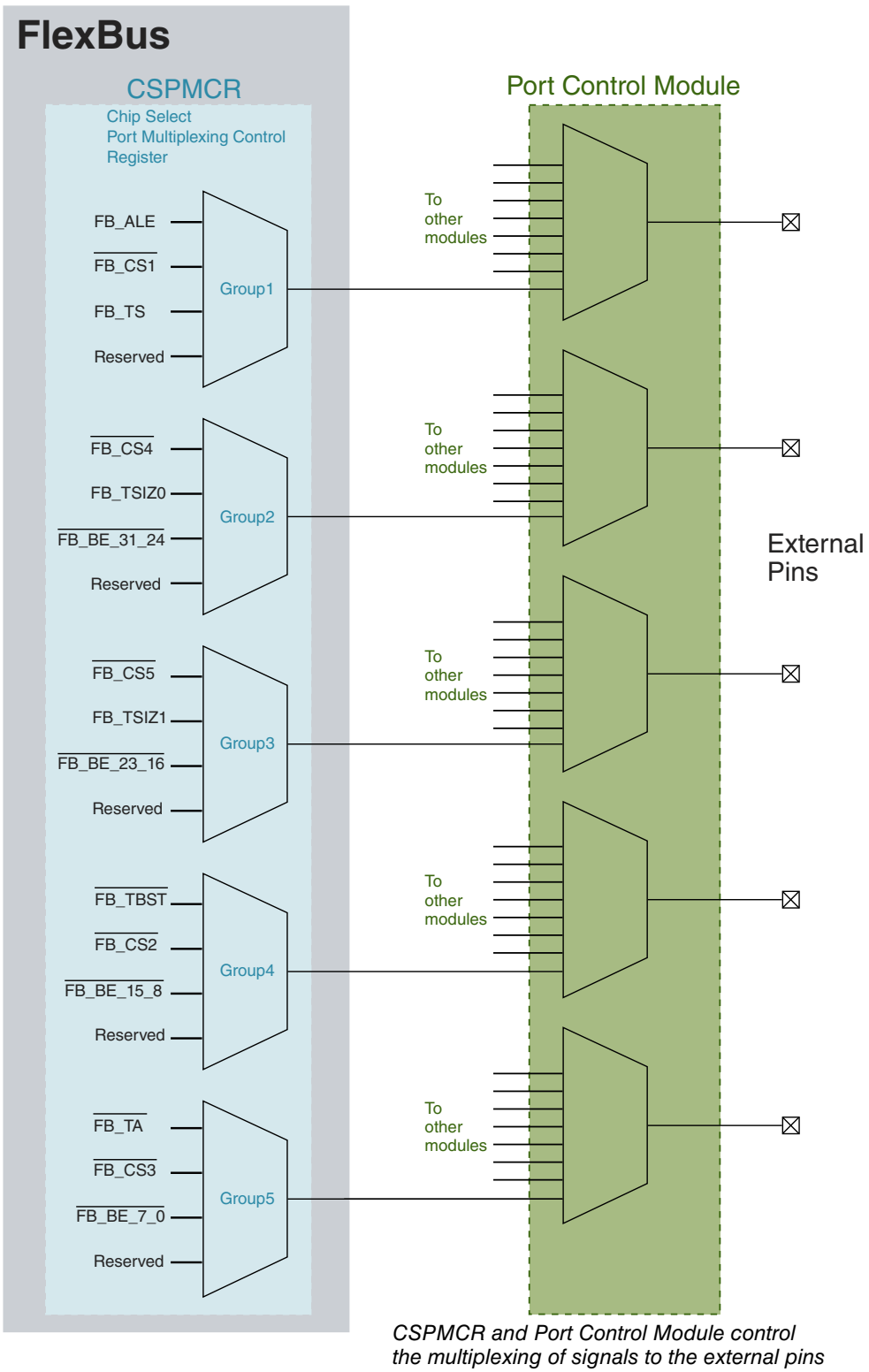



**Figure 20-4. FlexBus and Port Control module**

### 20.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GROUP1 | | | | GROUP2 | | | | GROUP3 | | | | GROUP4 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GROUP5 | | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 20.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-28<br>GROUP1 | FlexBus Signal Group 1 Multiplex control<br>Controls the multiplexing of the FB_ALE, FB_CS1_B, and FB_TS_B signals.<br>0000b - FB_ALE<br>0001b - FB_CS1_B<br>0010b - FB_TS_B |
| 27-24<br>GROUP2 | FlexBus Signal Group 2 Multiplex control<br>Controls the multiplexing of the FB_CS4_B, FB_TSIZ0, and FB_BE_31_24_B signals.<br>0000b - FB_CS4_B<br>0001b - FB_TSIZ0<br>0010b - FB_BE_31_24_B |
| 23-20<br>GROUP3 | FlexBus Signal Group 3 Multiplex control<br>Controls the multiplexing of the FB_CS5_B, FB_TSIZ1, and FB_BE_23_16_B signals.<br>0000b - FB_CS5_B<br>0001b - FB_TSIZ1<br>0010b - FB_BE_23_16_B |
| 19-16<br>GROUP4 | FlexBus Signal Group 4 Multiplex control<br>Controls the multiplexing of the FB_TBST_B, FB_CS2_B, and FB_BE_15_8_B signals.<br>0000b - FB_TBST_B<br>0001b - FB_CS2_B<br>0010b - FB_BE_15_8_B |
| 15-12<br>GROUP5 | FlexBus Signal Group 5 Multiplex control<br>Controls the multiplexing of the FB_TA_B, FB_CS3_B, and FB_BE_7_0_B signals.<br>**NOTE:** When GROUP5 is not 0000b, you must write 1b to the CSCR[AA] bit. Otherwise, the bus hangs during a transfer.<br>0000b - FB_TA_B<br>0001b - FB_CS3_B. You must also write 1b to CSCR[AA].<br>0010b - FB_BE_7_0_B. You must also write 1b to CSCR[AA]. |
| 11-0<br>— | Reserved |

# 20.5 Functional description

## 20.5.1 Address and data bus configurations

**NOTE**

Address and data line configurations (like 32-bit non-multiplexed address and data busses) require signals to actually be available on the pins of your specific device's package. To make sure that the address and data bus signals are available on a specific device's pins, see your device's signal multiplexing table for that device package.

FlexBus supports:

- **Multiplexed address and data lines:**
  - Multiplexed 32-bit address and 32-bit data
  - Multiplexed 32-bit address and 16-bit data (non-multiplexed 16-bit address and 16-bit data)
  - Multiplexed 32-bit address and 8-bit data (non-multiplexed 24-bit address and 8-bit data)
- **Nonmultiplexed address and data lines:**
  - Non-multiplexed 32-bit address and 32-bit data busses
  - Non-multiplexed 32-bit address and 16-bit data busses
  - Non-multiplexed 32-bit address and 8-bit data busses
- **20-bit addressing:**
  - Up to a 20-bit address (multiplexed) with 16-bit data (write-masking of upper/lower bytes is not supported)
  - Up to a 20-bit address (multiplexed) with 8-bit data
  - Up to a 20-bit address (non-multiplexed) with 8-bit data

## 20.5.2 Address comparison

When a bus cycle is routed to FlexBus, FlexBus compares the transfer address to the base address (see CSAR[BA]) and base address mask (see CSMR[BAM]). This table describes how FlexBus decides to assert a chip-select and complete the bus cycle based on the address comparison.

| When the transfer address | Then FlexBus |
|---|---|
| Matches one address register configuration | Asserts the appropriate chip-select, generating a FlexBus bus cycle as defined in the appropriate CSCR.<br>If CSMR[WP] is set and a write access is performed, FlexBus terminates the internal bus cycle with a bus error, does not assert a chip-select, and does not perform an external bus cycle. |
| Does not match an address register configuration | Terminates the transfer with a bus error response, does not assert a chip-select, and does not perform a FlexBus cycle. |
| Matches more than one address register configuration | Terminates the transfer with a bus error response, does not assert a chip-select, and does not perform a FlexBus cycle. |

### 20.5.3 Address driven on address bus

Regardless of the external memory's or peripheral's address size,

- For multiplexed address and data busses, FlexBus always drives a 32-bit address on the FB_AD bus.
- For nonmultiplexed address and data busses, FlexBus always drives a 24-bit address on the FB_A bus.

### 20.5.4 Connecting address/data lines

The external device must connect its address and data lines as follows:

- Address lines
  - FB_AD from FB_AD0 upward
- Data lines
  - If CSCR[BLS] = 0, FB_AD from FB_AD31 downward
  - If CSCR[BLS] = 1, FB_AD from FB_AD0 upward

### 20.5.5 Bit ordering

No bit ordering is required when connecting address and data lines to the FB_A and FB_D buses. For example, a full 16-bit address/16-bit data device connects its addr15–addr0 to FB_AD16–FB_AD1 and data15–data0 to FB_AD31–FB_AD16. See Data-byte alignment and physical connections for a graphical connection.

### 20.5.6 Global chip-select

After system reset, FB_CS0(active-low) acts as the global (boot) chip-select to support external boot memory accesses before system initialization. Its operation differs from other external chip-select outputs after system reset. After system reset, FB_CS0 is asserted for every external access. No other chip-select can be used until until you write 1b to CSMR0[V]. At this point FB_CS0 functions as programmed. After this, FB_CS5–FB_CS1 can be used as well.

## 20.5.7 Data transfer signals

Data transfers between FlexBus and the external memory or peripheral involve these signals:

- Address/data bus (FB_AD31–FB_AD0 (multiplexed address/data bus)FB_A23–FB_A0, FB_D31–FB_D0 (nonmultiplexed address/data bus))
- Control signals (FB_TS_b/FB_ALE, FB_TA_b, FB_CS_b*n*, FB_OE_b, FB_R/W_b, FB_BE*n*)
- Attribute signals (FB_TBST_b, FB_TSIZ1–FB_TSIZ0)

## 20.5.8 Signal transitions

These signals change on the rising edge of the FlexBus clock (FB_CLK):

- Address
- Write data
- FB_TS_b/FB_ALE
- active low FB_CSn
- All attribute signals

FlexBus latches the read data on the rising edge of the clock.

## 20.5.9 Data-byte alignment and physical connections

The device aligns data transfers in FlexBus byte lanes with the number of lanes depending on the data port width.

The following figure shows the byte lanes that external memory or peripheral connects to and the sequential transfers of a 32-bit transfer for the supported port sizes when byte lane shift is disabled. For example, an 8-bit memory connects to the single lane

FB_AD31–FB_AD24 (FB_BE_31_24, active low). A 32-bit transfer through this 8-bit port takes 4 transfers, starting with the LSB to the MSB. A 32-bit transfer through a 32-bit port requires one transfer on each 4-byte lane.

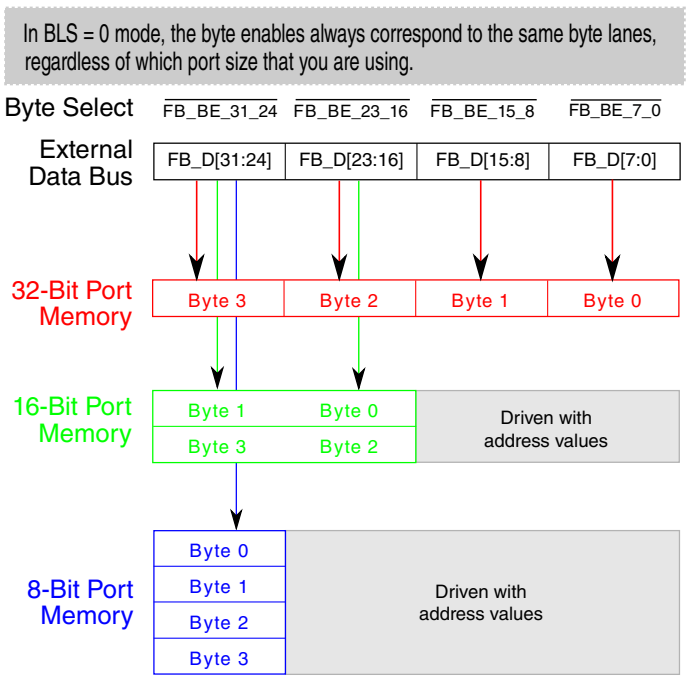



**Figure 20-5. Connections for external memory port sizes (CSCRn[BLS] = 0)**

The following figure shows the byte lanes that external memory or peripheral connects to and the sequential transfers of a 32-bit transfer for the supported port sizes when byte lane shift is enabled.

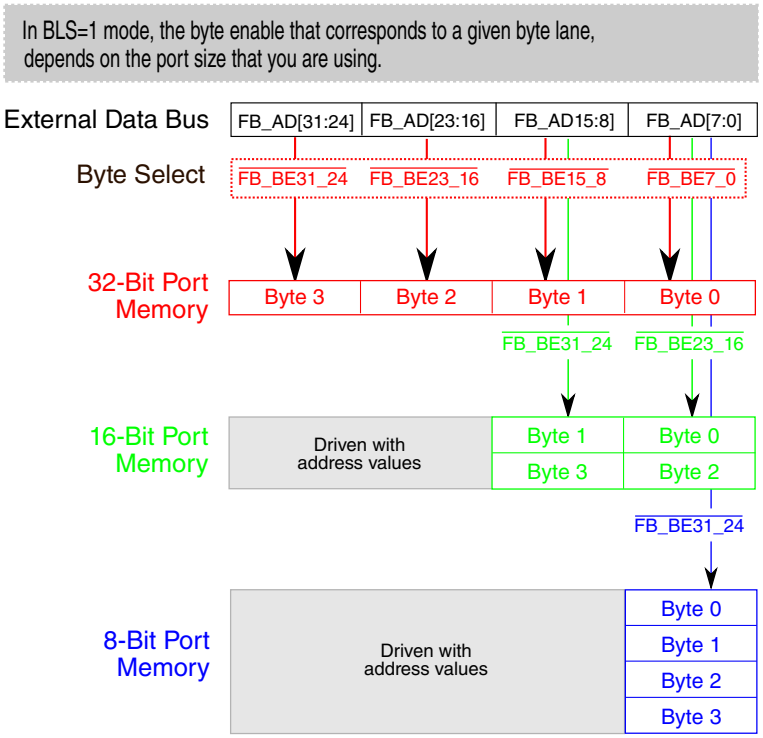



**Figure 20-6. Connections for external memory port sizes (CSCRn[BLS] = 1)**

## 20.5.10 Address/data bus multiplexing

FlexBus supports a single 32-bit wide multiplexed address and data bus (FB_AD31–FB_AD0). FlexBus always drives the full 32-bit address on the first clock of a bus cycle. During the data phase, the FB_AD31– FB_AD0 lines used for data are determined by the programmed port size and BLS setting for the corresponding chip-select. FlexBus continues to drive the address on any FB_AD31– FB_AD0 lines not used for data.

### 20.5.10.1 FlexBus multiplexed operating modes for CSCRn[BLS]=0

This table shows the supported combinations of address and data bus widths when CSCRn[BLS] is 0b.

| Port size and phase | | FB_AD | | | |
|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 |
| 32-bit | Address phase | Address | | | |
| | Data phase | Data | | | |
| 16-bit | Address phase | Address | | | |
| | Data phase | Data | | Address | |

*Table continues on the next page...*

| Port size and phase | | FB_AD | | | |
|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 |
| 8-bit | Address phase | Address | | | |
| | Data phase | Data | Address | | |

## 20.5.10.2 FlexBus multiplexed operating modes for CSCRn[BLS]=1

This table shows the supported combinations of address and data bus widths when CSCRn[BLS] is 1b.

| Port size and phase | | FB_AD | | | |
|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 |
| 32-bit | Address phase | Address | | | |
| | Data phase | Data | | | |
| 16-bit | Address phase | Address | | | |
| | Data phase | Address | | Data | |
| 8-bit | Address phase | Address | | | |
| | Data phase | Address | | | Data |

## 20.5.11 Data transfer states

Basic data transfers occur in 4 clocks or states. (See Basic Read Bus Cycle and Basic Write Bus Cycle for examples of basic data transfers.) The FlexBus state machine controls the data-transfer operation. This figure shows the state-transition diagram for basic read and write cycles.

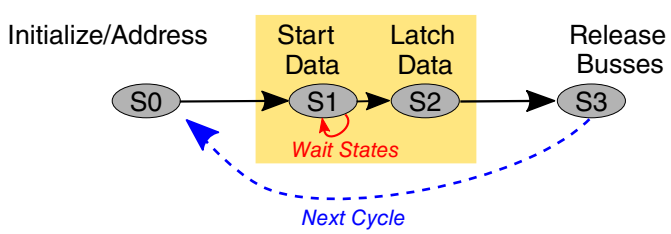



**Figure 20-7. Data transfer state transitions**

| State | | Cycle | Description |
|---|---|---|---|
| Initialize/ address | S0 | All | The read or write cycle is initiated. On the rising clock edge, FlexBus:<br>• Places a valid address on FB_ADn |

*Table continues on the next page...*

| State | | Cycle | Description |
|---|---|---|---|
| | | | • Asserts FB_TS_b/FB_ALE<br>• Drives FB_R/W_b high for a read and low for a write |
| Start data phase | S1 | All | FlexBus:<br>• Negates FB_TS_b/FB_ALE on the rising edge of FB_CLK<br>• Asserts FB_CSn (active low)<br>• Drives the data on FB_AD31– FB_AD*X* for writes<br>• Tristates FB_AD31– FB_AD*X* for reads<br>• Continues to drive the address on FB_AD pins that are unused for data<br><br>• If the external memory or peripheral asserts FB_TA_b, then the process moves to S2.<br>• If FB_TA_b is not asserted internally or externally, then S1 repeats. |
| | | Read | The external memory or peripheral drives the data before the next rising edge of FB_CLK (the rising edge that begins S2) with FB_TA_b asserted. |
| Latch data | S2 | All | • For internal termination, FlexBus negates FB_CSn (active low) and the transfer is complete.<br>• For external termination, the external memory or peripheral negates FB_TA_b, then FlexBus negates FB_CSn (active low) after the rising edge of FB_CLK at the end of S2. |
| | | Read | FlexBus latches the data on the rising clock edge entering S2. The external memory or peripheral can stop driving the data after this edge, or continue to drive the data until the end of S3 or drive the data through any additional address hold cycles. |
| Release busses | S3 | All | FlexBus invalidates the address, data, and FB_R/W_b on the rising edge of FB_CLK at the beginning of S3, terminating the transfer. |

## 20.5.12 FlexBus Timing Examples

**Note**

The timing diagrams in this section use signal names that may not be included on your particular device, so ignore these signals.

**Note**

Throughout this section:

- FB_D[*X*] indicates a 32-, 16-, or 8-bit wide data bus
- FB_A[*Y*] indicates an address bus that can be 32, 24, or 16 bits wide.

### 20.5.12.1 Basic Read Bus Cycle

During a read cycle, the MCU receives data from memory or a peripheral device.

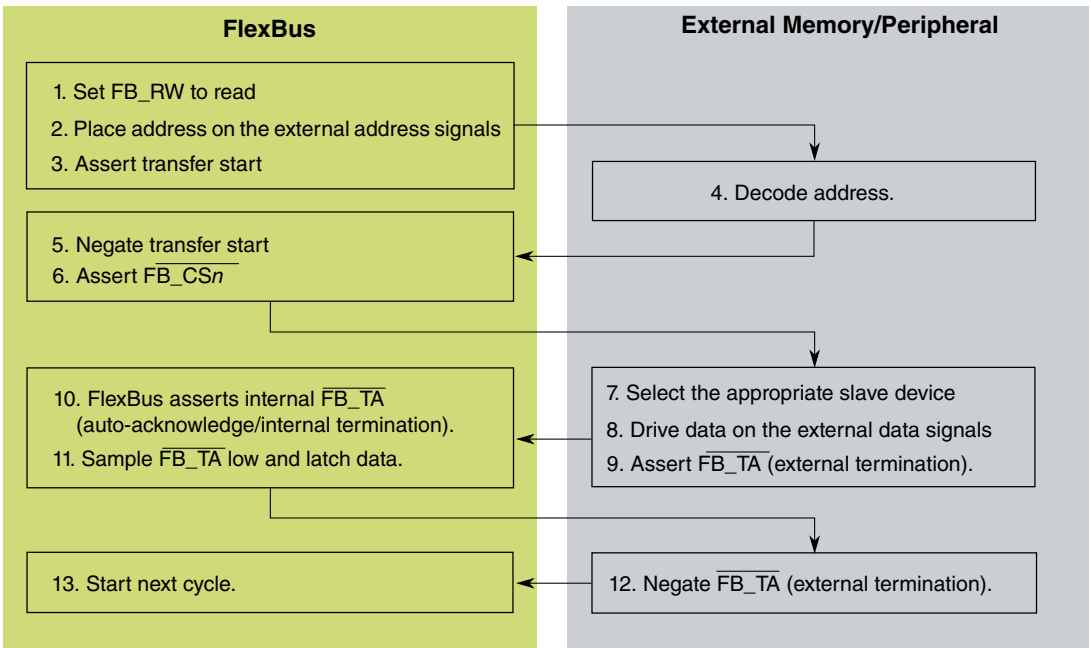



**Figure 20-8. Read Cycle Flowchart**

## Note

FB_TA_b (active low) does not have to be driven by the external device for internally-terminated bus cycles.

## Note

The processor drives the data lines during the first clock cycle of the transfer with the full 32-bit address. This may be ignored by standard connected devices using non-multiplexed address and data buses. However, some applications may find this feature beneficial.

The address and data busses are muxed between the FlexBus and another module. At the end of the read bus cycles the address signals are indeterminate.

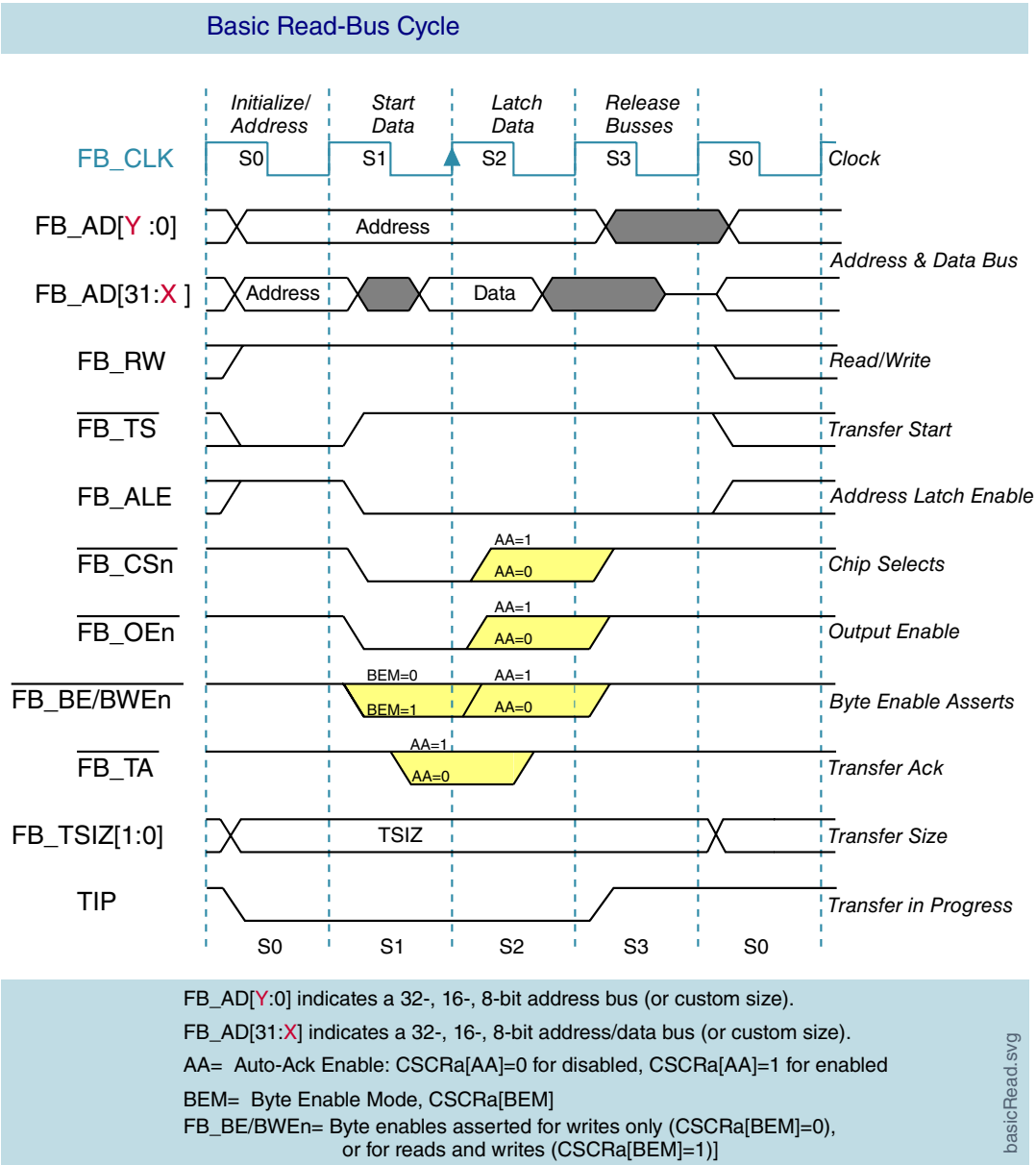



**Figure 20-9. Basic Read Bus Cycle**

## 20.5.12.2 Basic Write Bus Cycle

During a write cycle, the device sends data to memory or to a peripheral device.

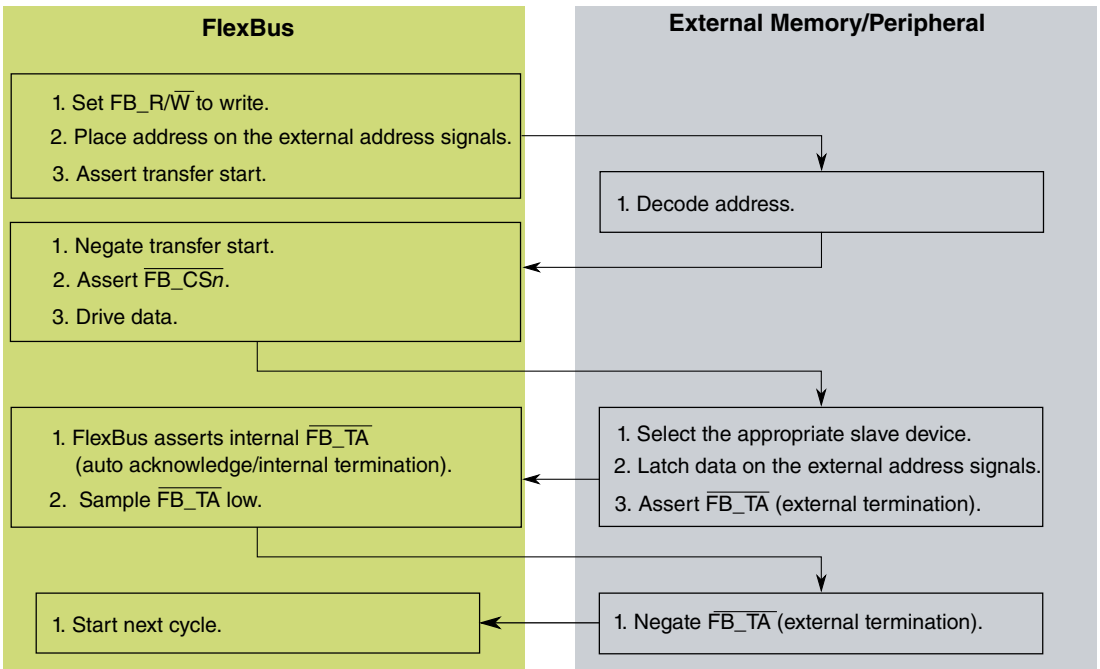



**Figure 20-10. Write Cycle Flowchart**

## Note

The address and data busses are muxed between the FlexBus and another module. At the end of the write bus cycles, the address signals are indeterminate.

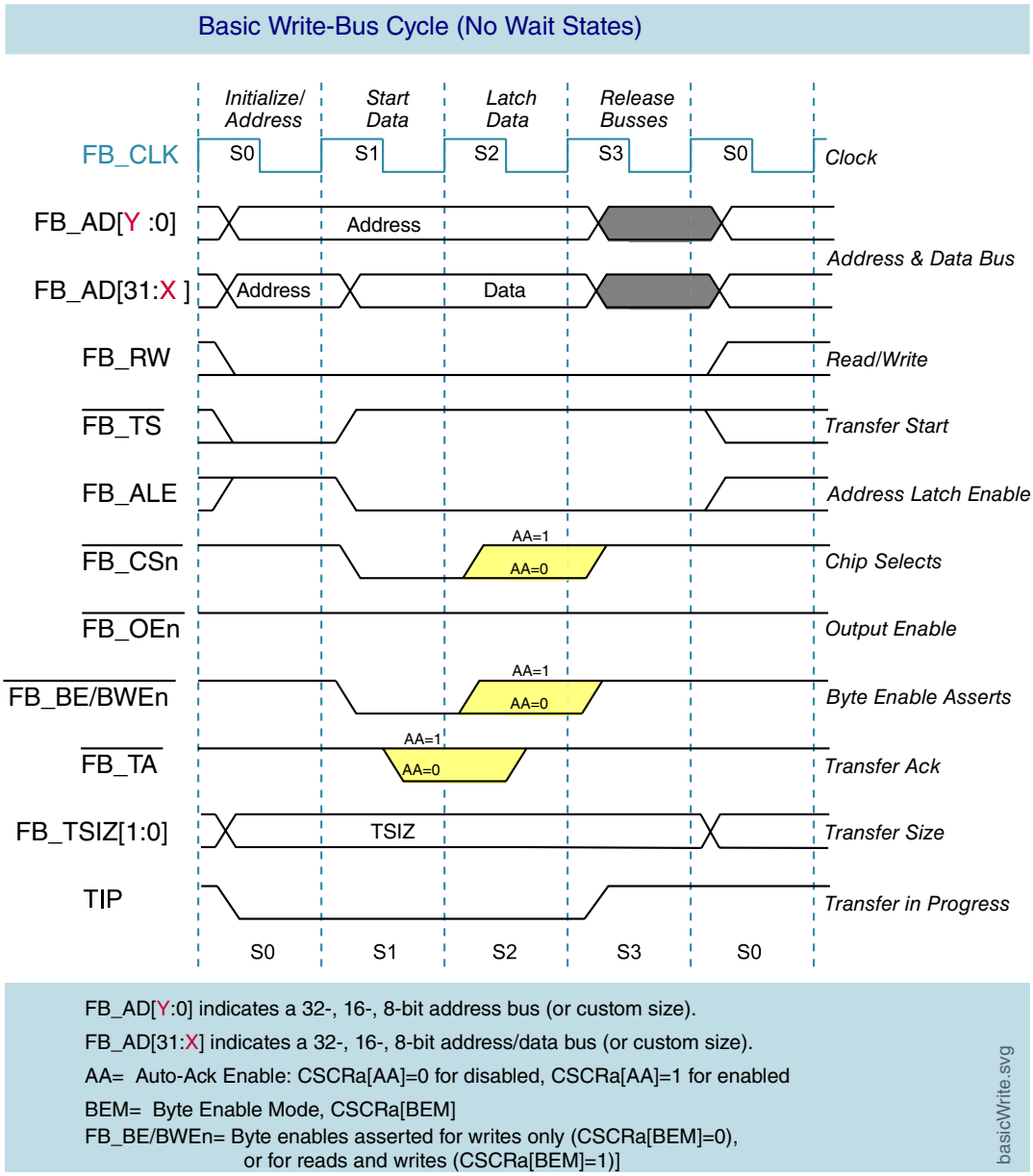



**Figure 20-11. Basic Write Bus Cycle**

## 20.5.12.3 Bus Cycle Sizing

This section shows timing diagrams for various port size scenarios.

**NOTE**

All examples in this "Bus Cycle Sizing" section assume BLS=0.

### 20.5.12.3.1 Bus Cycle Sizing—Byte Transfer, 8-bit Device, No Wait States

The next figure shows the basic byte read transfer to an 8-bit device with no wait states:

- For multiplexed address and data busses, the address is driven on the full FB_AD[31:0] bus in the 1st clock; the device tristates FB_AD[31:24] on the 2nd clock and continues to drive address on FB_AD[23:0] throughout the bus cycle.
- For nonmultiplexed address and data busses, the address is driven on the FB_A[23:8] bus throughout the bus cycle.
- The external device returns the read data on FB_AD[31:24] and may tristate the data line or continue driving the data 1 clock after FB_TA_b (active low) is sampled asserted.

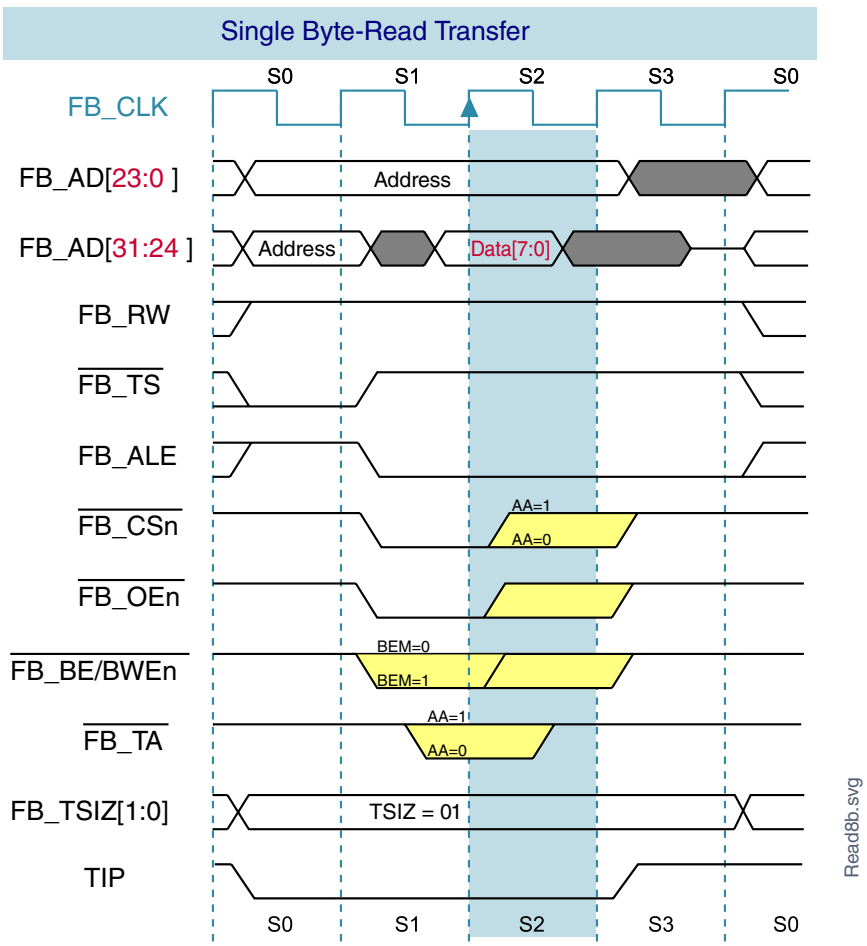



**Figure 20-12. Single Byte-Read Transfer**

The next figure shows the similar configuration for a write transfer. The data is driven from the 2nd clock on FB_AD[31:24].

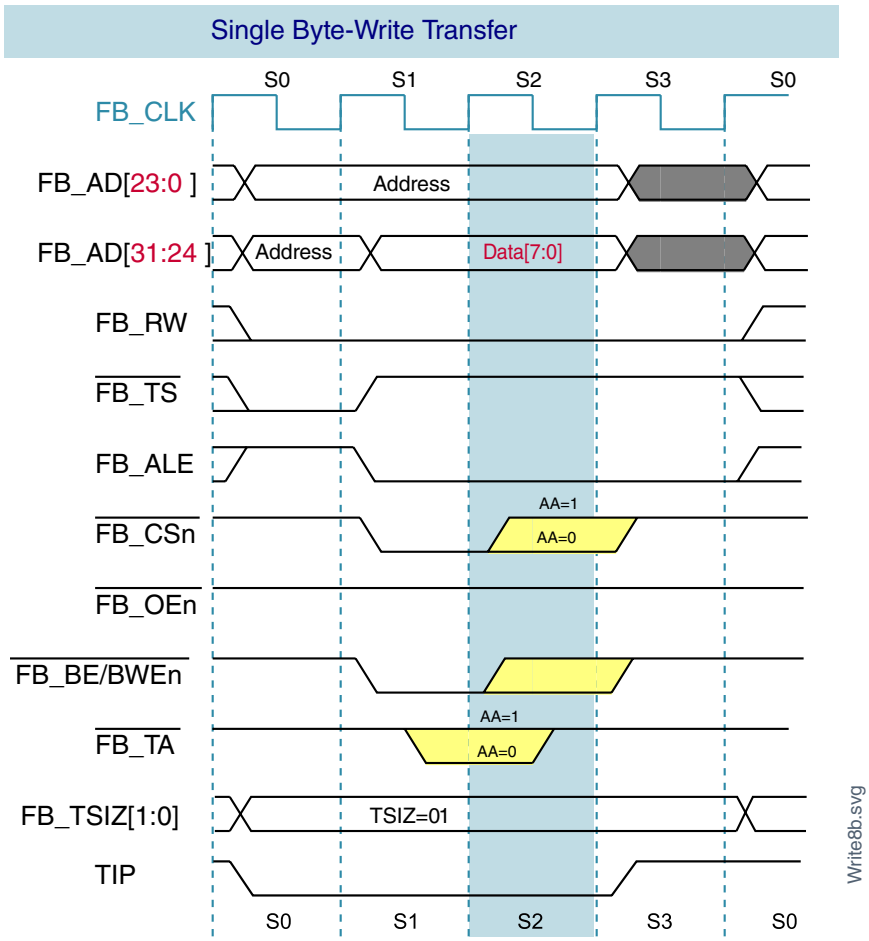



**Figure 20-13. Single Byte-Write Transfer**

### 20.5.12.3.2 Bus Cycle Sizing—Word Transfer, 16-bit Device, No Wait States

The next figure shows a basic word read transfer to a 16-bit device with no wait states.

- For multiplexed address and data busses, the address is driven on the full FB_AD[31:0] bus in the first clock; the device tristates FB_AD[31:16] on the second clock and continues to drive address on FB_AD[15:0] throughout the bus cycle.
- For nonmultiplexed address and data busses, the address is driven on the FB_A[23] bus throughout the bus cycle.
- The external device returns the read data on FB_AD[31:16] and may tristate the data line or continue driving the data one clock after FB_TA_b (active low) is sampled asserted.

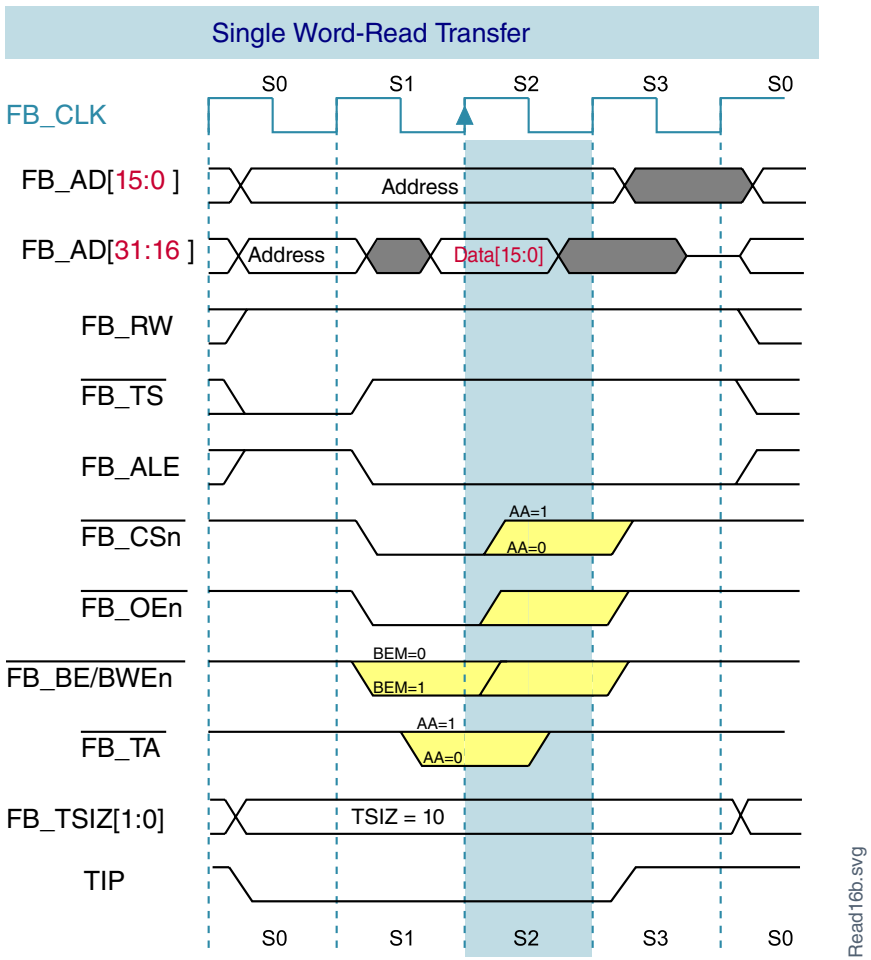



**Figure 20-14. Single Word-Read Transfer**

The next figure shows a basic word write transfer. The data is driven from the second clock on FB_AD[31:16].

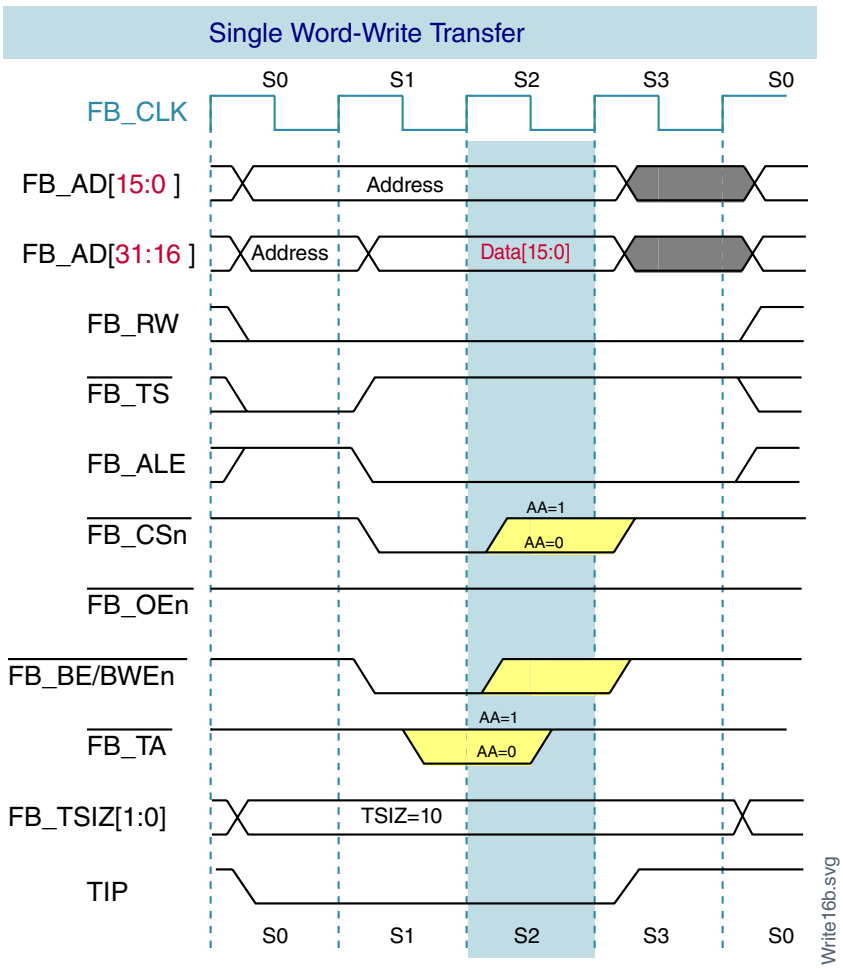



**Figure 20-15. Single Word-Write Transfer**

### 20.5.12.3.3 Bus Cycle Sizing—Longword Transfer, 32-bit Device, No Wait States

The next figure shows a longword read from a 32-bit device.

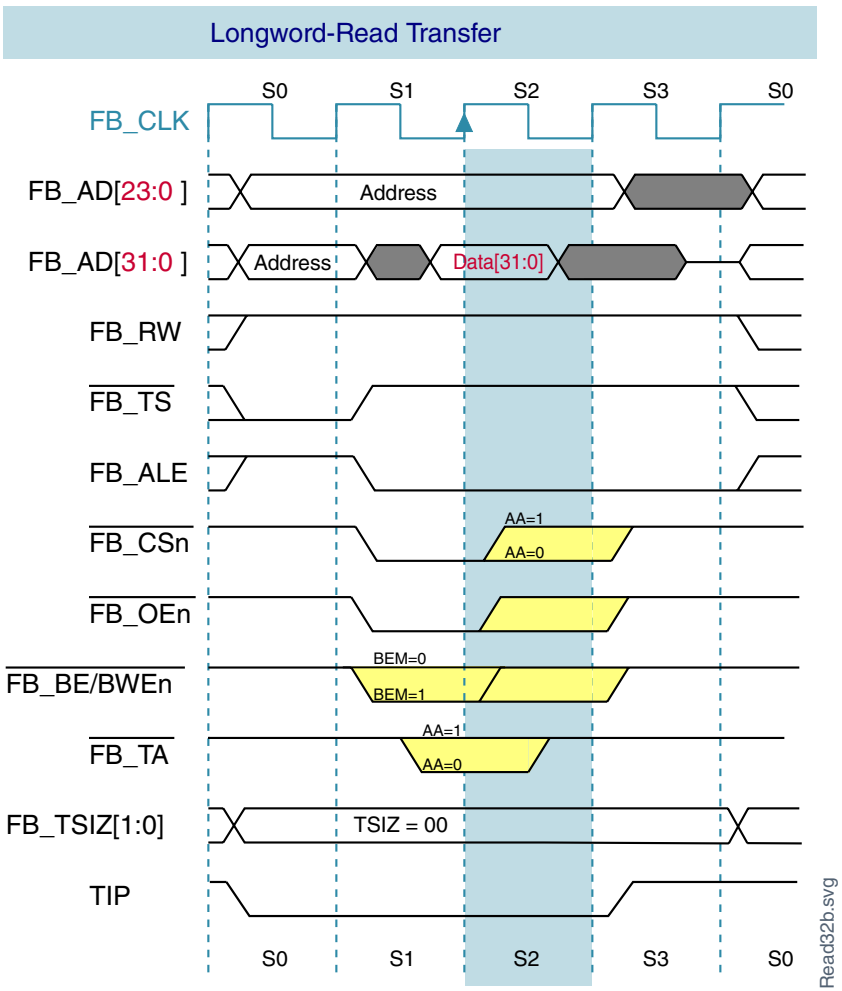



**Figure 20-16. Longword-Read Transfer**

The next figure shows a longword write to a 32-bit device.

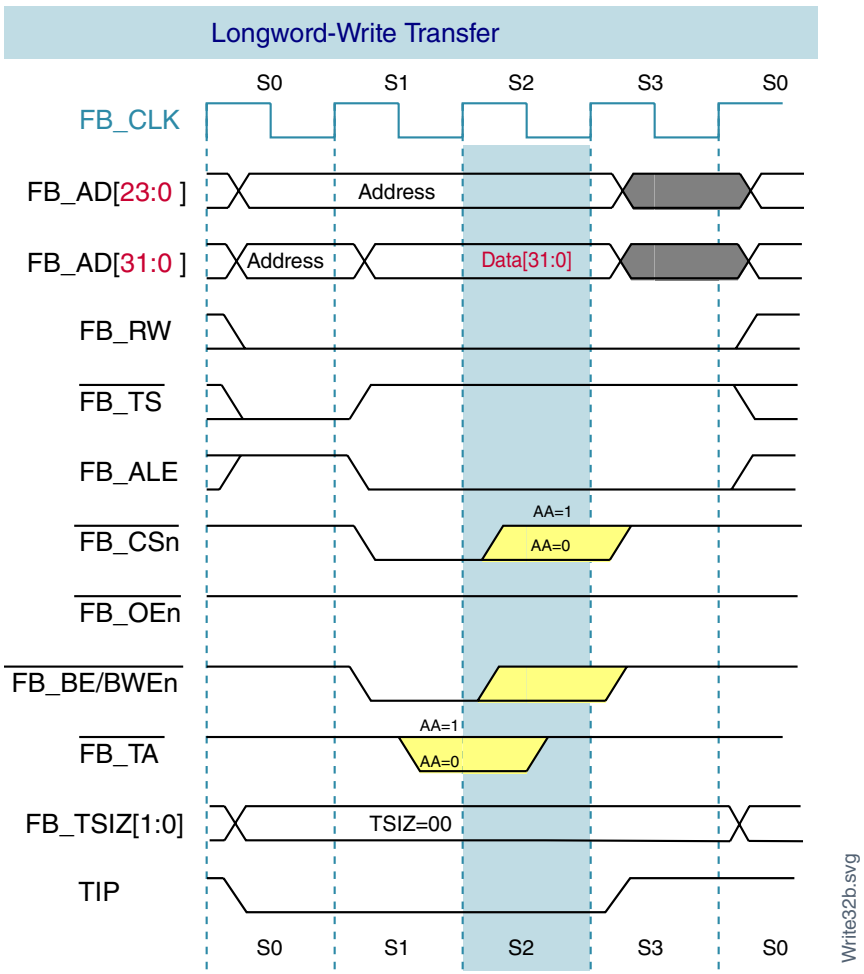



**Figure 20-17. Longword-Write Transfer**

## 20.5.12.4 Timing Variations

The FlexBus module has several features that can change the timing characteristics of a basic read- or write-bus cycle to provide additional address setup, address hold, and time for a device to provide or latch data.

### 20.5.12.4.1 Wait States

Wait states can be inserted before each beat of a transfer by programming the CSCR*n* registers. Wait states can give the peripheral or memory more time to return read data or sample write data.

The following figures show the basic read and write bus cycles (also shown in Basic Read Bus Cycle and Bus Cycle Sizing—Word Transfer, 16-bit Device, No Wait States) with the default of no wait states respectively.

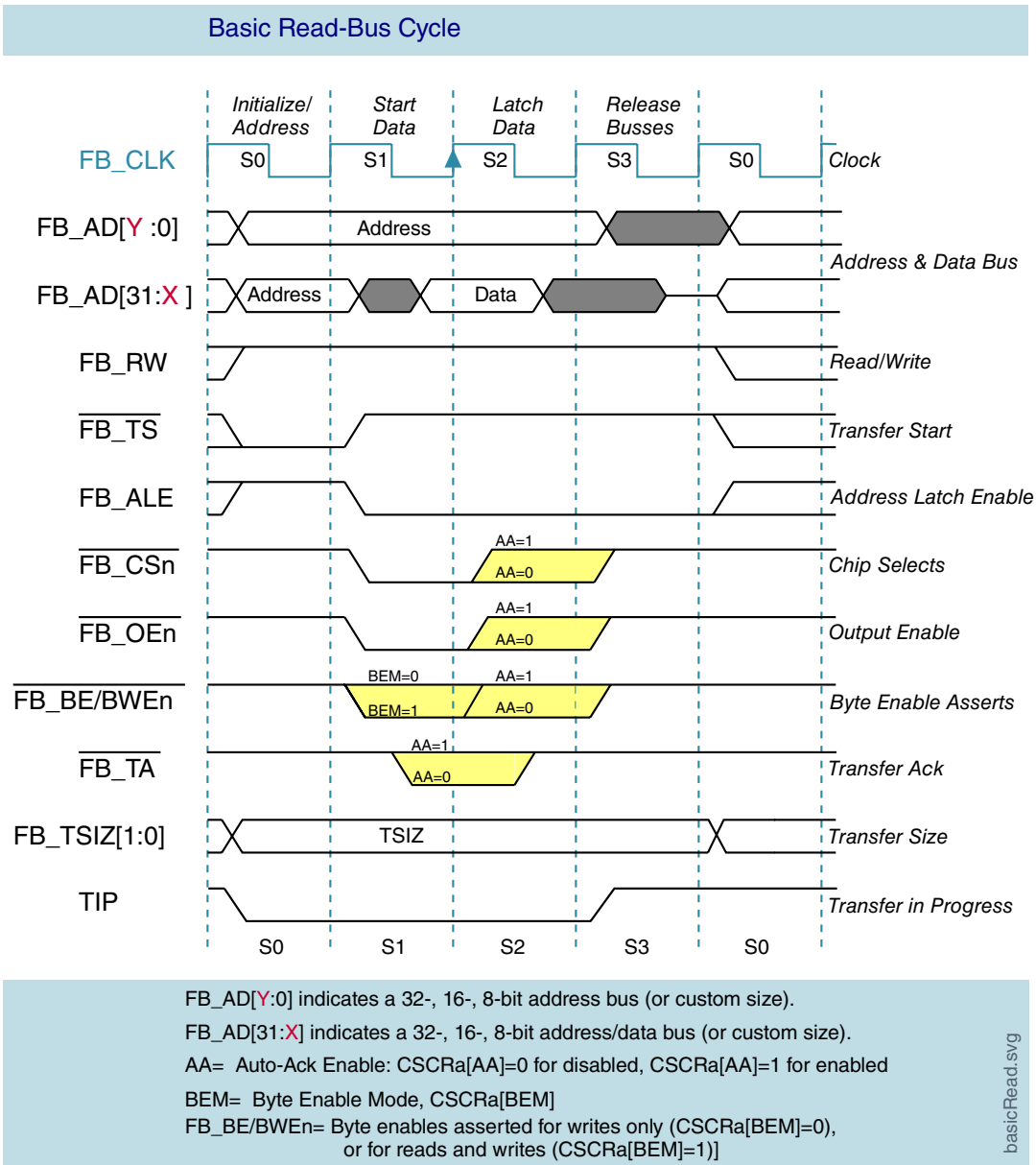



**Figure 20-18. Basic Read-Bus Cycle (No Wait States)**

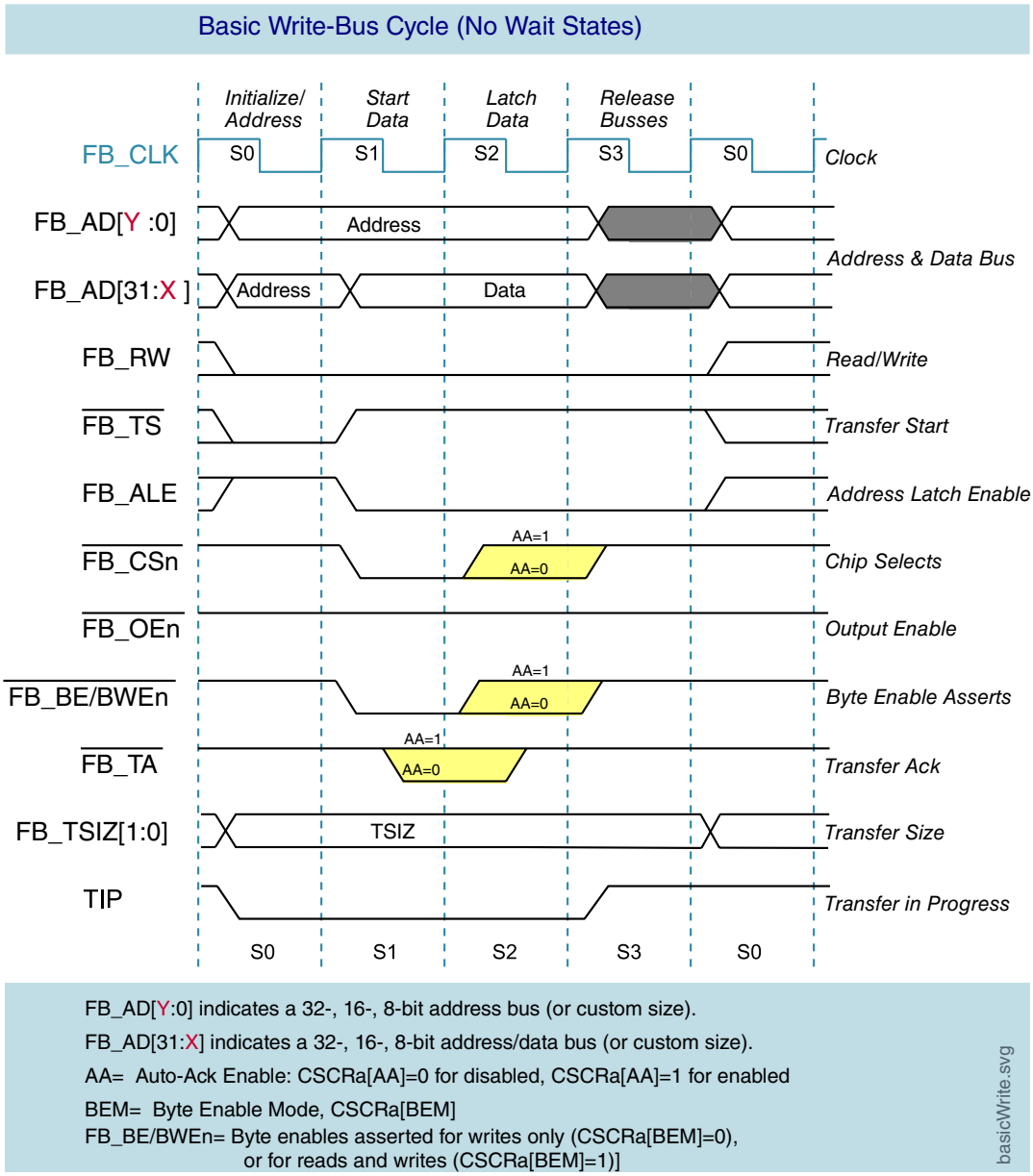
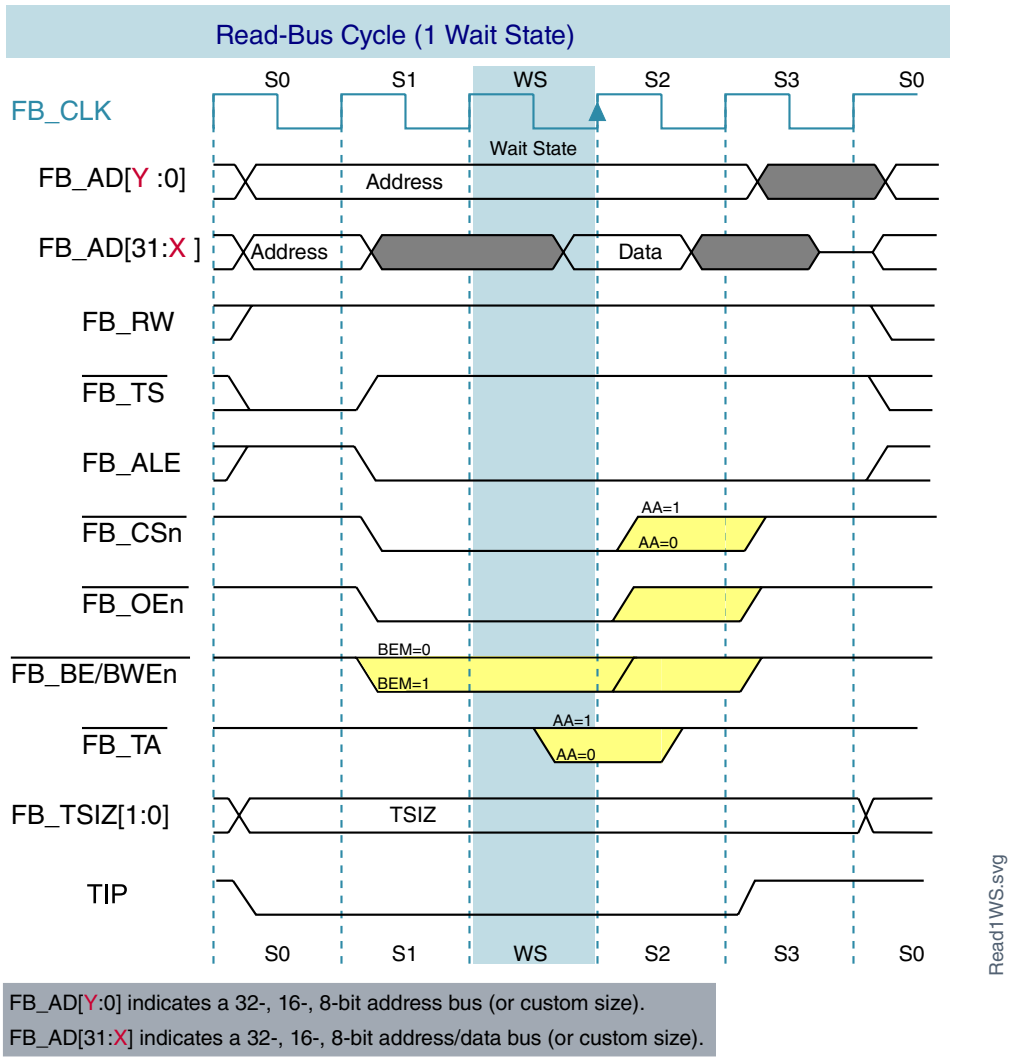



**Figure 20-19. Basic Write-Bus Cycle (No Wait States)**

If wait states are used, the S1 state repeats continuously until the chip-select auto-acknowledge unit asserts internal transfer acknowledge or the external FB_TA_b is recognized as asserted. The following figures show a read and write cycle with one wait state respectively.




FB_AD[Y:0] indicates a 32-, 16-, 8-bit address bus (or custom size).
FB_AD[31:X] indicates a 32-, 16-, 8-bit address/data bus (or custom size).

**Figure 20-20. Read-Bus Cycle (One Wait State)**

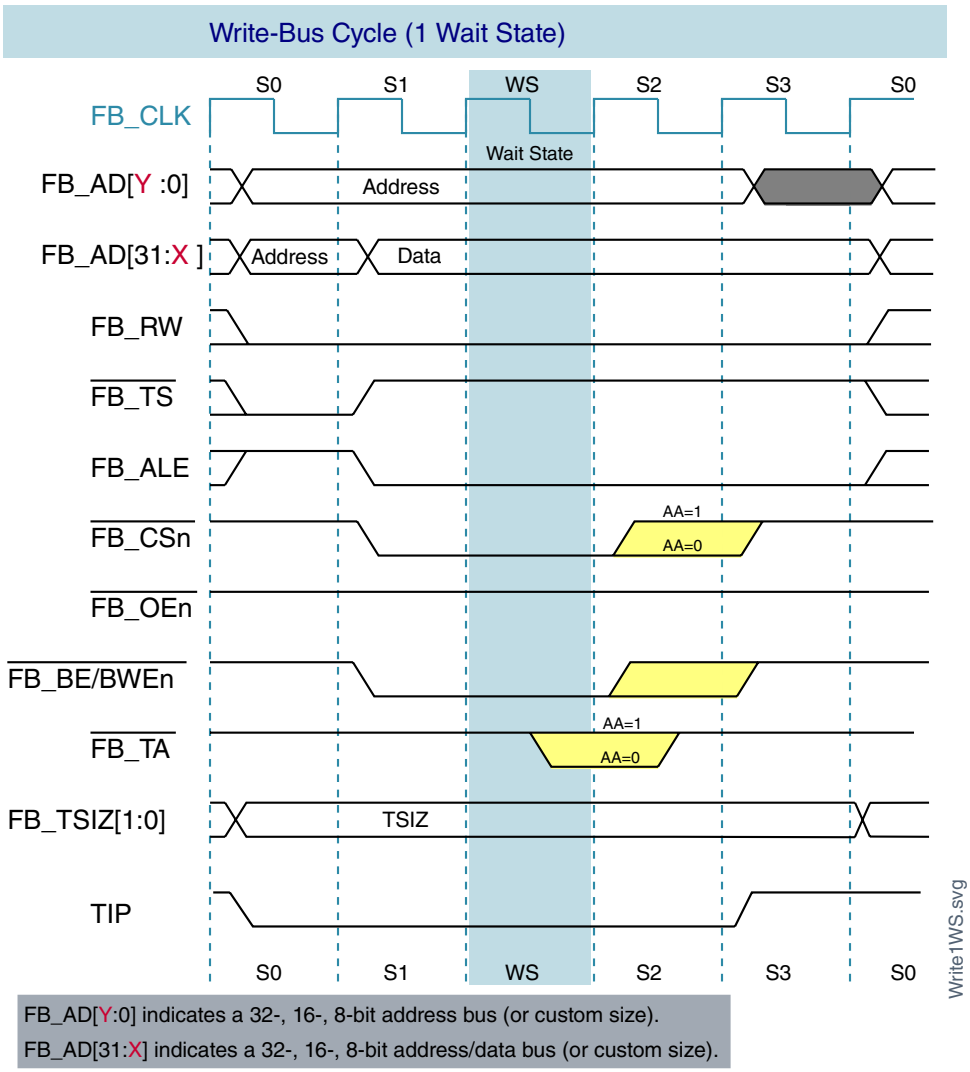



**Figure 20-21. Write-Bus Cycle (One Wait State)**

### 20.5.12.4.2 Address Setup and Hold

The timing of the assertion and negation of the chip selects, byte selects, and output enable can be programmed on a chip-select basis. Each chip-select can be programmed to assert one to four clocks after transfer start/address-latch enable (FB_TS_b/FB_ALE) is asserted. The following figures show read- and write-bus cycles with two clocks of address setup respectively.

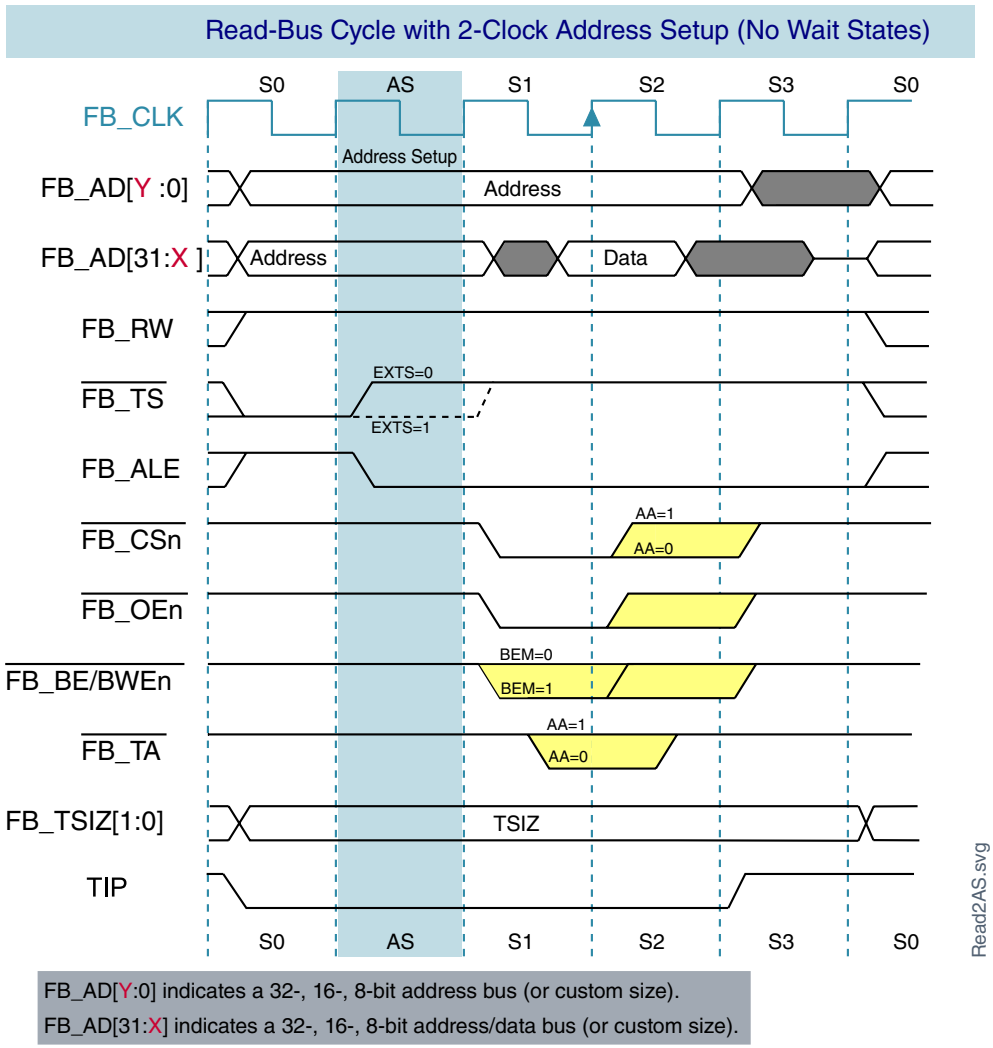



**Figure 20-22. Read-Bus Cycle with Two-Clock Address Setup (No Wait States)**

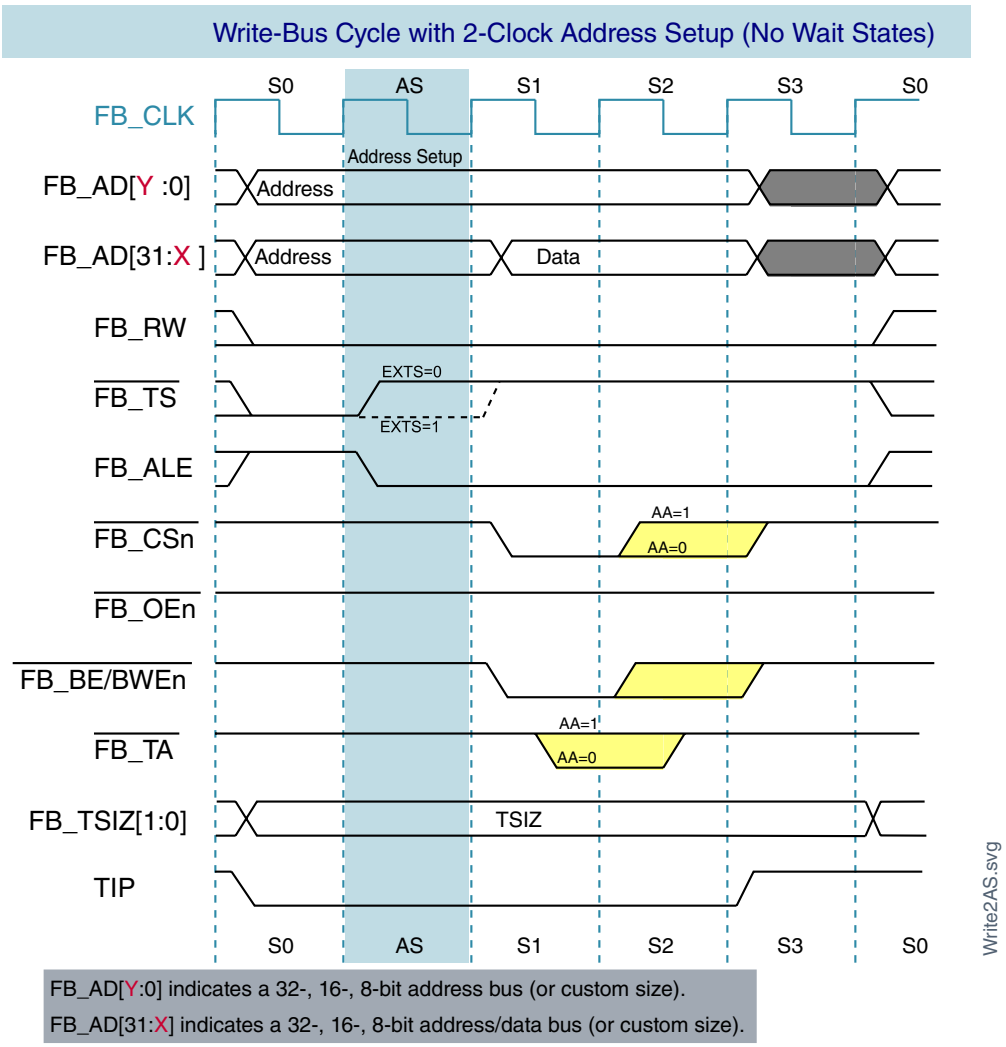



**Figure 20-23. Write-Bus Cycle with Two Clock Address Setup (No Wait States)**

In addition to address setup, a programmable address hold option for each chip select exists. Address and attributes can be held one to four clocks after chip-select, byte-selects, and output-enable negate. The following figures show read and write bus cycles with two clocks of address hold respectively.

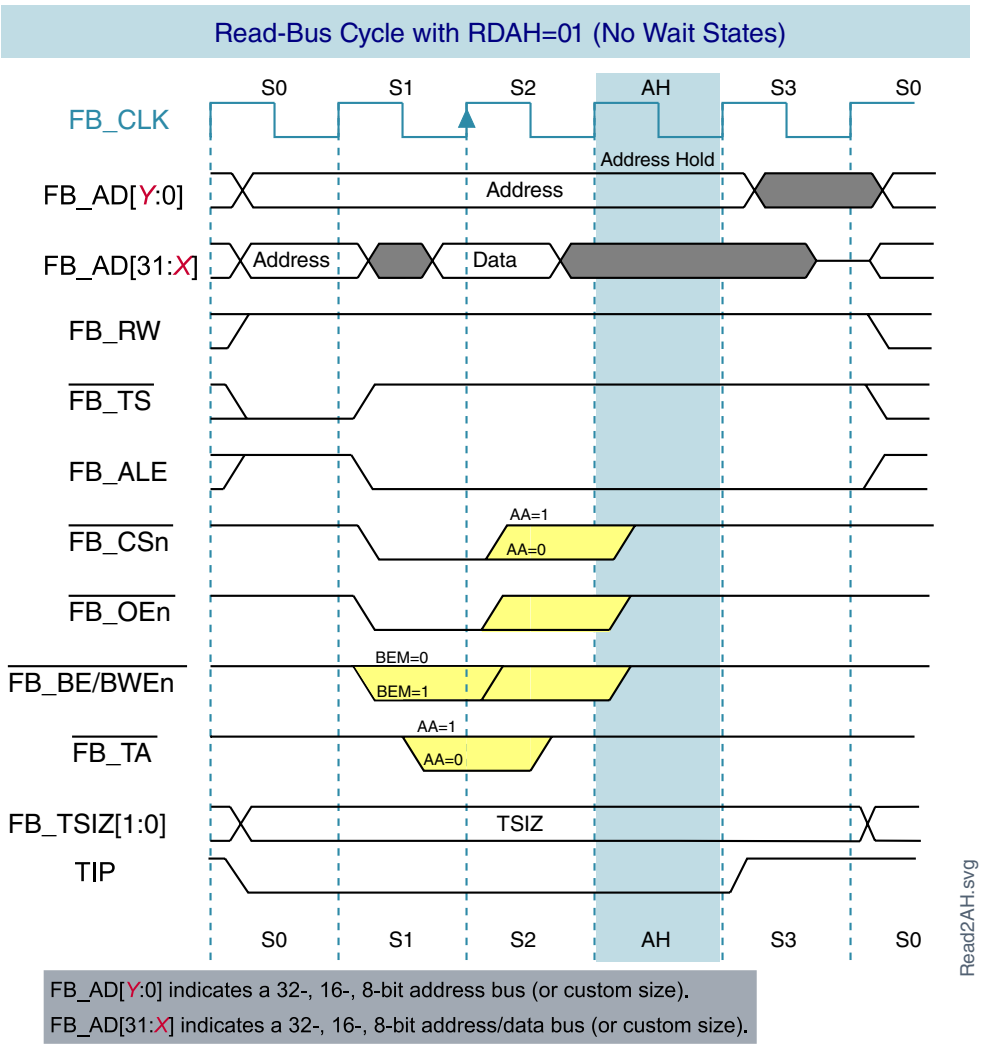



**Figure 20-24. Read Cycle with Two-Clock Address Hold (No Wait States)**

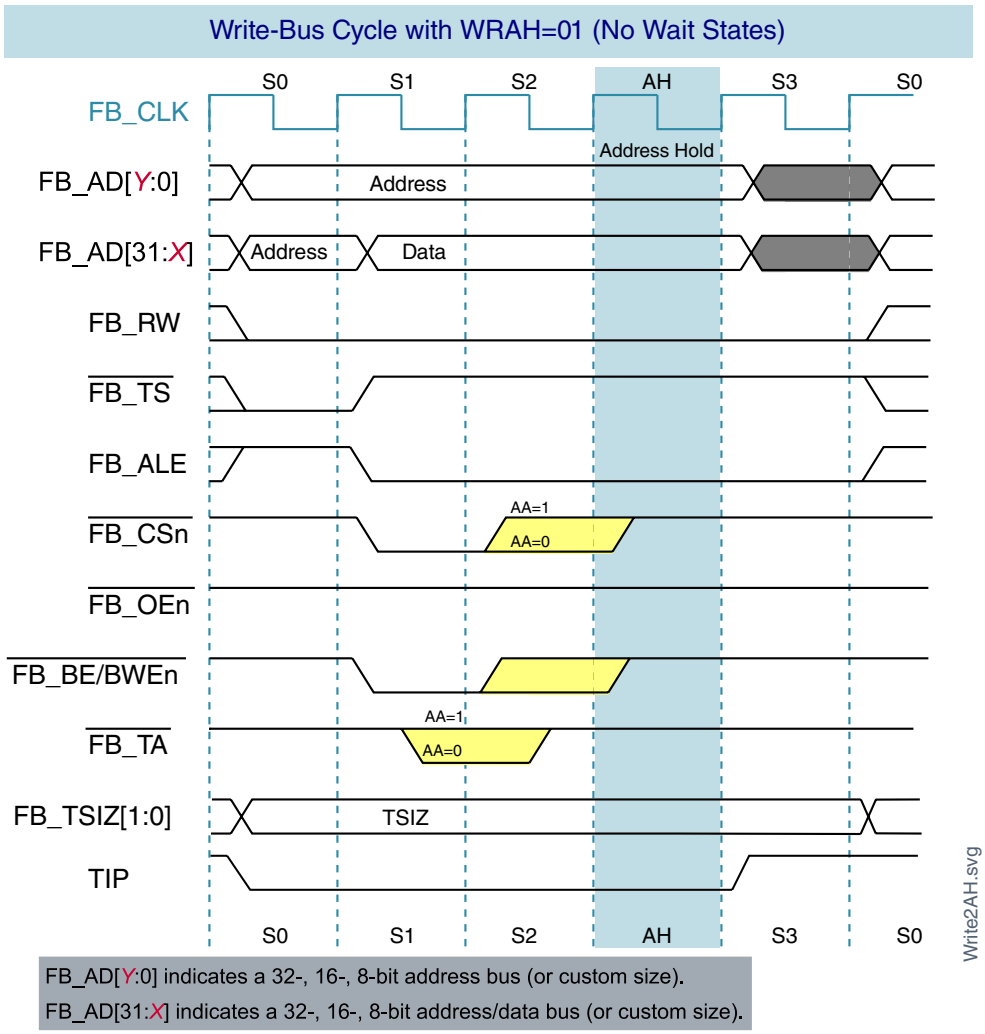



**Figure 20-25. Write Cycle with Two-Clock Address Hold (No Wait States)**

The following figure shows a bus cycle using address setup, wait states, and address hold.

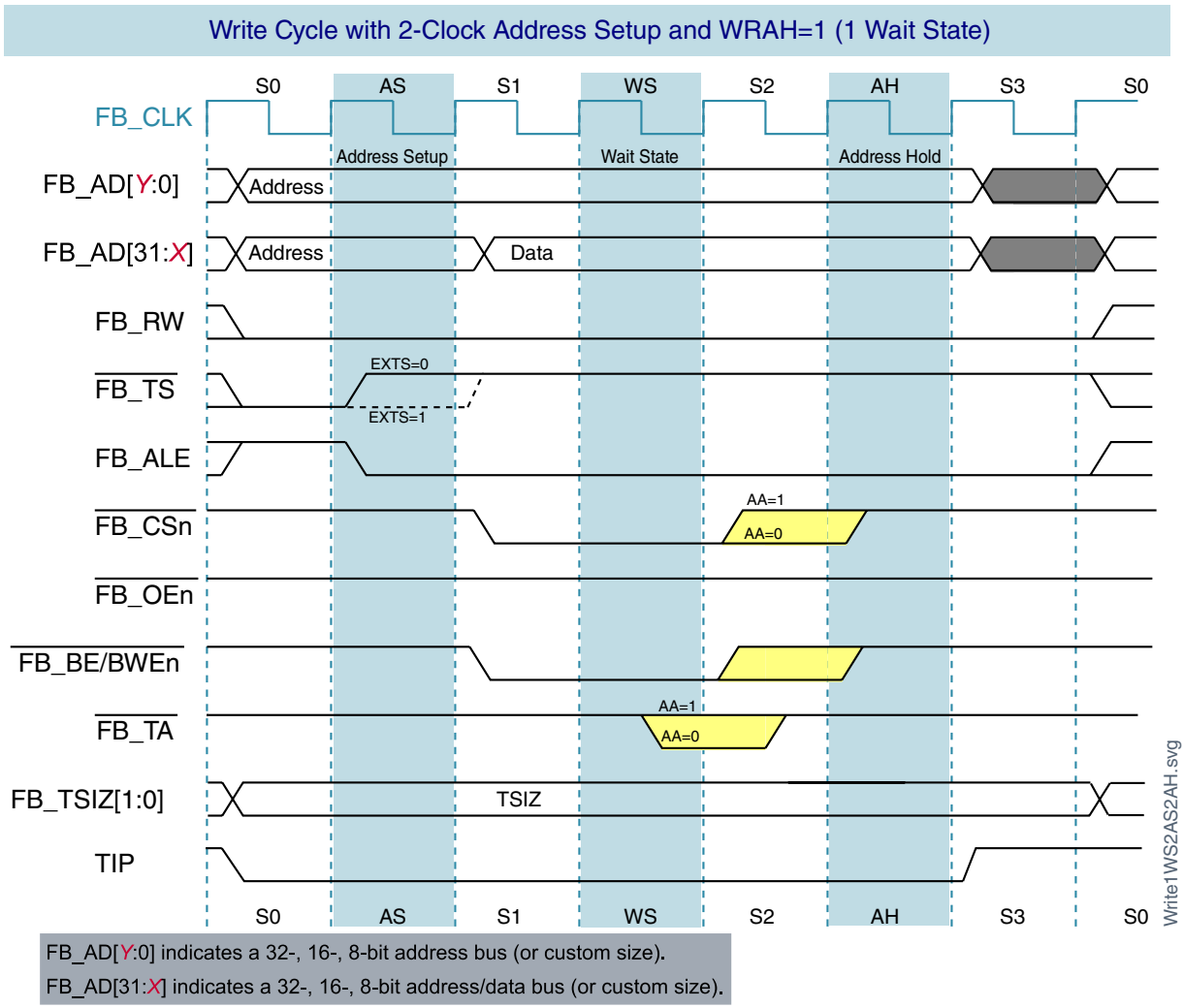



**Figure 20-26. Write Cycle with Two-Clock Address Setup and Two-Clock Hold (One Wait State)**

## 20.5.13 Burst cycles

The chip can be programmed to initiate burst cycles if its transfer size exceeds the port size of the selected destination. The initiation of a burst cycle is encoded on the transfer size pins (FB_TSIZ[1:0]). For burst transfers to smaller port sizes, FB_TSIZ[1:0] indicates the size of the entire transfer. For example, with bursting enabled, a 16-bit transfer to an 8-bit port takes two beats (two byte-sized transfers), for which FB_TSIZ[1:0] equals 10b throughout. A 32-bit transfer to an 8-bit port takes four beats (four byte-sized transfers), for which FB_TSIZ[1:0] equals 00b throughout.

### 20.5.13.1 Enabling and inhibiting burst

The CSCRn registers enable bursting for reads, writes, or both.

Memory spaces can be declared burst-inhibited for reads and writes by writing 0b to the appropriate CSCRn[BSTR] and CSCRn[BSTW] fields.

### 20.5.13.2 Transfer size and port size translation

With bursting disabled, any transfer larger than the port size breaks into multiple individual transfers (e.g. <Addr><Data><Addr+1><Data><Addr+2><Data>). With bursting enabled, any transfer larger than the port size results in a burst cycle of multiple beats (e.g. <Addr><Data><Data><Data>). The following table shows the result of such transfer translations.

| Port size PS[1:0] | Transfer size FB_TSIZ[1:0] | Burst-inhibited: Number of transfers<br>Burst enabled: Number of beats |
|---|---|---|
| 01b (8 bit) | 10b (16 bits) | 2 |
| | 00b (32 bits) | 4 |
| | 11b (16 bytes) | 16 |
| 1Xb (16 bit) | 00b (32 bits) | 2 |
| | 11b (16 bytes) | 8 |
| 00b (32 bit) | 11b (line) | 4 |

The FlexBus can support X-1-1-1 burst cycles to maximize system performance, where X is the primary number of wait states (max 63). Delaying termination of the cycle can add wait states. If internal termination is used, different wait state counters can be used for the first access and the following beats.

### 20.5.13.3 32-bit-Read burst from 8-Bit port 2-1-1-1 (no wait states)

The following figure shows a 32-bit read to an 8-bit external chip programmed for burst enable. The transfer results in a 4-beat burst and the data is driven on FB_AD[31:24]. The transfer size is driven at 32-bit (00b) throughout the bus cycle.

**Note**

- In **non-multiplexed address/data mode:** the address on FB_A increments only during internally-terminated burst

cycles. The first address is driven throughout the entire burst for externally-terminated cycles.

- In **multiplexed address/data mode:** the address is driven on FB_AD only during the first cycle for all terminated cycles. In other words, the lower address lines only increment during internally-terminated burst cycles (assuming that the lower address lines are being driven with the address during the data phase).

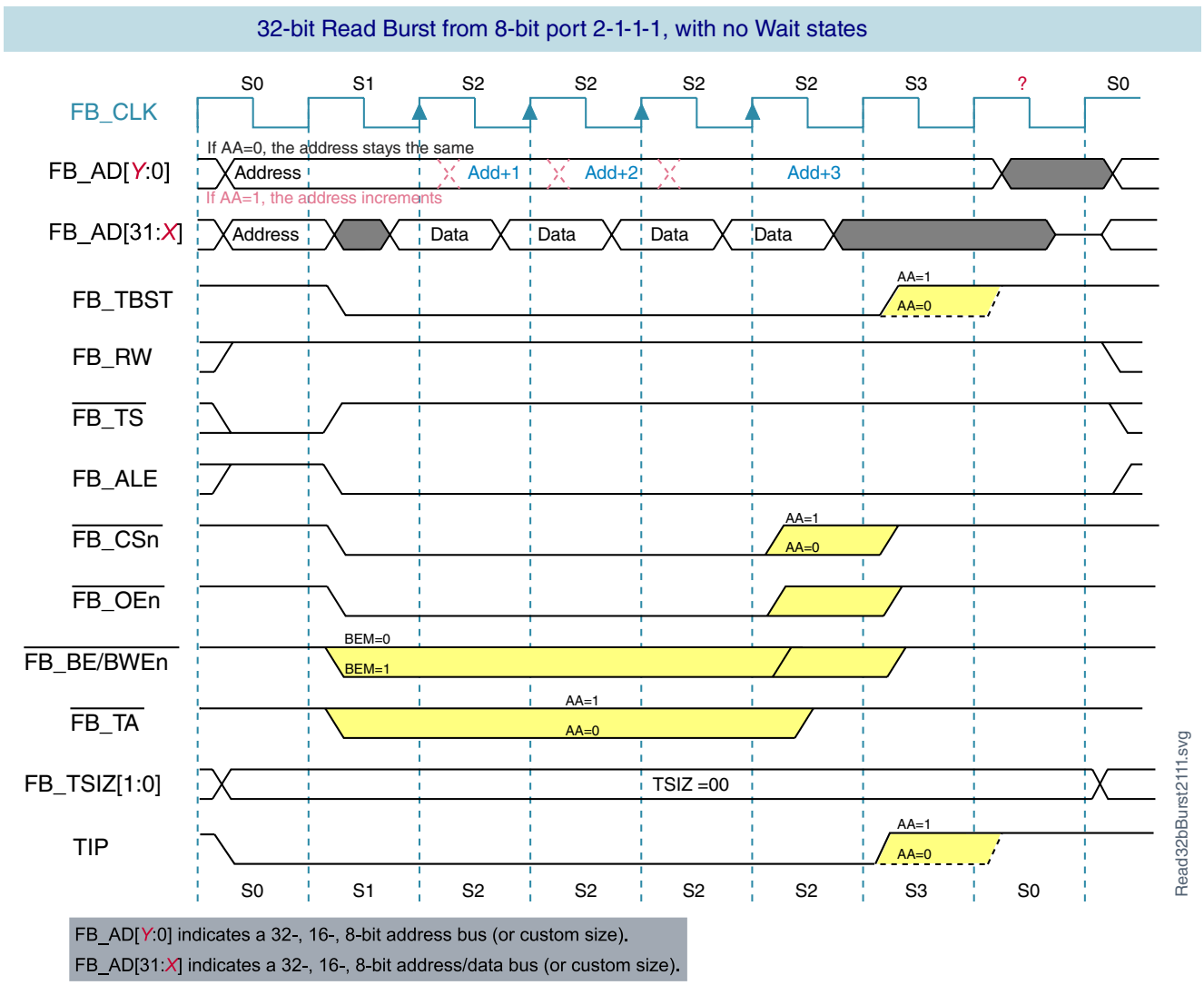



**Figure 20-27. 32-bit-Read burst from 8-Bit port 2-1-1-1 (no wait states)**

### 20.5.13.4 32-bit-Write burst to 8-Bit port 3-1-1-1 (no wait states)

The following figure shows a 32-bit write to an 8-bit external chip with burst enabled. The transfer results in a 4-beat burst and the data is driven on FB_AD[31:24]. The transfer size is driven at 32-bit (00b) throughout the bus cycle.

**Note**

The first beat of any write burst cycle has at least one wait state. If the bus cycle is programmed for zero wait states (CSCRn[WS] = 0b), one wait state is added. Otherwise, the programmed number of wait states are used.

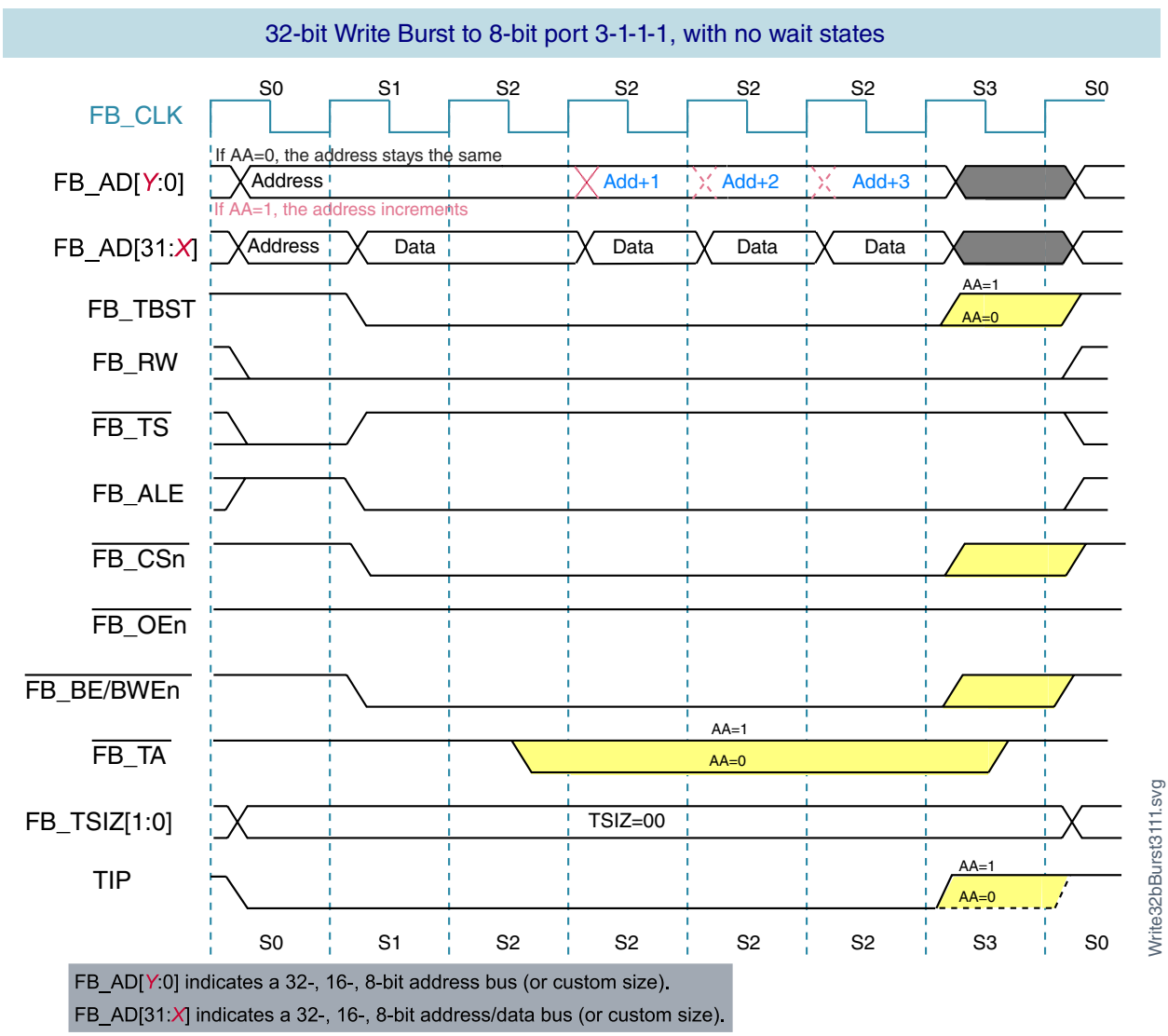


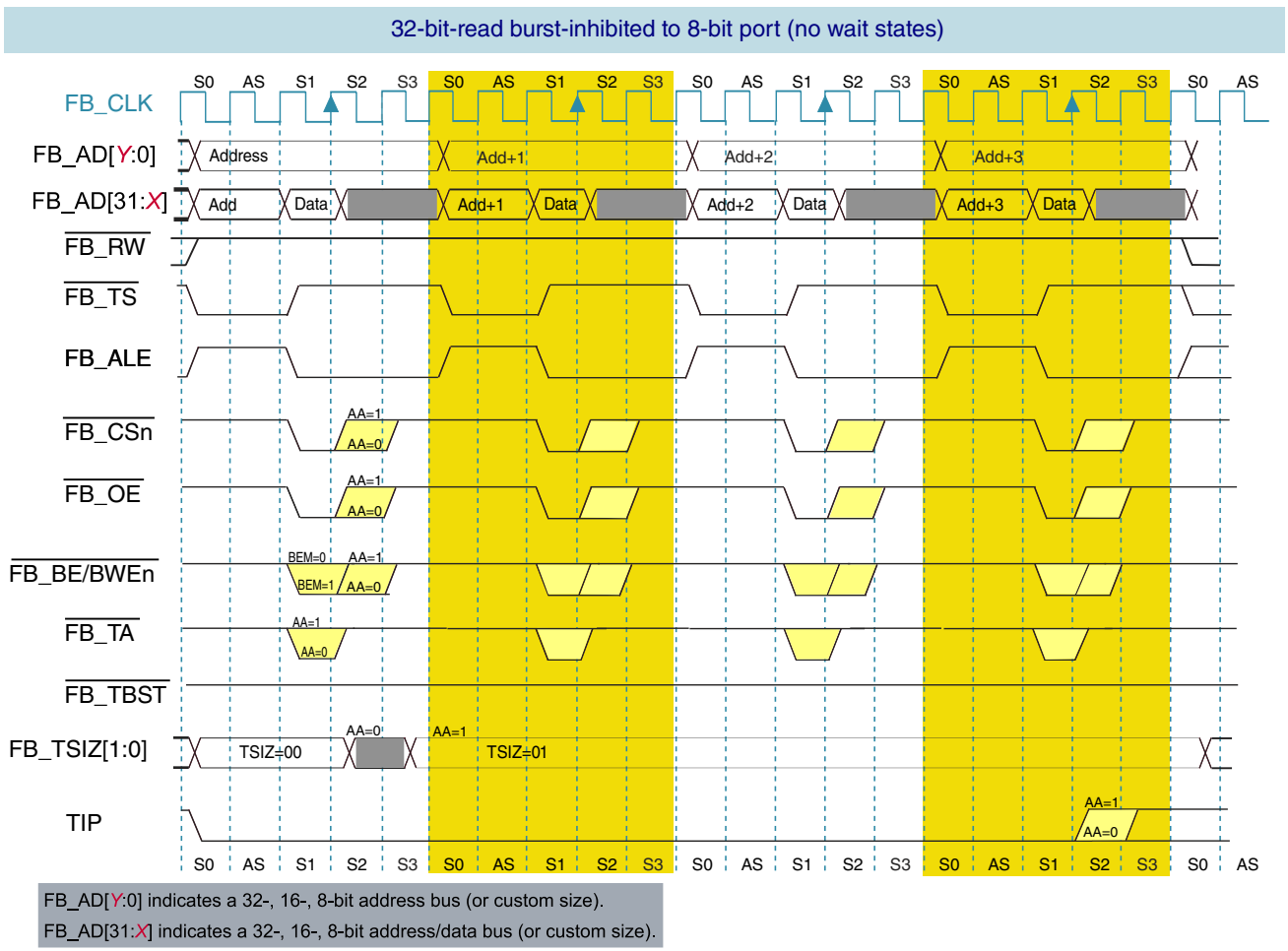

**Figure 20-28. 32-bit-Write burst to 8-Bit port 3-1-1-1 (no wait states)**

### 20.5.13.5 32-bit-read burst-inhibited from 8-bit port (no wait states)

The following figure shows a 32-bit read from an 8-bit device with burst inhibited. The transfer results in four individual transfers. The transfer size is driven at 32-bit (00b) during the first transfer and at byte (01b) during the next three transfers.

**Note**

There is an extra clock of address setup (AS) for each burst-inhibited transfer between states S0 and S1.




### 20.5.13.6 32-bit-write burst-inhibited to 8-bit port (no wait states)

The following figure shows a 32-bit write to an 8-bit device with burst inhibited. The transfer results in four individual transfers. The transfer size is driven at 32-bit (00b) during the first transfer and at byte (01b) during the next three transfers.

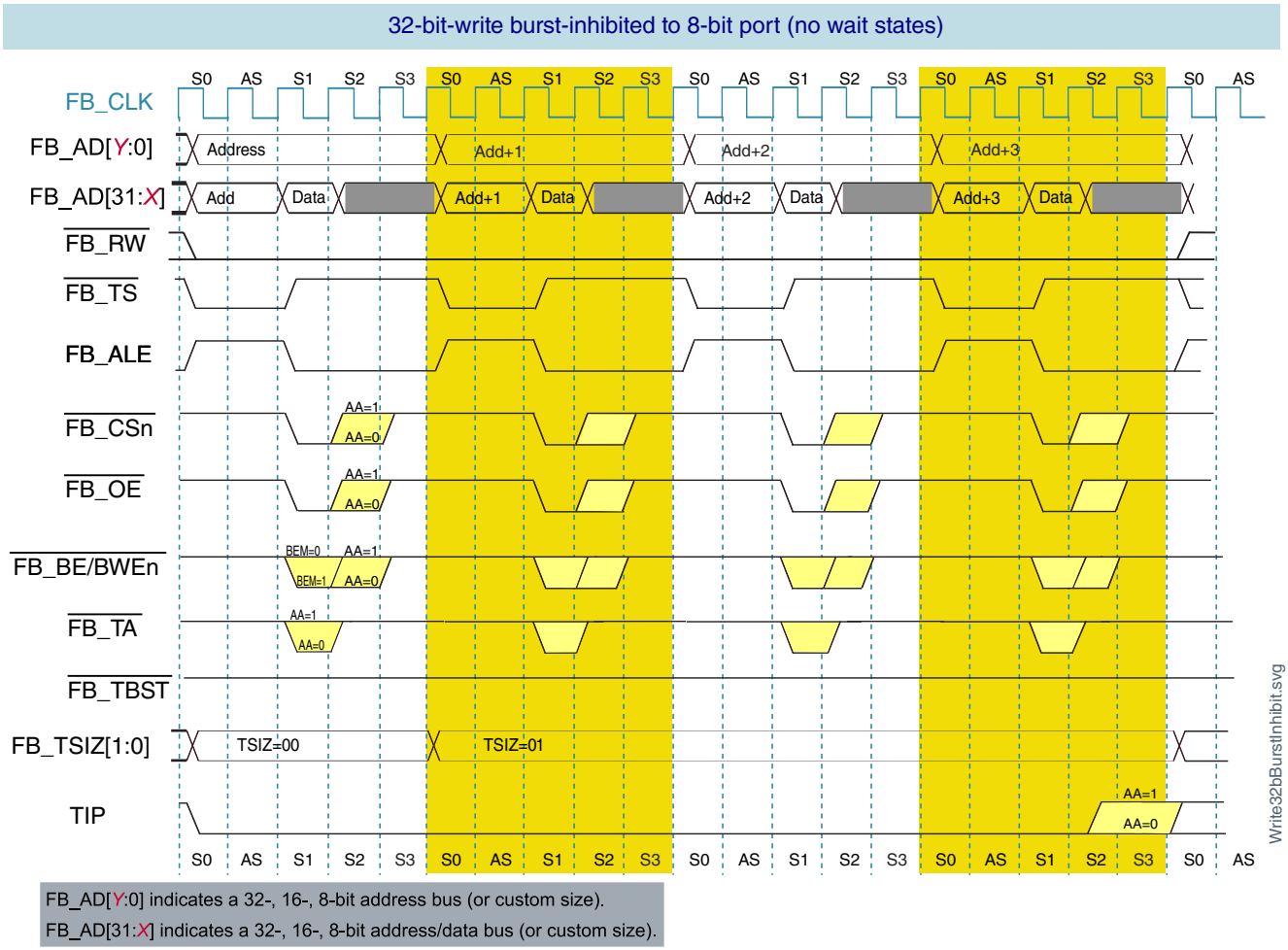



**Figure 20-29. 32-bit-write burst-inhibited to 8-bit port (no wait states)**

## 20.5.13.7 32-bit-read burst from 8-bit port 3-2-2-2 (one wait state)

The following figure illustrates another read burst transfer, but in this case a wait state is added between individual beats.

### Note

CSCRn[WS] determines the number of wait states in the first beat. However, for subsequent beats, the CSCRn[WS] (or CSCRn[SWS] if CSCRn[SWSEN] = 1b) determines the number of wait states.

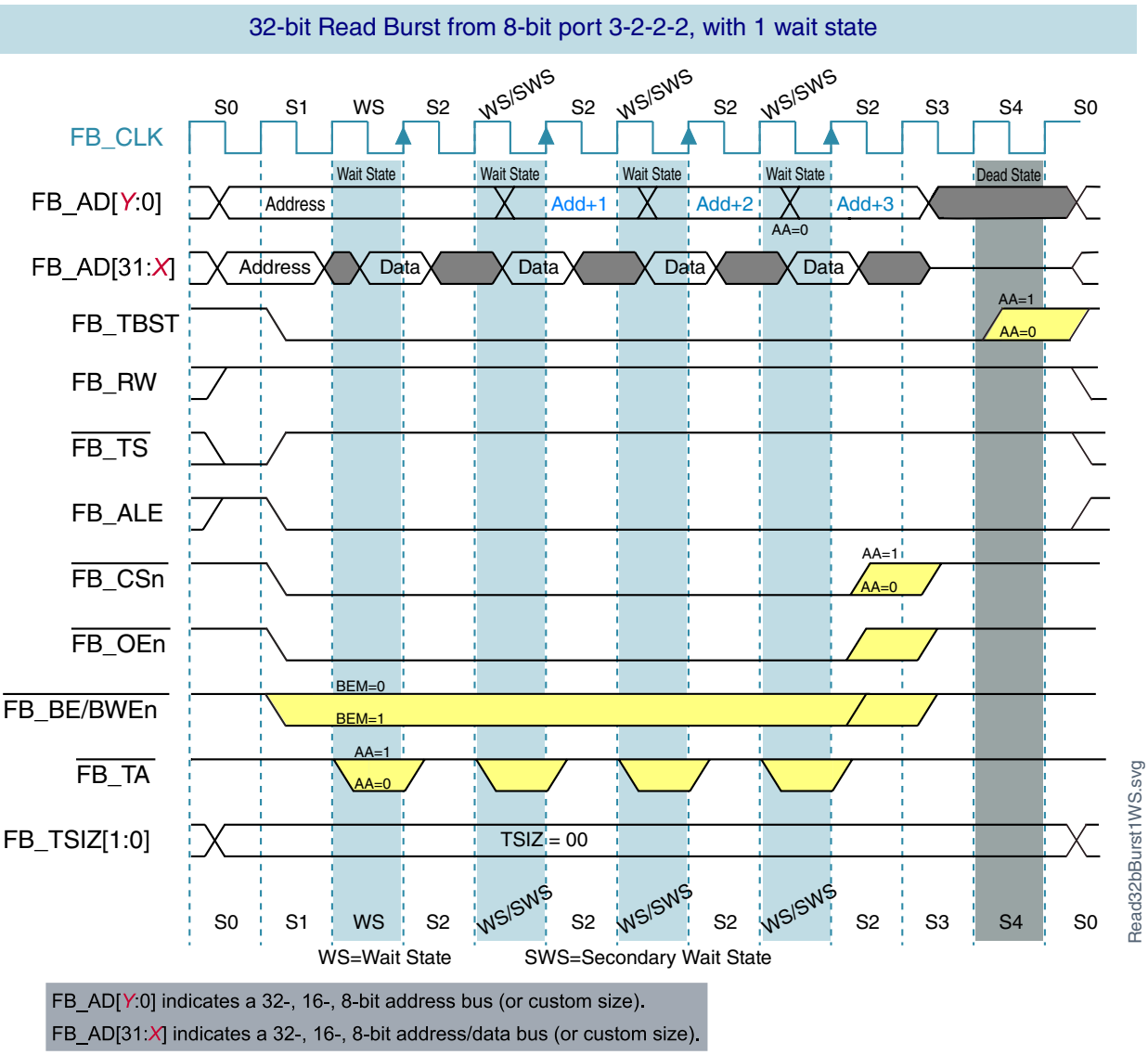



**Figure 20-30. 32-bit-read burst from 8-bit port 3-2-2-2 (one wait state)**

## 20.5.13.8 32-bit-write burst to 8-bit port 3-2-2-2 (one wait state)

The following figure illustrates a write burst transfer with one wait state.

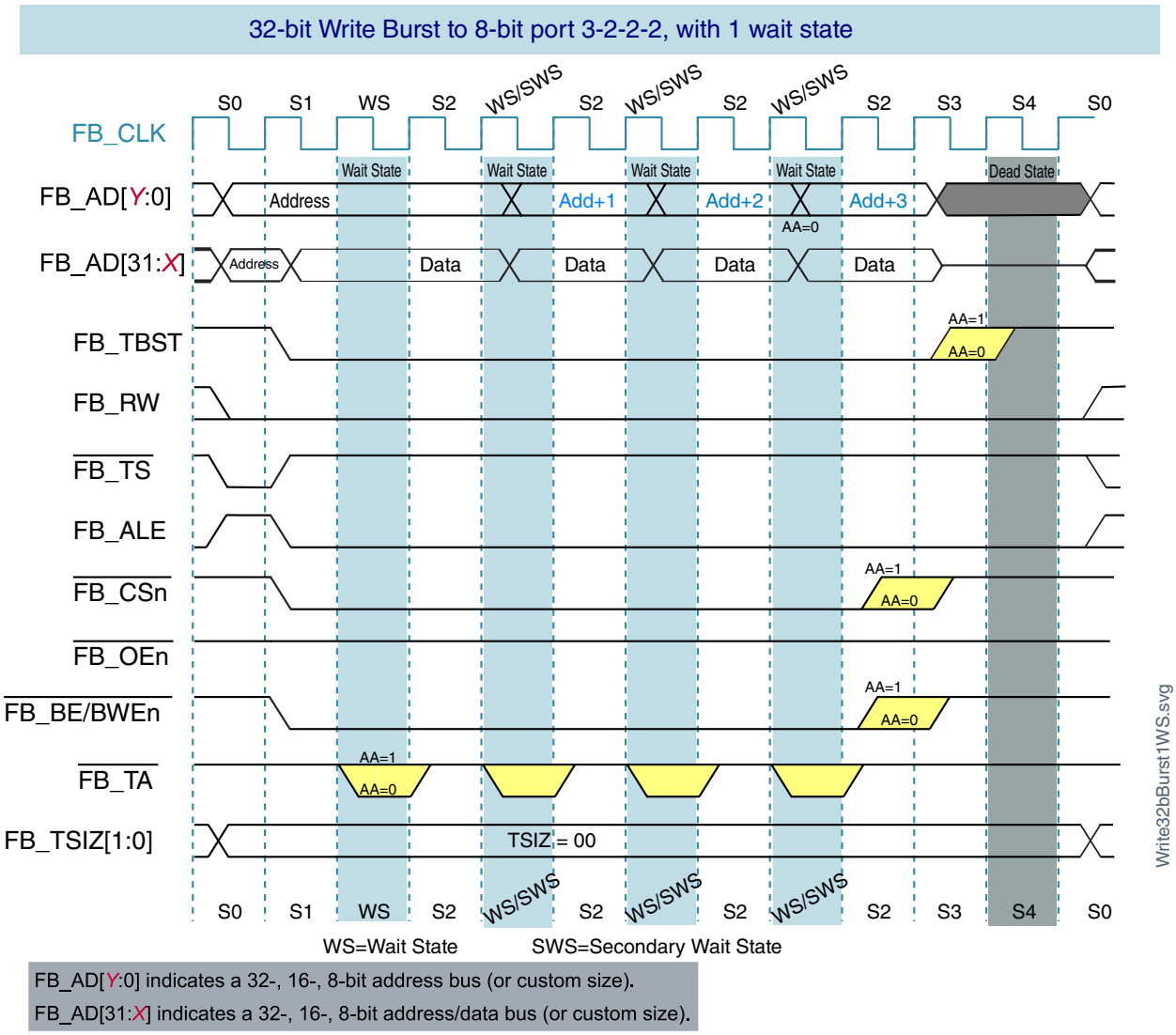



**Figure 20-31. 32-bit-write burst to 8-bit port 3-2-2-2 (one wait state)**

## 20.5.13.9 32-bit-read burst from 8-bit port 3-1-1-1 (address setup and hold)

If address setup and hold are used, only the first and last beat of the burst cycle are affected. The following figure shows a read cycle with one clock of address setup and address hold.

### Note

- In **non-multiplexed address/data mode:** the address on FB_A increments only during internally-terminated burst cycles (CSCRn[AA] = 1b). The attached device must be able to account for this, or a wait state must be added. The

first address is driven throughout the entire burst for externally-terminated cycles.

- In **multiplexed address/data mode:** the address is driven on FB_AD only during the first cycle for internally- and externally-terminated cycles.

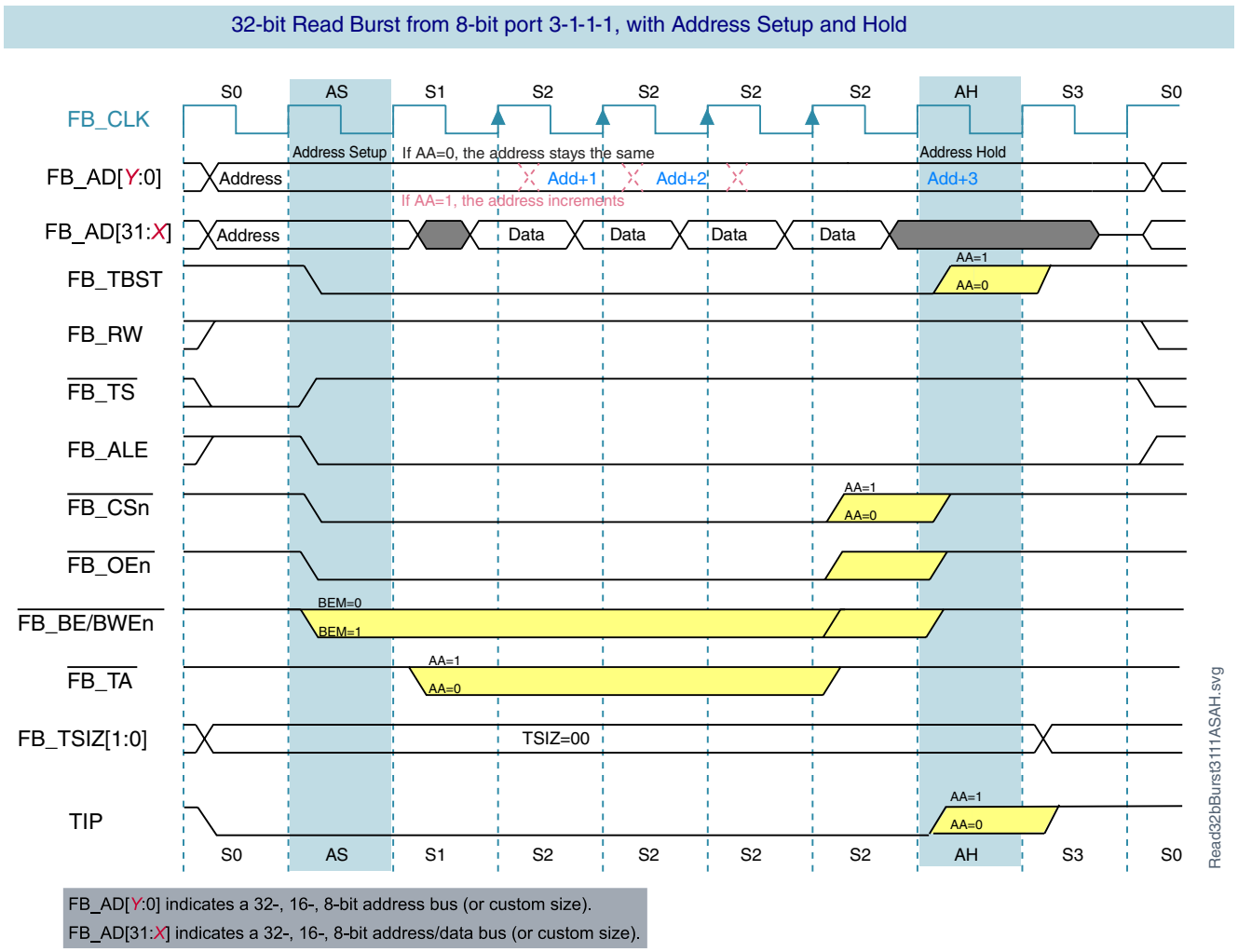



**Figure 20-32. 32-bit-read burst from 8-bit port 3-1-1-1 (address setup and hold)**

## 20.5.13.10 32-bit-write burst to 8-bit port 3-1-1-1 (address setup and hold)

The following figure shows a write cycle with one clock of address setup and address hold.

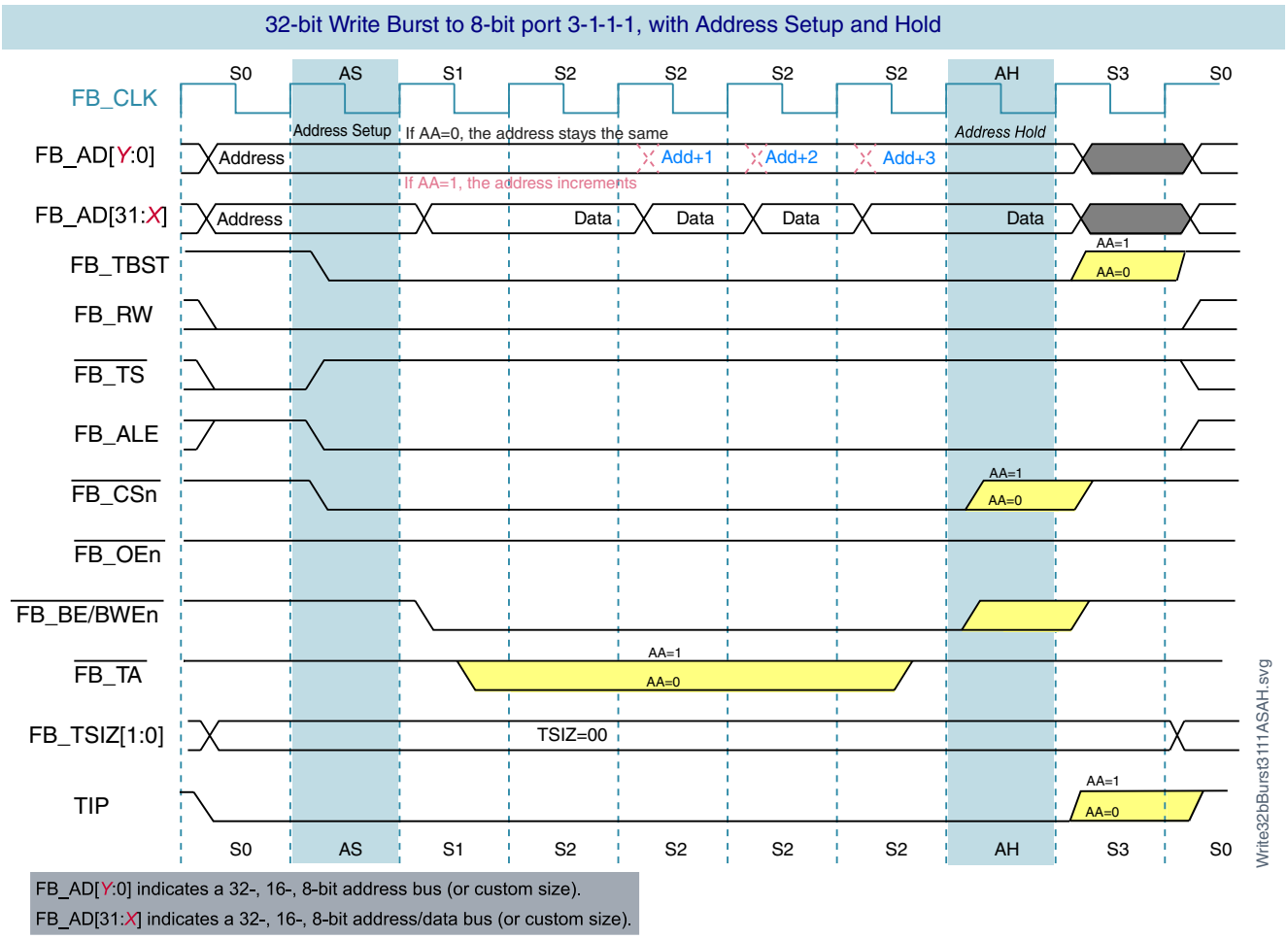



**Figure 20-33. 32-bit-write burst to 8-bit port 3-1-1-1 (address setup and hold)**

## 20.5.14 Misaligned operands

A data operand can reside in memory at any address; however, the transfer requires additional bus cycles if the data operand is misaligned. An operand is misaligned depending on the operand size and address as follows:

- A byte operand is never misaligned (it is aligned at any address)
- A 16-bit operand is misaligned at odd addresses
- A 32-bit operand is misaligned at any address that is not a multiple of four

### 20.5.14.1 Example: Aligned 32-bit transfer to a 32-bit port

This table shows an example of an *32-bit aligned* operand transfer from a longword address (or a byte address) to a 32-bit port. This example assumes CSCR[BLS] = 0.

| Transfer | Description | FB_D | | | | FB_A2–FB_A0 |
|---|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 | |
| 1 | Byte 0, Byte 1, Byte 2, Byte 3 all transfer at an offset of 0h. | Byte 0 | Byte 1 | Byte 2 | Byte 3 | 001 |

### 20.5.14.2 Example: Misaligned 32-bit transfer to a 32-bit port

This table shows an example of a misaligned 32-bit access from a longword address (or a byte address) to a 32-bit port. This example assumes CSCR[BLS] = 0.

| Transfer | Description | FB_D | | | | FB_A2–FB_A0 |
|---|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 | |
| 1 | Byte 0 transfers at an offset of 1h. | — | Byte 0 | — | — | 001 |
| 2 | A word (Byte 1 and Byte 2) transfers with an offset of 2h. | — | — | Byte 1 | Byte 2 | 010 |
| 3 | Byte 3 transfers with an offset of 0h and the transfer is complete. | Byte 3 | — | — | — | 100 |

FlexBus transfer sizes are 8, 16, or 32 bits (never 24 bits), which is why it needs to break this example into 3 transfers (8-bit, 16-bit, 8-bit), instead of 2 transfers (8-bit, 24-bit).

### 20.5.14.3 Example: Misaligned 16-bit transfer to a 32-bit port

This table shows an example of a misaligned 16-bit operand transfer from a byte address to a 32-bit port. This example assumes CSCR[BLS] = 0.

| Transfer | Description | FB_D | | | | FB_A2–FB_A0 |
|---|---|---|---|---|---|---|
| | | 31–24 | 23–16 | 15–8 | 7–0 | |
| 1 | Byte 0 transfers at an offset of 3h. | — | — | — | Byte 0 | 001 |
| 2 | Byte 1 transfers with an offset of 0h. | Byte 1 | — | — | — | 100 |

## 20.5.15 Extended Transfer Start/Address Latch Enable

The FB_TS_b/FB_ALE signal indicates that a bus transaction has begun and the address and attributes are valid. By default, the FB_TS_b/FB_ALE signal asserts for a single bus clock cycle. When CSCR*n*[EXTS] is set, the FB_TS_b/FB_ALE signal asserts and remain asserted until the first positive clock edge after FB_CS_b*n* asserts. See the following figure.

**NOTE**

When EXTS is set, CSCR*n*[WS] must be programmed to have at least one primary wait state.

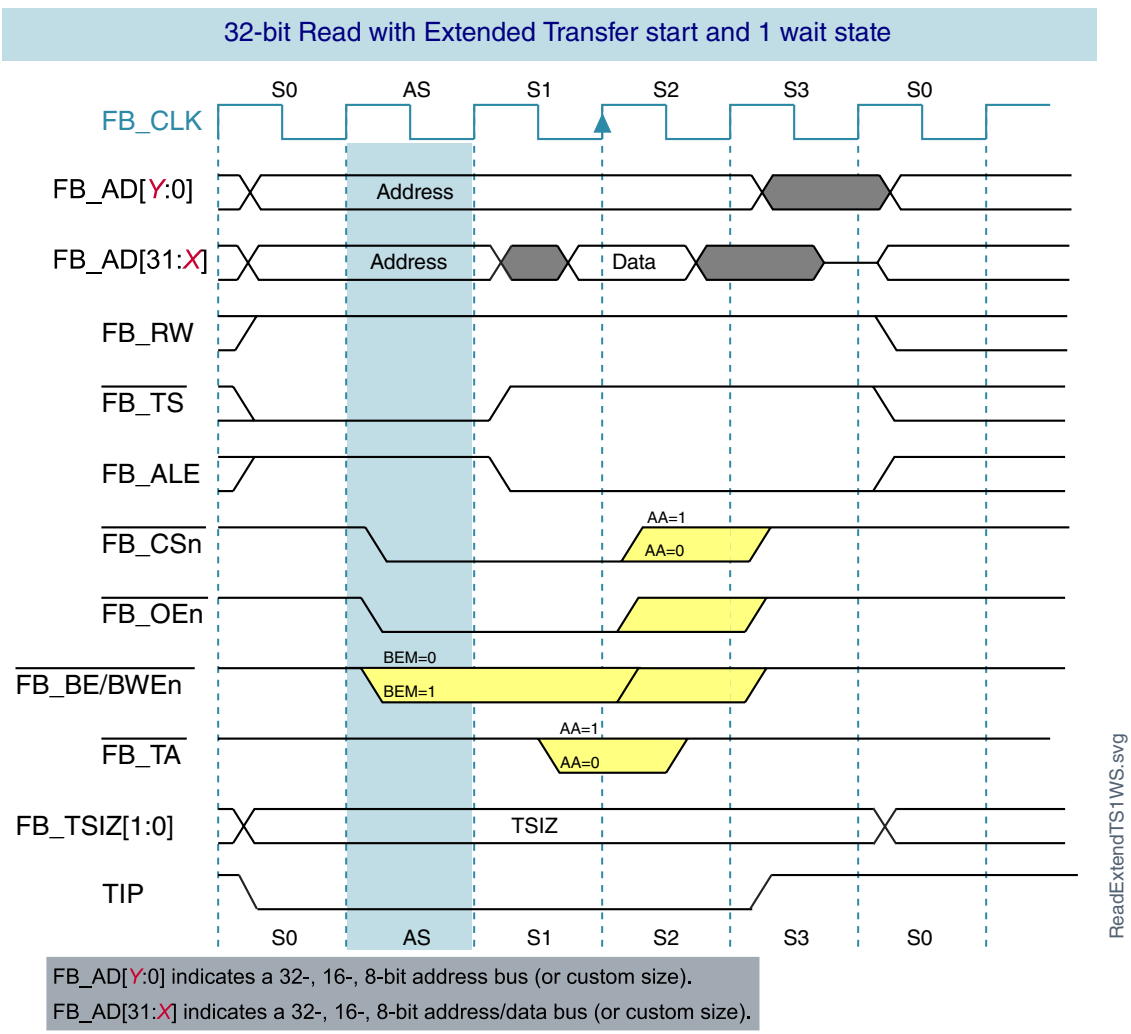



**Figure 20-34. Read-Bus Cycle with CSCR*n*[EXTS] = 1 (One Wait State)**

### 20.5.16 Bus errors

These types of accesses cause a transfer to terminate with a bus error:

- Any access when the clock to the module is disabled
- A write to a write-protected address range
- An access whose address is not in a range covered by a chip-select
- An access whose address is in a range covered by more than one chip-selects
- A write to a reserved address in the memory map
- A write to a reserved field in the CSPMCR
- Any FlexBus accesses when FlexBus is secure

If the auto-acknowledge feature is disabled (CSCR[AA] is 0) for an address that generates an error, then the transfer can be terminated by asserting FB_TA_b. If the processor must manage a bus error differently, then asserting an interrupt to the core along with FB_TA when the bus error occurs can invoke an interrupt handler.

The device can hang if FlexBus is configured for external termination and the CSPMCR is not configured for FB_TA.

## 20.6 Initialization/Application Information

### 20.6.1 Initializing a chip-select

Before using any other chip select, chip select 0 (CS0) should be initialized, to take it out of global chip select mode.

To initialize a chip-select:

1. Write to the associated CSAR.
2. Write to the associated CSCR.
3. Write to the associated CSMR, including writing "1b" to the Valid field (CSMRn[V]).

### 20.6.2 Reconfiguring a chip-select

To reconfigure a previously-used chip-select:

1. Invalidate the chip-select by writing 0b to the associated CSMR's Valid field (CSMRn[V]).
2. Write to the associated CSAR.

3. Write to the associated CSCR.
4. Write to the associated CSMR, including writing "1b" to the Valid field (CSMRn[V]).

# Chapter 21
# AHB Low Power Cache Controller (AHB_LPCAC)

## 21.1 Chip-specific LPCAC information

### 21.1.1 Cache controls

LPCAC is the cache for CM4 or RI5CY core. MCM0_CPCR2 register is used to clear and disable the LPCAC.

## 21.2 Introduction

The AHB Low Power Cache Controller (LPCAC) is a processor-local Level 1 (L1) bus cache (lpcac) controller for use with cores using AMBA-AHB input/output buses. It is optimized for minimum RunIDD (dynamic power).

### 21.2.1 Overview

The LPCAC has a 32-bit datapath AMBA-AHB input bus and a 32-bit datapath AMBA-AHB output bus. Internally, LPCAC decodes a cacheable region (with the cache region writethrough).

Any input AHB accesses with addresses that:

- hit the cache region, use a portion of the attached RAM as a cache; cache misses use the AHB output bus to fetch the miss data.
- miss the cache region, go to the AHB output bus.

## 21.3 Signal descriptions

The LPCAC has no external signals.

## 21.4 Memory map and Input Control description

### 21.4.1 Memory regions

The LPCAC decodes cacheable address regions.

- The cache memory region may be composed of one or more disjoint memory regions and the total size may be larger than the cache storage.

Any address not in the cacheable memory regions is in the pass-through memory region. All accesses to the pass-through memory regions are non-cacheable. For all access to the cacheable memory regions, the accesses is cacheable if the attributes of the access indicate it can be cached, else it is non-cacheable.

See the System Memory Map for cache memory range information.

### 21.4.2 Input controls

The LPCAC does not have a module-resident programming model. An external block supplies the needed LPCAC control inputs.

**Table 21-1. Control inputs for LPCAC**

| Function | Control input | Description |
|---|---|---|
| clear lpcac | clr_lpcac | One cycle active-high pulse clears the lpcac. |
| disable lpcac | dis_lpcac | • 0 = lpcac enabled (reset configuration)<br>• 1 = lpcac disabled, all cacheable accesses pass from LPCAC input to output bus |
| ignore CACHE_MAP | mode_ctl_hprot | • 0 = use LPCAC Memory Regions to determine if a access to a given address may be cached. Then, if the attributes of the access indicate it can be cached, the access is cached.<br>• 1 = Use only the attributes of the access to indicate it can be cached. |

## 21.5 Functional description

This section provides further information on the LPCAC operation.

The LPCAC is cleared and enabled by a reset.

The LPCAC examines every valid AHB input access. If the access hits the cache memory region and the cache is enabled, the access goes to the cache portion of the LPCAC. If the access does not go to the cache, the access is passed directly to the AHB output bus.

The LPCAC cache has a 4 word (128-bit) cache line size:

- Byte, half-word, or word read cache misses do an aligned 4 word-size cache miss read to the LPCAC AHB output bus then load the needed data in the cache. The cache miss uses a 4 beat, 32-bit per beat, wrapped burst bus access to fetch the cache miss data.
- Read cache hits return the desired read data from the cache.
- Byte, half-word, or word write cache misses do the desired write operation only to the LPCAC AHB output bus (writethrough mode has a no cache allocation on write miss policy).
- Byte, half-word, or word write cache hits do the desired write operation to the cache and to the LPCAC AHB output bus.

### 21.5.1 Cache Functional description

The LPCAC cache is an 8-way, 2-set-associative write-through design. It supports total cache capacities of 4 KB for a 32-bit wide cache miss datapath.

#### 21.5.1.1 Cache controls

The cache control inputs are as follows:

- clr_lpcac: clear lpcac
- dis_lpcac: disable lpcac
- mode_ctl_hprot: ignore LPCAC Memory Regions

See Input controls for more information.

# Chapter 22
# AHB Low Power Local Memory Controller (AHB_LPLMEM)

## 22.1 Chip-specific LPLMEM information

### 22.1.1 Cache controls

LPLMEM is the cache for CM0+ or ZERO-RISCY core. MCM1_CPCR2 register is used to clear and disable the LPLMEM.

## 22.2 Introduction

The AHB Low Power Local Memory (LPLMEM) Controller is a processor-local Level 1 (L1) bus cache (bcache) controller for use with cores using AMBA-AHB input/output buses. It is optimized for minimum RunIDD (dynamic power).

### 22.2.1 Overview

The LPLMEM has a 32-bit datapath AMBA-AHB input bus and a 32-bit datapath AMBA-AHB output bus. Internally, LPLMEM decodes a cacheable region (with the cache region writethrough).

Any input AHB accesses with addresses that

- hit the cache region, use a portion of the attached RAM as a cache; cache misses use the AHB output bus to fetch the miss data.
- miss the cache region, go to the AHB output bus.

There are also a small number of control inputs (cache enable and cache clear control).

## 22.3 Signal descriptions

The LPLMEM has no external signals.

## 22.4 Memory map and Input Control description

### 22.4.1 Memory regions

The LPLMEM decodes cacheable address regions.

- The cache memory region may be composed of one or more disjoint memory regions and the total size may be larger than the cache storage.

Any address not in the cacheable memory regions is in the pass-through memory region.

### 22.4.2 Input controls

The LPLMEM does not have a module-resident programming model. An external block supplies the needed LPLMEM control inputs.

**Table 22-1. Control inputs for LPLMEM**

| Function | Control input | Description |
|---|---|---|
| clear bcache | clr_bcache | One cycle active-high pulse clears the bcache. |
| disable bcache | dis_bcache | • 0 = bcache enabled (reset configuration)<br>• 1 = bcache disabled, all cacheable accesses pass from LPLMEM input to output bus |

## 22.5 Functional description

This section provides further information on the LPLMEM operation.

The LPLMEM is cleared and enabled by a reset.

The LPLMEM examines every valid AHB input access. If the access hits the cache memory region and the cache is enabled, the access goes to the cache portion of the LPLMEM. If the access does not go to the cache, the access is passed directly to the AHB output bus.

The LPLMEM cache has a word (32-bit) cache line size:

- Byte, half-word, or word read cache misses do an aligned word-size cache miss read to the LPLMEM AHB output bus then load the needed data in the cache.
- Read cache hits return the desired read data from the cache.
- Byte, half-word, or word write cache misses do the desired write operation only to the LPLMEM AHB output bus (writethrough mode has a no cache allocation on write miss policy).
- Byte, half-word, or word write cache hits do the desired write operation to the cache and to the LPLMEM AHB output bus.

## 22.5.1 Cache Functional description

The LPLMEM cache is an 8-way, 2-set-associative write-through design. It supports total cache capacities of 4 KB for a 32-bit wide cache miss datapath.

### 22.5.1.1 Cache controls

There are 2 cache control inputs:

- clr_bcache: clear bcache
- dis_bcache: disable bcache

# Chapter 23
# Signal Multiplexing and Signal Descriptions

## 23.1 Signal Multiplexing Introduction

To optimize functionality in small packages, pins have several functions available via signal multiplexing. This chapter illustrates which of this device's signals are multiplexed on which external pin.

The Port Control block controls which signal is present on the external pin. Reference that chapter to find which register controls the operation of a specific pin.

## 23.2 Clock gating

The clock to the port control module can be gated on and off using the SCGC5[PORTx] bits in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing the corresponding module, set SCGC5[PORTx] in the SIM module to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution chapter.

# 23.3 Pinouts

## 23.3.1 RV32M1 Subfamily Pinout

**Table 23-1. RV32M1 Pinout**

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| C1 | PTB3 | LPADC0_SE0 | LPADC0_SE0 | PTB3/ RF0_EXT_OSC_EN | LPSPI0_PCS3 | LPUART1_TX | I2S0_TX_FS | FB_AD10 | TPM0_CH1 | — |
| C2 | PTB4/ LLWU_P6 | LPADC0_SE1 | LPADC0_SE1 | PTB4/ LLWU_P6 / RF0_RF_OFF/ RF0_DFT_RESET | LPSPI0_SCK | LPUART1_CTS | I2S0_TX_BCLK | FB_AD9 | TPM0_CH2 | — |
| D2 | PTB5 | DISABLED | — | PTB5/ RF0_ACTIVE | LPSPI0_SOUT | LPUART1_RTS | I2S0_MCLK | FB_AD8 | TPM0_CH3 | — |
| E1 | PTB6/ LLWU_P7 | DISABLED | — | PTB6/ LLWU_P7 | LPSPI0_PCS2 | LPI2C1_SDA | I2S0_RX_BCLK | FB_AD7 | TPM0_CH4 | RF0_BSM_FRAME |
| E2 | PTB7/ LLWU_P8 | LPADC0_SE2 | LPADC0_SE2 | PTB7/ LLWU_P8 | LPSPI0_SIN | LPI2C1_SDAS | I2S0_RX_FS | FB_AD6 | TPM0_CH5 | RF0_BSM_DATA |
| F5 | PTB8/ LLWU_P9 | DISABLED | — | PTB8/ LLWU_P9 / RF0_EARLY_WARNING | LPSPI0_PCS0 | LPI2C1_SCLS | I2S0_RXD0 | FB_AD5 | — | LPTMR0_ALT1 |
| F4 | PTB9 | LPADC0_SE3 | LPADC0_SE3 | PTB9/ SPM_LPREQ | LPSPI0_PCS1 | LPI2C1_SCL | I2S0_RXD1 | FB_RW_b | — | FXIO0_D0 |
| D5 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| C9 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| G6 | PTB11 | DISABLED | — | PTB11 | LPUART2_RX | LPI2C1_SDAS | LPI2C0_SDA | FB_AD27 | — | FXIO0_D1 |
| G4 | PTB12 | DISABLED | — | PTB12 | LPUART2_TX | LPI2C1_SCLS | LPI2C0_SCL | FB_AD26 | TPM3_CLKIN | FXIO0_D2 |
| G3 | PTB13 | DISABLED | — | PTB13 | LPUART2_CTS | LPI2C1_SDA | LPI2C0_SDAS | FB_AD25 | TPM3_CH0 | FXIO0_D3 |
| G2 | PTB14 | DISABLED | — | PTB14 | LPUART2_RTS | LPI2C1_SCL | LPI2C0_SCLS | FB_AD24 | TPM3_CH1 | FXIO0_D4 |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| G1 | PTB15 | DISABLED | — | PTB15 | — | LPI2C1_HREQ | LPI2C3_SCL | FB_CS5_b/ FB_TSIZ1/ FB_BE23_16_b | TPM0_CLKIN | FXIO0_D5 |
| H5 | PTB16/ LLWU_P10 | DISABLED | — | PTB16/ LLWU_P10 | — | LPUART3_CTS | LPI2C3_SDA | FB_CS4_b/ FB_TSIZ0/ FB_BE31_24_b | — | FXIO0_D6 |
| K5 | PTB17 | DISABLED | — | PTB17 | — | LPUART3_RTS | LPI2C3_SCLS | FB_TBST_b/ FB_CS2_b/ FB_BE15_8_b | — | FXIO0_D7 |
| H2 | PTB18 | DISABLED | — | PTB18 | LPSPI1_PCS1 | LPUART2_RX | LPI2C3_SDAS | FB_CS3_b/ FB_BE7_0_b | FB_TA_b | FXIO0_D8 |
| K4 | PTB19 | DISABLED | — | PTB19 | LPSPI1_PCS3 | LPUART2_TX | — | FB_ALE/ FB_CS1_b/ FB_TS_b | TPM1_CLKIN | FXIO0_D9 |
| J1 | PTB20/ LLWU_P11 | DISABLED | — | PTB20/ LLWU_P11 | LPSPI1_SCK | LPUART2_CTS | — | FB_CS0_b | TPM1_CH0 | FXIO0_D10 |
| J2 | PTB21 | DISABLED | — | PTB21 | LPSPI1_SOUT | LPUART2_RTS | LPI2C2_HREQ | FB_AD4 | TPM1_CH1 | FXIO0_D11 |
| L1 | PTB22/ LLWU_P12 | DISABLED | — | PTB22/ LLWU_P12 | LPSPI1_PCS2 | LPUART0_CTS | LPI2C2_SDA | FB_AD3 | TPM2_CLKIN | FXIO0_D12 |
| J4 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| J3 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| L2 | PTB24 | DISABLED | — | PTB24 | LPSPI1_SIN | LPUART0_RTS | LPI2C2_SCL | FB_AD2 | EWM_IN | FXIO0_D13 |
| L6 | PTB25/ LLWU_P13 | DISABLED | — | PTB25/ LLWU_P13 | LPSPI1_PCS0 | LPUART0_RX | LPI2C2_SDAS | FB_AD1 | EWM_OUT_b | FXIO0_D14 |
| L4 | PTB26 | DISABLED | — | PTB26 | USB0_SOF_OUT | LPUART0_TX | LPI2C2_SCLS | FB_AD0 | LPCMP0_OUT | RF0_BSM_CLK |
| M4 | PTB28/ LLWU_P14 | DISABLED | — | PTB28/ LLWU_P14 | — | LPUART3_RX | I2S0_TXD0 | FB_A16 | — | FXIO0_D15 |
| L3 | PTB29 | DISABLED | — | PTB29 | — | LPUART3_TX | I2S0_TX_FS | FB_A17 | — | FXIO0_D16 |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| M5 | PTB30 | DISABLED | — | PTB30 | — | — | I2S0_TX_BCLK | FB_A18 | — | — |
| M7 | PTB31 | DISABLED | — | PTB31 | — | — | I2S0_RXD0 | FB_A19 | — | — |
| N1 | PTC0 | DISABLED | — | PTC0 | — | — | I2S0_RX_FS | FB_A20 | — | — |
| M2 | PTC1 | DISABLED | — | PTC1 | — | — | I2S0_RX_BCLK | FB_A21 | — | — |
| N3 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| J10 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| N2 | PTC7/ LLWU_P15 | LPCMP0_IN0 | LPCMP0_IN0 | PTC7/ LLWU_P15 | LPSPI0_PCS3 | LPUART0_RX | LPI2C1_HREQ | — | TPM0_CH0 | LPTMR1_ALT1 |
| P3 | PTC8 | LPCMP0_IN1 | LPCMP0_IN1 | PTC8 | LPSPI0_SCK | LPUART0_TX | LPI2C0_HREQ | — | TPM0_CH1 | — |
| R1 | PTC9/ LLWU_P16 | LPADC0_SE4/ LPCMP0_IN2 | LPADC0_SE4/ LPCMP0_IN2 | PTC9/ LLWU_P16 | LPSPI0_SOUT | LPUART0_CTS | LPI2C0_SDA | — | TPM0_CH2 | LPTMR0_ALT2 |
| R2 | PTC10 | LPADC0_SE5 | LPADC0_SE5 | PTC10 | LPSPI0_PCS2 | LPUART0_RTS | LPI2C0_SCL | — | TPM0_CH3 | — |
| T1 | PTC11/ LLWU_P17 | LPADC0_SE6 | LPADC0_SE6 | PTC11/ LLWU_P17 | LPSPI0_SIN | LPI2C1_SDA | LPI2C0_SDAS | — | TPM0_CH4 | EWM_IN |
| R3 | PTC12/ LLWU_P18 | LPADC0_SE7 | LPADC0_SE7 | PTC12/ LLWU_P18 | LPSPI0_PCS0 | LPI2C1_SCL | LPI2C0_SCLS | — | TPM0_CH5 | EWM_OUT_b |
| U1 | VDD_DCDC | VDD_DCDC | VDD_DCDC | — | — | — | — | — | — | — |
| U2 | LP | LP | LP | — | — | — | — | — | — | — |
| U3 | GND | GND | GND | — | — | — | — | — | — | — |
| T4 | LN | LN | LN | — | — | — | — | — | — | — |
| T3 | VOUT_RF | VOUT_RF | VOUT_RF | — | — | — | — | — | — | — |
| T5 | VOUT_CORE | VOUT_CORE | VOUT_CORE | — | — | — | — | — | — | — |
| R9 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| P9 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| N15 | VDD_CORE | VDD_CORE | VDD_CORE | — | — | — | — | — | — | — |
| P6 | PTC27 | DISABLED | — | PTC27 | — | — | — | — | TPM0_CH4 | — |
| U5 | PTC28 | DISABLED | — | PTC28 | — | LPSPI0_PCS1 | — | — | TPM0_CH3 | FXIO0_D17 |

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| N6 | PTC29 | DISABLED | — | PTC29 | LPUART1_RX | LPSPI0_PCS3 | — | — | TPM0_CH2 | FXIO0_D18 |
| R7 | PTC30 | DISABLED | — | PTC30 | LPUART1_TX | LPSPI0_SCK | — | — | TPM0_CH1 | FXIO0_D19 |
| R5 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| R13 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| T7 | PTD0 | DISABLED | — | PTD0 | LPUART1_CTS | LPSPI0_SOUT | — | — | TPM0_CH0 | FXIO0_D20 |
| P7 | PTD1 | DISABLED | — | PTD1 | LPUART1_RTS | LPSPI0_PCS2 | — | — | EWM_IN | FXIO0_D21 |
| U7 | PTD2 | DISABLED | — | PTD2 | SDHC0_D7 | LPSPI0_SIN | — | — | EWM_OUT_b | FXIO0_D22 |
| T8 | PTD3 | DISABLED | — | PTD3 | SDHC0_D6 | LPSPI0_PCS0 | EMVSIM0_CLK | — | TPM2_CLKIN | FXIO0_D23 |
| N8 | PTD4 | DISABLED | — | PTD4 | SDHC0_D5 | LPSPI2_PCS1 | EMVSIM0_RST | — | — | FXIO0_D24 |
| N10 | PTD5 | LPADC0_SE8 | LPADC0_SE8 | PTD5 | SDHC0_D4 | LPSPI2_PCS3 | EMVSIM0_VCCEN | — | — | FXIO0_D25 |
| U9 | PTD6 | LPADC0_SE9 | LPADC0_SE9 | PTD6 | SDHC0_D1 | LPSPI2_SCK | EMVSIM0_IO | TRACE_D3 | TPM2_CH5 | FXIO0_D26 |
| P10 | PTD7 | LPADC0_SE10 | LPADC0_SE10 | PTD7 | SDHC0_D0 | LPSPI2_SOUT | EMVSIM0_PD | TRACE_D2 | TPM2_CH4 | FXIO0_D27 |
| T9 | PTD8/ LLWU_P19 | LPADC0_SE11 | LPADC0_SE11 | PTD8/ LLWU_P19 | SDHC0_DCLK | LPSPI2_PCS2 | LPI2C1_SDAS | TRACE_D1 | TPM2_CH3 | FXIO0_D28 |
| U11 | PTD9 | LPADC0_SE12 | LPADC0_SE12 | PTD9 | SDHC0_CMD | LPSPI2_SIN | LPI2C1_SCLS | TRACE_D0 | TPM2_CH2 | FXIO0_D29 |
| P11 | PTD10/ LLWU_P20 | LPADC0_SE13 | LPADC0_SE13 | PTD10/ LLWU_P20 | SDHC0_D3 | LPSPI2_PCS0 | LPI2C1_SDA | TRACE_CLKOUT | TPM2_CH1 | FXIO0_D30 |
| R11 | PTD11 | LPADC0_SE14 | LPADC0_SE14 | PTD11 | SDHC0_D2 | USB0_SOF_OUT | LPI2C1_SCL | CLKOUT | TPM2_CH0 | FXIO0_D31 |
| P5 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| P13 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| N4 | USB0_VSS | USB0_VSS | USB0_VSS | — | — | — | — | — | — | — |
| T11 | USB0_DP | USB0_DP | USB0_DP | — | — | — | — | — | — | — |
| T12 | USB0_DM | USB0_DM | USB0_DM | — | — | — | — | — | — | — |
| T13 | VOUT33 | VOUT33 | VOUT33 | — | — | — | — | — | — | — |
| U13 | VREGIN | VREGIN | VREGIN | — | — | — | — | — | — | — |
| U15 | VDDA | VDDA | VDDA | — | — | — | — | — | — | — |
| U16 | VREFH | VREFH | VREFH | — | — | — | — | — | — | — |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| T15 | VREF_OUT | VREF_OUT/ LPADC0_SE15/ LPCMP0_IN5/ LPCMP1_IN5 | VREF_OUT/ LPADC0_SE15/ LPCMP0_IN5/ LPCMP1_IN5 | — | — | — | — | — | — | — |
| U17 | VREFL | VREFL | VREFL | — | — | — | — | — | — | — |
| T16 | VSSA | VSSA | VSSA | — | — | — | — | — | — | — |
| T17 | DAC0_OUT | DAC0_OUT/ LPADC0_SE16/ LPCMP0_IN3/ LPCMP1_IN3 | DAC0_OUT/ LPADC0_SE16/ LPCMP0_IN3/ LPCMP1_IN3 | — | — | — | — | — | — | — |
| R14 | PTE0 | LPCMP1_IN4 | LPCMP1_IN4 | PTE0 | — | — | — | — | EWM_IN | — |
| R16 | PTE1/ LLWU_P21 | LPADC0_SE18 | LPADC0_SE18 | PTE1/ LLWU_P21 | SDHC0_D1 | LPI2C0_SDAS | LPSPI3_PCS1 | — | EWM_OUT_b | LPTMR1_ALT2 |
| P12 | PTE2 | LPADC0_SE19 | LPADC0_SE19 | PTE2 | SDHC0_D0 | LPI2C0_SCLS | LPSPI3_PCS3 | — | LPCMP1_OUT | — |
| N12 | PTE3/ LLWU_P22 | LPADC0_SE20/ LPCMP1_IN0 | LPADC0_SE20/ LPCMP1_IN0 | PTE3/ LLWU_P22 | SDHC0_D7 | LPI2C0_SDA | LPSPI3_SCK | — | TPM0_CLKIN | LPTMR0_ALT3 |
| M11 | PTE4 | LPADC0_SE21/ LPCMP1_IN1 | LPADC0_SE21/ LPCMP1_IN1 | PTE4 | SDHC0_D6 | LPI2C0_SCL | LPSPI3_SOUT | CLKOUT | TPM1_CLKIN | RF0_DTM_RX |
| R17 | PTE5 | LPCMP1_IN2 | LPCMP1_IN2 | PTE5 | SDHC0_DCLK | LPI2C0_HREQ | LPSPI3_PCS2 | — | LPCMP1_OUT | RF0_DTM_TX |
| J8 | VDD_CORE | VDD_CORE | VDD_CORE | — | — | — | — | — | — | — |
| K10 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| D13 | VDDIO2 | VDDIO2 | VDDIO2 | — | — | — | — | — | — | — |
| P16 | PTE8/ LLWU_P23 | LPADC0_SE22 | LPADC0_SE22 | PTE8/ LLWU_P23 | SDHC0_D5 | LPUART3_RX | LPSPI3_SIN | — | TPM1_CH0 | LPTMR2_ALT1 |
| N16 | PTE9/ LLWU_P24 | LPADC0_SE23 | LPADC0_SE23 | PTE9/ LLWU_P24 | SDHC0_CMD | LPUART3_TX | LPSPI3_PCS0 | — | TPM1_CH1 | FXIO0_D0 |
| M13 | PTE10/ LLWU_P25 | DISABLED | — | PTE10/ LLWU_P25 | SDHC0_D4 | LPUART3_CTS | LPI2C3_SDA | — | TPM3_CH0 | LPTMR2_ALT3 |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| M14 | PTE11 | DISABLED | — | PTE11 | SDHC0_D3 | LPUART3_RTS | LPI2C3_SCL | — | TPM3_CH1 | FXIO0_D1 |
| L12 | PTE12/ LLWU_P26 | DISABLED | — | PTE12/ LLWU_P26 | SDHC0_D2 | — | LPI2C3_SDAS | — | TPM3_CLKIN | FXIO0_D2 |
| N17 | PTE13 | DISABLED | — | PTE13 | I2S0_TX_BCLK | — | LPI2C3_SCLS | — | TPM3_CH0 | FXIO0_D3 |
| L16 | PTE14 | DISABLED | — | PTE14 | I2S0_TX_FS | — | LPI2C3_HREQ | — | TPM3_CH1 | FXIO0_D4 |
| L17 | PTE15 | DISABLED | — | PTE15 | I2S0_TXD0 | — | — | — | TPM3_CLKIN | FXIO0_D5 |
| L14 | PTE16 | DISABLED | — | PTE16 | I2S0_RX_BCLK | — | — | — | TPM2_CH0 | FXIO0_D6 |
| L15 | PTE17 | DISABLED | — | PTE17 | I2S0_RX_FS | — | — | — | TPM2_CH1 | FXIO0_D7 |
| K13 | PTE18 | DISABLED | — | PTE18 | I2S0_RXD0 | — | — | — | TPM2_CH2 | FXIO0_D8 |
| K16 | PTE19 | DISABLED | — | PTE19 | I2S0_MCLK | — | — | — | TPM2_CH3 | FXIO0_D9 |
| J17 | PTE21 | DISABLED | — | PTE21 | I2S0_TXD1 | USB0_SOF_OUT | — | — | TPM2_CH4 | FXIO0_D10 |
| J16 | PTE22 | DISABLED | — | PTE22 | I2S0_RXD1 | LPI2C3_HREQ | — | — | TPM2_CH5 | FXIO0_D11 |
| J14 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| J15 | VDDIO2 | VDDIO2 | VDDIO2 | — | — | — | — | — | — | — |
| H14 | PTE27 | DISABLED | — | PTE27 | LPUART3_CTS | LPI2C3_SDAS | — | — | — | FXIO0_D28 |
| G14 | PTE28 | DISABLED | — | PTE28 | LPUART3_RTS | LPI2C3_SCLS | — | — | — | FXIO0_D29 |
| G15 | PTE29 | DISABLED | — | PTE29 | LPUART3_RX | LPI2C3_SDA | — | — | — | FXIO0_D30 |
| G17 | PTE30 | DISABLED | — | PTE30 | LPUART3_TX | LPI2C3_SCL | — | — | TPM2_CLKIN | FXIO0_D31 |
| H13 | TAMPER3 | TAMPER3 | TAMPER3 | — | — | — | — | — | — | — |
| G12 | TAMPER2 | TAMPER2 | TAMPER2 | — | — | — | — | — | — | — |
| F13 | TAMPER1/ RTC_CLKOUT | TAMPER1/ RTC_CLKOUT | TAMPER1/ RTC_CLKOUT | — | — | — | — | — | — | — |
| F14 | TAMPER0/ RTC_WAKEUP_b | TAMPER0/ RTC_WAKEUP_b | TAMPER0/ RTC_WAKEUP_b | — | — | — | — | — | — | — |
| G16 | VBAT | VBAT | VBAT | — | — | — | — | — | — | — |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| E17 | XTAL32 | XTAL32 | XTAL32 | — | — | — | — | — | — | — |
| E16 | EXTAL32 | EXTAL32 | EXTAL32 | — | — | — | — | — | — | — |
| E15 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| C17 | VDD_RF | VDD_RF | VDD_RF | — | — | — | — | — | — | — |
| B17 | VDD_RF | VDD_RF | VDD_RF | — | — | — | — | — | — | — |
| A16 | GANT | GANT | GANT | — | — | — | — | — | — | — |
| A15 | ANT | ANT | ANT | — | — | — | — | — | — | — |
| B13 | VDD_RF | VDD_RF | VDD_RF | — | — | — | — | — | — | — |
| A11 | XTAL_RF | XTAL_RF | XTAL_RF | — | — | — | — | — | — | — |
| B11 | EXTAL_RF | EXTAL_RF | EXTAL_RF | — | — | — | — | — | — | — |
| C11 | RF_CLKOUT | RF_CLKOUT | RF_CLKOUT | — | — | — | — | — | — | — |
| D12 | RESET_b | RESET_b | RESET_b | — | — | — | — | — | — | — |
| B10 | PTA0 | NMI_b | — | PTA0 | — | — | — | — | — | NMI_b |
| E12 | PTA1/ LLWU_P0 | JTAG_TCLK/ SWD_CLK | — | PTA1/ LLWU_P0 / RF_ANT_B | LPUART0_CTS | LPI2C0_SDAS | LPUART1_CTS | — | — | JTAG_TCLK/ SWD_CLK |
| F11 | PTA2/ LLWU_P1 | JTAG_TDI | — | PTA2/ LLWU_P1 / RF_ANT_A | LPUART0_RX | LPI2C0_SDA | LPUART1_RX | — | — | JTAG_TDI |
| D11 | PTA3 | JTAG_TDO/ SWD_SWO | — | PTA3/ RF0_TX_SWITCH | LPUART0_TX | LPI2C0_SCL | LPUART1_TX | — | TPM0_CLKIN | JTAG_TDO/ SWD_SWO |
| B9 | PTA4 | JTAG_TMS/ SWD_DIO | — | PTA4/ RF0_RX_SWITCH | LPUART0_RTS | LPI2C0_SCLS | LPUART1_RTS | — | LPCMP0_OUT | JTAG_TMS/ SWD_DIO |
| H9 | VDD_CORE | VDD_CORE | VDD_CORE | — | — | — | — | — | — | — |
| E14 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| K9 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| E10 | PTA9 | RV_JTAG_TCLK | — | PTA9 | LPI2C2_SDAS | LPSPI3_SCK | — | FB_A23 | — | RV_JTAG_TCLK |
| A9 | PTA10 | RV_JTAG_TDI | — | PTA10 | LPI2C2_SCLS | LPSPI3_SOUT | — | FB_A22 | — | RV_JTAG_TDI |
| E8 | PTA14 | RV_JTAG_TDO | — | PTA14 | LPI2C2_SDA | — | — | FB_AD23 | LPCMP0_OUT | RV_JTAG_TDO |
| A7 | PTA15 | RV_JTAG_TMS | — | PTA15 | LPI2C2_SCL | — | — | FB_AD22 | — | RV_JTAG_TMS |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| A17 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| E4 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| F7 | PTA17 | DISABLED | — | PTA17 | LPI2C2_HREQ | LPSPI3_PCS1 | EMVSIM0_CLK | FB_AD21 | — | — |
| D8 | PTA18 | DISABLED | — | PTA18 | LPSPI2_PCS1 | LPSPI3_PCS3 | EMVSIM0_RST | FB_AD20 | — | — |
| D7 | PTA19 | DISABLED | — | PTA19 | LPSPI2_PCS3 | LPSPI3_SCK | EMVSIM0_VCCEN | FB_AD19 | TPM2_CH5 | — |
| C7 | PTA20 | DISABLED | — | PTA20 | LPSPI2_SCK | LPSPI1_PCS1 | EMVSIM0_IO | FB_AD18 | TPM2_CH4 | — |
| B7 | PTA21 | DISABLED | — | PTA21 | LPSPI2_SOUT | — | EMVSIM0_PD | FB_AD17 | TPM2_CH3 | — |
| B6 | PTA22/ LLWU_P2 | DISABLED | — | PTA22/ LLWU_P2 | LPSPI2_PCS2 | — | LPI2C2_HREQ | FB_AD16 | TPM2_CH2 | — |
| E6 | PTA23 | DISABLED | — | PTA23 | LPSPI2_SIN | LPSPI1_PCS3 | LPI2C2_SDA | FB_AD15 | TPM2_CH1 | — |
| D6 | PTA24 | DISABLED | — | PTA24 | LPSPI2_PCS0 | LPSPI1_SCK | LPI2C2_SCL | FB_OE_b | TPM2_CH0 | — |
| B5 | PTA25 | DISABLED | — | PTA25 | LPUART1_RX | LPSPI3_SOUT | LPI2C2_SDAS | FB_AD31 | — | — |
| A5 | PTA26 | DISABLED | — | PTA26 | LPUART1_TX | LPSPI3_PCS2 | LPI2C2_SCLS | FB_AD30 | — | — |
| A3 | PTA27 | DISABLED | — | PTA27 | LPUART1_CTS | LPSPI3_SIN | — | FB_AD29 | — | — |
| A2 | PTA28 | DISABLED | — | PTA28 | LPUART1_RTS | LPSPI3_PCS0 | — | FB_AD28 | — | — |
| A13 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| E3 | VDDIO1 | VDDIO1 | VDDIO1 | — | — | — | — | — | — | — |
| A1 | PTA30/ LLWU_P3 | DISABLED | — | PTA30/ LLWU_P3 | LPUART2_CTS | LPSPI1_SOUT | — | FB_AD14 | TPM1_CH0 | LPTMR2_ALT2 |
| C4 | PTA31 | DISABLED | — | PTA31 | LPUART2_RTS | LPSPI1_PCS2 | — | FB_AD13 | TPM1_CH1 | — |
| B3 | PTB0 | DISABLED | — | PTB0 | LPUART2_TX | LPSPI1_SIN | USB0_SOF_OUT | CLKOUT | TPM1_CLKIN | — |
| C3 | PTB1/ LLWU_P4 | DISABLED | — | PTB1/ LLWU_P4 | LPUART2_RX | LPSPI1_PCS0 | I2S0_TXD1 | FB_AD12 | — | LPTMR1_ALT3 |
| B1 | PTB2/ LLWU_P5 | DISABLED | — | PTB2/ LLWU_P5 / RF0_RFOSC_EN | LPSPI0_PCS1 | LPUART1_RX | I2S0_TXD0 | FB_AD11 | TPM0_CH0 | — |
| B14 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| B15 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| B16 | VSS | VSS | VSS | — | — | — | — | — | — | — |

*Table continues on the next page...*

## Table 23-1. RV32M1 Pinout (continued)

| 176 VFBGA | Pin Name | DEFAULT | ALT0 | ALT1 | ALT2 | ALT3 | ALT4 | ALT5 | ALT6 | ALT7 |
|---|---|---|---|---|---|---|---|---|---|---|
| C5 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| C13 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| C15 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| C16 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| D9 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| D15 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| F16 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| H8 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| H10 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| K8 | VSS | VSS | VSS | — | — | — | — | — | — | — |
| N14 | VDD_CORE | VDD_CORE | VDD_CORE | — | — | — | — | — | — | — |
| R15 | VSSA | VSSA | VSSA | — | — | — | — | — | — | — |
| B2 | PTA30/ LLWU_P3 | DISABLED | — | PTA30/ LLWU_P3 | LPUART2_CTS | LPSPI1_SOUT | — | FB_AD14 | TPM1_CH0 | LPTMR2_ALT2 |

## 23.3.2 Pin properties

The following table lists the pin properties.

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | PTB3 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| C2 | PTB4/ LLWU_ P6 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| D2 | PTB5 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E1 | PTB6/ LLWU_ P7 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| E2 | PTB7/ LLWU_ P8 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| F5 | PTB8/ LLWU_ P9 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| F4 | PTB9 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| D5 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C9 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| G6 | PTB11 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G4 | PTB12 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G3 | PTB13 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G2 | PTB14 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G1 | PTB15 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| H5 | PTB16/ LLWU_ P10 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| K5 | PTB17 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| H2 | PTB18 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| K4 | PTB19 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J1 | PTB20/ LLWU_ P11 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J2 | PTB21 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L1 | PTB22/ LLWU_ P12 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J4[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| J3 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| L2 | PTB24 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L6 | PTB25/ LLWU_ P13 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L4 | PTB26 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| M4 | PTB28/ LLWU_ P14 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L3 | PTB29 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| M5 | PTB30 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| M7 | PTB31 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| N1 | PTC0 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| M2 | PTC1 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| N3[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| J10 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| N2 | PTC7/ LLWU_ P15 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| P3 | PTC8 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| R1 | PTC9/ LLWU_ P16 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| R2 | PTC10 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| T1 | PTC11/ LLWU_ P17 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| R3 | PTC12/ LLWU_ P18 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| U1 | VDD_D CDC | — | — | — | — | — | — | — | — | — | — | — |
| U2 | LP | — | — | — | — | — | — | — | — | — | — | — |
| U3 | GND | — | — | — | — | — | — | — | — | — | — | — |
| T4 | LN | — | — | — | — | — | — | — | — | — | — | — |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T3 | VOUT_RF | — | — | — | — | — | — | — | — | — | — | — |
| T5 | VOUT_CORE | — | — | — | — | — | — | — | — | — | — | — |
| R9 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| P9 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| N15 | VDD_CORE | — | — | — | — | — | — | — | — | — | — | — |
| — | PTC26 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| P6 | PTC27 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| U5 | PTC28 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| N6 | PTC29 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| R7 | PTC30 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| R5 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| R13 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| T7 | PTD0 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| P7 | PTD1 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| U7 | PTD2 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| T8 | PTD3 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| N8 | PTD4 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| N10 | PTD5 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| U9 | PTD6 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| P10 | PTD7 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| T9 | PTD8/ LLWU_P19 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| U11 | PTD9 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| P11 | PTD10/ LLWU_P20 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| R11 | PTD11 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| P5 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P13[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| N4 | USB0_VSS | — | — | — | — | — | — | — | — | — | — | — |
| T11 | USB0_DP | — | — | — | — | — | — | — | — | — | — | — |
| T12 | USB0_DM | — | — | — | — | — | — | — | — | — | — | — |
| T13 | VOUT33 | — | — | — | — | — | — | — | — | — | — | — |
| U13 | VREGIN | — | — | — | — | — | — | — | — | — | — | — |
| U15 | VDDA | — | — | — | — | — | — | — | — | — | — | — |
| U16 | VREFH | — | — | — | — | — | — | — | — | — | — | — |
| T15 | VREF_OUT | — | — | — | — | — | — | — | — | — | — | — |
| U17 | VREFL | — | — | — | — | — | — | — | — | — | — | — |
| T16 | VSSA | — | — | — | — | — | — | — | — | — | — | — |
| T17 | DAC0_OUT | — | — | — | — | — | — | — | — | — | — | — |
| R14 | PTE0 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| R16 | PTE1/LLWU_P21 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| P12 | PTE2 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| N12 | PTE3/LLWU_P22 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| M11 | PTE4 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| R17 | PTE5 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| J8 | VDD_CORE | — | — | — | — | — | — | — | — | — | — | — |
| K10[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| D13 | VDDIO2 | — | — | — | — | — | — | — | — | — | — | — |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P16 | PTE8/ LLWU_ P23 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| N16 | PTE9/ LLWU_ P24 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| M13 | PTE10/ LLWU_ P25 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| M14 | PTE11 | HD | Hi-Z | N | PD | FS | N | N | Y | Y | N | Y |
| L12 | PTE12/ LLWU_ P26 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| N17 | PTE13 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L16 | PTE14 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L17 | PTE15 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L14 | PTE16 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| L15 | PTE17 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| K13 | PTE18 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| K16 | PTE19 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J17 | PTE21 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J16 | PTE22 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| J14[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| J15 | VDDIO2 | — | — | — | — | — | — | — | — | — | — | — |
| H14 | PTE27 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G14 | PTE28 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G15 | PTE29 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| G17 | PTE30 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| H13 | TAMPE R3 | ND | Hi-Z | N | PU | FS | N | N | N | N | N | N |
| G12 | TAMPE R2 | ND | Hi-Z | N | PU | FS | N | N | N | N | N | N |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F13 | TAMPER1/ RTC_CLKOUT | ND | Hi-Z | N | PU | FS | N | N | N | N | N | N |
| F14 | TAMPER0/ RTC_WAKEUP_b | ND | Hi-Z | N | PU | FS | N | N | N | N | N | N |
| G16 | VBAT | — | — | — | — | — | — | — | — | — | — | — |
| E17 | XTAL32 | — | — | — | — | — | — | — | — | — | — | — |
| E16 | EXTAL32 | — | — | — | — | — | — | — | — | — | — | — |
| E15 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C17 | VDD_RF | — | — | — | — | — | — | — | — | — | — | — |
| B17 | VDD_RF | — | — | — | — | — | — | — | — | — | — | — |
| A16 | GANT | — | — | — | — | — | — | — | — | — | — | — |
| A15 | ANT | — | — | — | — | — | — | — | — | — | — | — |
| B13 | VDD_RF | — | — | — | — | — | — | — | — | — | — | — |
| A11 | XTAL_RF | — | — | — | — | — | — | — | — | — | — | — |
| B11 | EXTAL_RF | — | — | — | — | — | — | — | — | — | — | — |
| C11 | RF_CLKOUT | — | — | — | — | — | — | — | — | — | — | — |
| D12 | RESET_b | ND | H | N | PU | — | Y | — | — | — | — | — |
| B10 | PTA0 | ND | H | Y | PU | FS | N | N | Y | Y | N | N |
| E12 | PTA1/ LLWU_P0 | ND | L | Y | PD | FS | N | N | Y | Y | N | N |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F11 | PTA2/ LLWU_ P1 | ND | H | Y | PU | FS | N | N | Y | Y | N | N |
| D11 | PTA3 | ND | H | Y | PU | FS | N | N | Y | Y | N | N |
| B9 | PTA4 | ND | H | Y | PU | FS | N | N | Y | Y | N | N |
| H9 | VDD_C ORE | — | — | — | — | — | — | — | — | — | — | — |
| E14 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| K9 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| E10 | PTA9 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A9 | PTA10 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| E8 | PTA14 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A7 | PTA15 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A17[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| E4 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |
| F7 | PTA17 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| D8 | PTA18 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| D7 | PTA19 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| C7 | PTA20 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B7 | PTA21 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B6 | PTA22/ LLWU_ P2 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| E6 | PTA23 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| D6 | PTA24 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B5 | PTA25 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A5 | PTA26 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A3 | PTA27 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A2 | PTA28 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| A13[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| E3 | VDDIO1 | — | — | — | — | — | — | — | — | — | — | — |

*Table continues on the next page...*

| 176 VFBGA | Pin Name | Drive strength | Default status after POR | Pullup/ Pulldown enable after POR | Pullup/ pulldown selection after POR | Slew rate after POR | Passive pin filter after POR | Open drain enable at reset | Open drain enable control | Pin interrupt | Fast capability | Fast high drive |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | PTA30/ LLWU_ P3 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| C4 | PTA31 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B3 | PTB0 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | Y | N |
| C3 | PTB1/ LLWU_ P4 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B1 | PTB2/ LLWU_ P5 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |
| B14 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| B15 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| B16 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C5[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C13[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C15[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| C16[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| D9 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| D15 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| F16[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| H8[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| H10[1] | VSS | — | — | — | — | — | — | — | — | — | — | — |
| K8 | VSS | — | — | — | — | — | — | — | — | — | — | — |
| N14 | VDD_C ORE | — | — | — | — | — | — | — | — | — | — | — |
| R15[1] | VSSA | — | — | — | — | — | — | — | — | — | — | — |
| T2 | VDD_D CDC | — | — | — | — | — | — | — | — | — | — | — |
| B2 | PTA30/ LLWU_ P3 | ND | Hi-Z | N | PD | FS | N | N | Y | Y | N | N |

1. Optional, can be left as NC.

| Properties | Abbreviation | Descriptions |
|---|---|---|
| Driver strength | ND | Normal drive |
| | HD | High drive |
| Default status after POR | Hi-Z | High impendence |
| | H | High level |
| | L | Low level |
| Pullup/ pulldown Enable after POR | Y | Enabled |
| | N | Disabled |
| Pullup/ pulldown selection after POR | PU | Pullup |
| | PD | Pulldown |
| Slew rate after POR | FS | Fast slew rate |
| | SS | Slow slew rate |
| Passive Pin Filter after POR | N | Disabled |
| | Y | Enabled |
| Open drain | N | Disabled[1] |
| | Y | Enabled |
| Pin interrupt | Y | Yes |
| Fast capability | N | Not support fast capability |
| | Y | Support fast capability |
| Fast high drive | N | Not support fast high drive |
| | Y | Support fast high drive |

1. When an LPI2C module is enabled and a pin is functional for LPI2C, this pin is (pseudo-) open drain enabled. When an LPUART module is enabled and a pin is functional for LPUART, this pin is (pseudo-) open drain configurable.

## 23.3.3 RV32M1 pinouts diagram

The below figure shows the pinout diagram for the devices supported by this document. Many signals may be multiplexed onto a single pin. To determine what signals can be used on which pin, see the previous section.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17A | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTA30/LLWU_P3 | PTA28 | PTA27 | NC | PTA26 | NC | PTA15 | NC | PTA10 | NC | XTAL_RF | NC | VSS | NC | ANT | GANT | VSS | A |
| B | PTB2/LLWU_P5 | PTA30/LLWU_P3 | PTB0 | NC | PTA25 | PTA22/LLWU_P2 | PTA21 | NC | PTA4 | PTA0 | EXTAL_RF | NC | VDD_RF | VSS | VSS | VSS | VDD_RF | B |
| C | PTB3 | PTB4/LLWU_P6 | PTB1/LLWU_P4 | PTA31 | VSS | NC | PTA20 | NC | VDDIO1 | NC | RF_CLKOUT | NC | VSS | NC | VSS | VSS | VDD_RF | C |
| D | NC | PTB5 | NC | NC | VSS | PTA24 | PTA19 | PTA18 | VSS | NC | PTA3 | RESET_b | VDDIO2 | NC | VSS | NC | NC | D |
| E | PTB6/LLWU_P7 | PTB7/LLWU_P8 | VDDIO1 | VDDIO1 | NC | PTA23 | NC | PTA14 | NC | PTA9 | NC | PTA1/LLWU_P0 | NC | VSS | VSS | EXTAL32 | XTAL32 | E |
| F | NC | NC | NC | PTB9 | PTB8/LLWU_P9 | NC | PTA17 | NC | NC | NC | PTA2/LLWU_P1 | NC | TAMPER1/RTC_CLKOUT | TAMPER0/RTC_WAKEUP | NC | VSS | NC | F |
| G | PTB15 | PTB14 | PTB13 | PTB12 | NC | PTB11 | NC | NC | NC | NC | NC | TAMPER2 | NC | PTE28 | PTE29 | VBAT | PTE30 | G |
| H | NC | PTB18 | NC | NC | PTB16/LLWU_P10 | NC | NC | VSS | VDD_CORE | VSS | NC | NC | TAMPER3 | PTE27 | NC | NC | NC | H |
| J | PTB20/LLWU_P11 | PTB21 | VDDIO1 | VSS | NC | NC | NC | VDD_CORE | NC | VDDIO1 | NC | NC | NC | VSS | VDDIO2 | PTE22 | PTE21 | J |
| K | NC | NC | NC | PTB19 | PTB17 | NC | NC | VSS | VDDIO1 | VSS | NC | NC | PTE18 | NC | NC | PTE19 | NC | K |
| L | PTB22/LLWU_P12 | PTB24 | PTB29 | PTB26 | NC | PTB25/LLWU_P13 | NC | NC | NC | NC | NC | PTE12/LLWU_P26 | NC | PTE16 | PTE17 | PTE14 | PTE15 | L |
| M | NC | PTC1 | NC | PTB28/LLWU_P14 | PTB30 | NC | PTB31 | NC | NC | NC | PTE4 | NC | PTE10/LLWU_P25 | PTE11 | NC | NC | NC | M |
| N | PTC0 | PTC7/LLWU_P15 | VSS | USB0_VSS | NT | PTC29 | NT | PTD4 | NT | PTD5 | NT | PTE3/LLWU_P22 | NT | VDD_CORE | VDD_CORE | PTE9/LLWU_P24 | PTE13 | N |
| P | NC | NC | PTC8 | NC | VDDIO1 | PTC27 | PTD1 | NC | VSS | PTD7 | PTD10/LLWU_P20 | PTE2 | VSS | NC | NC | PTE8/LLWU_P23 | NC | P |
| R | PTC9/LLWU_P16 | PTC10 | PTC12/LLWU_P18 | NC | VDDIO1 | NC | PTC30 | NC | VDDIO1 | NC | PTD11 | NC | VSS | PTE0 | VSSA | PTE1/LLWU_P21 | PTE5 | R |
| T | PTC11/LLWU_P17 | VDD_DCDC | VOUT_RF | LN | VOUT_CORE | NC | PTD0 | PTD3 | PTD8/LLWU_P19 | NC | USB0_DP | USB0_DM | VOUT33 | NC | VREF_OUT | VSSA | DAC0_OUT | T |
| U | VDD_DCDC | LP | GND | NC | PTC28 | NC | PTD2 | NC | PTD6 | NC | PTD9 | NC | VREGIN | NC | VDDA | VREFH | VREFL | U |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | |

**Figure 23-1. VFBGA Pinout Diagram**

## 23.4 Module Signal Description Tables

The following sections correlate the chip-level signal name with the signal name used in the module's chapter. They also briefly describe the signal function and direction.

### 23.4.1 Core Modules

**Table 23-2. JTAG ARM Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| JTAG_TMS | TMS | Test Mode Select | Input |
| JTAG_TCLK | TCK | Test Clock | Input |
| JTAG_TDI | TDI | Test Data In | Input |
| JTAG_TDO | TDO | Test Data Out | Output |

**Table 23-3. RISC-V core JTAG Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| RV_JTAG_TMS | TMS | Test Mode Select | Input |
| RV_JTAG_TCLK | TCK | Test Clock | Input |
| RV_JTAG_TDI | TDI | Test Data In | Input |
| RV_JTAG_TDO | TDO | Test Data Out | Output |

**Table 23-4. SWD Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SWD_DIO | SWD_DIO | Serial Wire Data Input/Output. The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally. | Input / Output |
| SWD_CLK | SWD_CLK | Serial Wire Clock. This pin is the clock for debug logic when in the Serial Wire Debug mode. This pin is pulled down internally. | Input |
| SWD_SWO | SWD_SWO | Trace output over a single pin | Output |

**Table 23-5. TPIU Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TRACE_CLKOUT | TRACECLK | Trace clock output from the ARM CoreSight debug block | O |
| TRACE_D[3:2] | TRACEDATA | Trace output data from the ARM CoreSight debug block used for 5-pin interface | O |
| TRACE_D[1:0] | TRACEDATA | Trace output data from the ARM CoreSight debug block used for both 5-pin and 3-pin interfaces | O |

## 23.4.2 System Modules

**Table 23-6. System Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| NMI_b | — | Non-maskable interrupt<br>**NOTE:** Driving the $\overline{\text{NMI}}$ signal low forces a non-maskable interrupt, if the $\overline{\text{NMI}}$ function is selected on the corresponding pin. | I |
| RESET_b | — | Reset bi-directional signal | I/O |
| VDD_CORE | — | MCU power | I |
| VDDIO1 | — | I/O rings power | I |
| VDDIO2 | — | I/O rings power | I |

*Table continues on the next page...*

**Table 23-6. System Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VDD_RF | — | Radio power supply | I |
| VSS | — | MCU ground | I |
| VSSA | — | Analog ground | I |

**Table 23-7. EWM Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EWM_IN | EWM_in | EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active-low. | I |
| EWM_OUT_b | $\overline{\text{EWM_out}}$ | EWM reset out signal | O |

**Table 23-8. LLWU0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LLWU_Pn (n=[31:0] | LLWU_Pn (n=[31:0] | External Pin Wakeup inputs | I |

## 23.4.3 Memories and Memory Interfaces

**Table 23-9. FlexBus Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FB_CLKOUT | FB_CLK | FlexBus Clock Output | O |
| FB_A[23:16] | FB_A[23:16] | This is the address and data bus | I/O |
| FB_AD[31:0] | FB_ADn | This is the address and data bus, FB_AD.<br>The number of byte lanes carrying the data is determined by the port size associated with the matching chip-select.<br>The full 32-bit address is driven on the first clock of a bus cycle (address phase). After the first clock, the data is driven on the bus (data phase). During the data phase, the address is driven on the pins not used for data. For example, in 16-bit mode, the lower address is driven on FB_AD15–FB_AD0, and in 8-bit mode, the lower address is driven on FB_AD23–FB_AD0. | I/O |
| FB_CSn_b (n=[5:0]) | $\overline{\text{FB_CS}}$n (n=[5:0]) | General Purpose Chip-Selects—Indicate which external memory or peripheral is selected. A particular chip-select is asserted when the transfer address is within the external memory's or peripheral's address space, as defined in CSAR[BA] and CSMR[BAM]. | O |

*Table continues on the next page...*

**Table 23-9. FlexBus Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FB_BE31_24_b<br>FB_BE23_16_b<br>FB_BE15_8_b<br>FB_BE7_0_b | FB_BE_31_24,<br>FB_BE_23_16,<br>FB_BE_15_8,<br>FB_BE_7_0 | Byte Enables—Indicate that data is to be latched or driven onto a specific byte lane of the data bus. CSCR[BEM] determines if these signals are asserted on reads and writes or on writes only.<br>For external SRAM or flash devices, the FB_BE outputs should be connected to individual byte strobe signals. | O |
| FB_OE_b | FB_OE | Output Enable—Sent to the external memory or peripheral to enable a read transfer. This signal is asserted during read accesses only when a chip-select matches the current address decode. | O |
| FB_RW_b | FB_RW | Read/Write—Indicates whether the current bus operation is a read operation (FB_R/W high) or a write operation (FB_R/W low). | O |
| FB_TS_b | FB_TS | Transfer Start—Indicates that the chip has begun a bus transaction and that the address and attributes are valid. | O |
| FB_ALE | FB_ALE | Address Latch Enable—Indicates when the address is being driven on the FB_A bus (inverse of FB_TS). | O |
| FB_TSIZn (n=[1:0]) | FB_TSIZn (n=[1:0]) | Transfer Size—Indicates (along with FB_TBST) the data transfer size of the current bus operation. The interface supports 8-, 16-, and 32-bit operand transfers and allows accesses to 8-, 16-, and 32-bit data ports. | O |
| FB_TA_b | FB_TA | Transfer Acknowledge—Indicates that the external data transfer is complete. When FB_TA is asserted during a read transfer, FlexBus latches the data and then terminates the transfer. When FB_TA is asserted during a write transfer, the transfer is terminated. | I |
| FB_TBST_b | FB_TBST | Transfer Burst—Indicates that a burst transfer is in progress as driven by the chip. A burst transfer can be 2 to 16 beats depending on FB_TSIZ1–FB_TSIZ0 and the port size. | O |

## 23.4.4 Clock Modules

**Table 23-10. RTC OSC Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EXTAL32 | EXTAL32 | 32.768 kHz oscillator input | I |
| XTAL32 | XTAL32 | 32.768 kHz oscillator output | O |

**Table 23-11. Radio OSC Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| XTAL_RF | RF XTAL | 26 or 32 MHz oscillator output | O |
| EXTAL_RF | RF EXTAL | 26 or 32 MHz oscillator input | I |

*Table continues on the next page...*

**Table 23-11. Radio OSC Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| RF0_EXT_OSC_EN | RF_EXT_OSC_EN | Internal request to turn on an External Osc for use by the internal Radio, this can also be a request from the SoC if it is using the RF Osc as it's clock. | O |
| RF0_RFOSC_EN | RF_INT_OSC_EN | External request to turn on the Radio's internal RF Osc. The Osc will be present on the XTAL_OUT pin if the RF Osc XTAL_OUT buffer is enabled by writing the Radio XTAL_OUT_BUF_EN bit in the RSIM ANA_TEST register. | I |
| RF_CLKOUT | XTAL_OUT | 26 MHz/32 MHz reference clock output | O |

## 23.4.5 Communication Interfaces

**Table 23-12. EMVSIM Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EMVSIM0_CLK | EMVSIM_SCLK | Card Clock. Clock to Smart Card. | O |
| EMVSIM0_RST | EMVSIM_SRST | Card Reset. Reset signal to Smart Card | O |
| EMVSIM0_VCCEN | EMVSIM_VCC_EN | Card Power Enable. This signal controls the power to Smart Card | O |
| EMVSIM0_IO | EMVSIM_IO | Card Data Line. Bi-directional data line. | I/O |
| EMVSIM0_PD | EMVSIM_PD | Card Presence Detect. Signal indicating presence or removal of card | I |

**Table 23-13. USB FS Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| USB0_DM | usb_dm | USB D- analog data signal on the USB bus. | I/O |
| USB0_DP | usb_dp | USB D+ analog data signal on the USB bus. | I/O |
| USB_SOF_OUT | — | USB start of frame signal. Can be used to make the USB start of frame available for external synchronization. | O |
| USB0_VSS | — | $V_{SS}$ for USB0 | — |

**Table 23-14. USB VREG Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VOUT33 | reg33_out | Regulator output voltage | O |
| VREGIN | vreg_in1 | Unregulated power supply | I |

## Table 23-15. LPSPI0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI0_SCK | SCK | Serial clock. Input in slave mode, output in master mode. | I/O |
| SPI0_PCS0 | PCS[0] | Peripheral Chip Select. Input in slave mode, output in master mode. | I/O |
| SPI0_PCS1 | PCS[1] | Peripheral Chip Select or Host Request. Host Request pin is selected when HREN=1 and HRSEL=0. Input in either slave mode or when used as Host Request, output in master mode. | I/O |
| SPI0_PCS2 | PCS[2] | Peripheral Chip Select or data pin 2 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI0_PCS3 | PCS[3] | Peripheral Chip Select or data pin 3 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI0_SOUT | SOUT | Serial Data Output. Can be configured as serial data input signal. Used as data pin 0 in quad-data and dual-data transfers. | I/O |
| SPI0_SIN | SIN | Serial Data Input. Can be configured as serial data output signal. Used as data pin 1 in quad-data and dual-data transfers. | I/O |

## Table 23-16. LPSPI1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI1_SCK | SCK | Serial clock. Input in slave mode, output in master mode. | I/O |
| SPI1_PCS0 | PCS[0] | Peripheral Chip Select. Input in slave mode, output in master mode. | I/O |
| SPI1_PCS1 | PCS[1] | Peripheral Chip Select or Host Request. Host Request pin is selected when HREN=1 and HRSEL=0. Input in either slave mode or when used as Host Request, output in master mode. | I/O |
| SPI1_PCS2 | PCS[2] | Peripheral Chip Select or data pin 2 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI1_PCS3 | PCS[3] | Peripheral Chip Select or data pin 3 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI1_SOUT | SOUT | Serial Data Output. Can be configured as serial data input signal. Used as data pin 0 in quad-data and dual-data transfers. | I/O |
| SPI1_SIN | SIN | Serial Data Input. Can be configured as serial data output signal. Used as data pin 1 in quad-data and dual-data transfers. | I/O |

## Table 23-17. LPSPI2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI2_SCK | SCK | Serial clock. Input in slave mode, output in master mode. | I/O |
| SPI2_PCS0 | PCS[0] | Peripheral Chip Select. Input in slave mode, output in master mode. | I/O |

*Table continues on the next page...*

## Table 23-17. LPSPI2 Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI2_PCS1 | PCS[1] | Peripheral Chip Select or Host Request. Host Request pin is selected when HREN=1 and HRSEL=0. Input in either slave mode or when used as Host Request, output in master mode. | I/O |
| SPI2_PCS2 | PCS[2] | Peripheral Chip Select or data pin 2 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI2_PCS3 | PCS[3] | Peripheral Chip Select or data pin 3 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI2_SOUT | SOUT | Serial Data Output. Can be configured as serial data input signal. Used as data pin 0 in quad-data and dual-data transfers. | I/O |
| SPI2_SIN | SIN | Serial Data Input. Can be configured as serial data output signal. Used as data pin 1 in quad-data and dual-data transfers. | I/O |

## Table 23-18. LPSPI3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SPI3_SCK | SCK | Serial clock. Input in slave mode, output in master mode. | I/O |
| SPI3_PCS0 | PCS[0] | Peripheral Chip Select. Input in slave mode, output in master mode. | I/O |
| SPI3_PCS1 | PCS[1] | Peripheral Chip Select or Host Request. Host Request pin is selected when HREN=1 and HRSEL=0. Input in either slave mode or when used as Host Request, output in master mode. | I/O |
| SPI3_PCS2 | PCS[2] | Peripheral Chip Select or data pin 2 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI3_PCS3 | PCS[3] | Peripheral Chip Select or data pin 3 during quad-data transfers. Input in slave mode, output in master mode, input in quad-data receive transfers, output in quad-data transmit transfers. | I/O |
| SPI3_SOUT | SOUT | Serial Data Output. Can be configured as serial data input signal. Used as data pin 0 in quad-data and dual-data transfers. | I/O |
| SPI3_SIN | SIN | Serial Data Input. Can be configured as serial data output signal. Used as data pin 1 in quad-data and dual-data transfers. | I/O |

## Table 23-19. LPI2C0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2C0_SCL | SCL | LPI2C clock line. In 4-wire mode, this is the SCL input pin. | I/O |
| I2C0_SDA | SDA | LPI2C data line. In 4-wire mode, this is the SDA input pin. | I/O |
| LPI2C0_HREQ | HREQ | Host request, can initiate an LPI2C master transfer if asserted and the I2C bus is idle. | I |

*Table continues on the next page...*

## Table 23-19. LPI2C0 Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPI2C0_SCLS | SCLS | Secondary I2C clock line. In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SCL pin. | I/O |
| LPI2C0_SDAS | SDAS | Secondary I2C data line. In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SDA pin. | I/O |

## Table 23-20. LPI2C1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2C1_SCL | SCL | LPI2C clock line. In 4-wire mode, this is the SCL input pin. | I/O |
| I2C1_SDA | SDA | LPI2C data line. In 4-wire mode, this is the SDA input pin. | I/O |
| LPI2C1_HREQ | HREQ | Host request, can initiate an LPI2C master transfer if asserted and the I2C bus is idle. | I |
| LPI2C1_SCLS | SCLS | Secondary I2C clock line. In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SCL pin. | I/O |
| LPI2C1_SDAS | SDAS | Secondary I2C data line. In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SDA pin. | I/O |

## Table 23-21. LPI2C2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2C2_SCL | SCL | LPI2C clock line. In 4-wire mode, this is the SCL input pin. | I/O |
| I2C2_SDA | SDA | LPI2C data line. In 4-wire mode, this is the SDA input pin. | I/O |
| LPI2C2_HREQ | HREQ | Host request, can initiate an LPI2C master transfer if asserted and the I2C bus is idle. | I |
| LPI2C2_SCLS | SCLS | Secondary I2C clock line. In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SCL pin. | I/O |
| LPI2C2_SDAS | SDAS | Secondary I2C data line. In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SDA pin. | I/O |

## Table 23-22. LPI2C3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2C3_SCL | SCL | LPI2C clock line. In 4-wire mode, this is the SCL input pin. | I/O |
| I2C3_SDA | SDA | LPI2C data line. In 4-wire mode, this is the SDA input pin. | I/O |

*Table continues on the next page...*

## Table 23-22. LPI2C3 Signal Descriptions (continued)

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPI2C3_HREQ | HREQ | Host request, can initiate an LPI2C master transfer if asserted and the I2C bus is idle. | I |
| LPI2C3_SCLS | SCLS | Secondary I2C clock line. In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SCL pin. | I/O |
| LPI2C3_SDAS | SDAS | Secondary I2C data line. In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, this the LPI2C slave SDA pin. | I/O |

## Table 23-23. LPUART0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPUART0_CTS | CTS | Clear to Send. | I |
| LPUART0_RTS | RTS | Request to send. | O |
| LPUART0_TX | TXD | Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data. | I/O |
| LPUART0_RX | RXD | Receive data. | I |

## Table 23-24. LPUART1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPUART1_CTS | CTS | Clear to Send. | I |
| LPUART1_RTS | RTS | Request to send. | O |
| LPUART1_TX | TXD | Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data. | I/O |
| LPUART1_RX | RXD | Receive data. | I |

## Table 23-25. LPUART2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPUART2_CTS | CTS | Clear to Send. | I |
| LPUART2_RTS | RTS | Request to send. | O |
| LPUART2_TX | TXD | Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data. | I/O |
| LPUART2_RX | RXD | Receive data. | I |

## Table 23-26. LPUART3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPUART3_CTS | CTS | Clear to Send. | I |
| LPUART3_RTS | RTS | Request to send. | O |
| LPUART3_TX | TXD | Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data. | I/O |
| LPUART3_RX | RXD | Receive data. | I |

## Table 23-27. SDHC Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| SDHC0_DCLK | CLK | Clock for MMC/SD/SDIO card | O |
| SDHC0_CMD | CMD | CMD line connect to card | I/O |
| SDHC0_D0 | DAT0 | DAT0 line in all modes. Also used to detect busy state | I/O |
| SDHC0_D1 | DAT1 | DAT1 line in 4/8-bit mode.Also used to detect interrupt in 1/4-bit mode | I/O |
| SDHC0_D2 | DAT2 | DAT2 line or Read Wait in 4-bit mode. Read Wait in 1-bit mode | I/O |
| SDHC0_D3 | DAT3 | DAT3 line in 4/8-bit mode or configured as card detection pin. May be configured as card detection pin in 1-bit mode | I/O |
| SDHC0_D4 | DAT4 | DAT4 line in 8-bit mode. Not used in other modes | I/O |
| SDHC0_D5 | DAT5 | DAT5 line in 8-bit mode. Not used in other modes | I/O |
| SDHC0_D6 | DAT6 | DAT6 line in 8-bit mode. Not used in other modes | I/O |
| SDHC0_D7 | DATA7 | DAT7 line in 8-bit mode.Not used in other modes | I/O |

## Table 23-28. $I^2S0$ Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2S0_TX_BCLK | SAI_TX_BCLK | Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated. | I/O |
| I2S0_TX_FS | SAI_TX_SYNC | Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated. | I/O |
| I2S0_TX_Dn n=[1:0] | SAI_TX_DATA[1:0] | Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word. | O |
| I2S0_RX_BCLK | SAI_RX_BCLK | Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated. | I/O |
| I2S0_RX_FS | SAI_RX_SYNC | Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated. | I/O |

*Table continues on the next page...*

**Table 23-28. I²S0 Signal Descriptions (continued)**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| I2S0_RX_Dn | SAI_RX_DATA[1:0] | Receive Data. The receive data is sampled synchronously by the bit clock. | I |
| I2S0_MCLK | MCLK_EN | Audio Master Clock. | I |

**Table 23-29. FlexIO Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| FXIO0_Dn (n=[31:0] | FXIO_Dn (n=[31:0] | Bidirectional FlexIO Shifter and Timer pin inputs/outputs | I/O |

## 23.4.6 Analog

**Table 23-30. LPADC0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPADC0_SE[23:0] | CHnA (n=[23:0] | A-side Analog Channel Inputs | I |
| VDDA | VDDA | Analog Power Supply | I |
| VSSA | VSSA | Analog Ground | I |

**Table 23-31. LPCMP0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPCMP0_IN[5:0] | IN[5:0] | Analog voltage inputs | I |
| LPCMP0_OUT | CMPO | Comparator output | O |

**Table 23-32. LPCMP1 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPCMP1_IN[5:0] | IN[5:0] | Analog voltage inputs | I |
| LPCMP1_OUT | CMPO | Comparator output | O |

**Table 23-33. LPDAC0 Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| DAC0_OUT | — | DAC output | O |

### Table 23-34. VREF Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| VREF_OUT | VREF_OUT | Internally-generated Voltage Reference output | O |

## 23.4.7 Human-Machine Interfaces (HMI)

### Table 23-35. GPIO Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| PTA[31:0][1] | PORTA31–PORTA0 | General-purpose input/output | I/O |
| PTB[31:0][1] | PORTB31–PORTB0 | General-purpose input/output | I/O |
| PTC[31:0][1] | PORTC31–PORTC0 | General-purpose input/output | I/O |
| PTD[31:0][1] | PORTD31–PORTD0 | General-purpose input/output | I/O |
| PTE[31:0][1] | PORTE31–PORTE0 | General-purpose input/output | I/O |

1. The available GPIO pins depends on the specific package. See the signal multiplexing section for which exact GPIO signals are available.

## 23.4.8 Timer Modules

### Table 23-36. LPTMR0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPTMR0_ALT[3:1] | LPTMR_ALTn,n=[3:1] | Pulse Counter Input - The LPTMR can select one of the input pins to be used in Pulse Counter mode. | I |

### Table 23-37. LPTMR1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPTMR1_ALT[3:1] | LPTMR_ALTn,n=[3:1] | Pulse Counter Input - The LPTMR can select one of the input pins to be used in Pulse Counter mode. | I |

### Table 23-38. LPTMR2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| LPTMR2_ALT[3:1] | LPTMR_ALTn,n=[3:1] | Pulse Counter Input - The LPTMR can select one of the input pins to be used in Pulse Counter mode. | I |

## Table 23-39. TPM0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TPM0_CH[5:0] | TPM_CHn (n=[5:0]) | TPM channel. A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input. | I/O |
| TPM0_CLKIN | TPM_EXTCLK | External clock. TPM external clock can be selected to increment the counter on every rising edge synchronized to the counter clock. | I |

## Table 23-40. TPM1 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TPM1_CH[1:0] | TPM_CHn (n=[1:0]) | TPM channel. A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input. | I/O |
| TPM1_CLKIN | TPM_EXTCLK | External clock. TPM external clock can be selected to increment the counter on every rising edge synchronized to the counter clock. | I |

## Table 23-41. TPM2 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TPM2_CH[5:0] | TPM_CHn (n=[5:0]) | TPM channel. A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input. | I/O |
| TPM2_CLKIN | TPM_EXTCLK | External clock. TPM external clock can be selected to increment the counter on every rising edge synchronized to the counter clock. | I |

## Table 23-42. TPM3 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TPM3_CH[1:0] | TPM_CHn (n=[1:0]) | TPM channel. A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input. | I/O |
| TPM3_CLKIN | TPM_EXTCLK | External clock. TPM external clock can be selected to increment the counter on every rising edge synchronized to the counter clock. | I |

## Table 23-43. TPM0 Signal Descriptions

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| EXTAL32 | EXTAL32 | 32.768 kHz oscillator input | I |
| XTAL32 | XTAL32 | 32.768 kHz oscillator output | O |
| RTC_CLKOUT | RTC_CLKOUT | Prescaler square-wave output or 32kHz crystal clock | O |
| RTC_WAKEUP_b | $\overline{\text{RTC_WAKEUP}}$ | Active low wakeup for external device | O |

## 23.4.9 Security Modules

**Table 23-44. Tamper Detect Signal Descriptions**

| Chip signal name | Module signal name | Description | I/O |
|---|---|---|---|
| TAMPER[3:0] [1] | RTC_TAMPER[3:0] | Tamper pin input | I |

1. The TAMPER signals have dedicated pins and are not included in the JTAG boundary scan. TAMPER0 is an exception because it is priority muxed with the RTC_WAKEUP_b function. If TAMPER0 is enabled as either an input or output, the RTC_WAKEUP_b function is disabled. The RTC_WAKEUP_b pin can be configured to assert on a Tamper Detect, if the RTC enables the DryIce tamper detect interrupt.

# Chapter 24
# Port Control and Interrupts (PORT)

## 24.1 Introduction

### 24.1.1 Overview

The Port Control and Interrupt (PORT) module provides support for port control, and external interrupt functions.

Most functions can be configured independently for each pin in the 32-bit port and affect the pin regardless of its pin muxing state.

There is one instance of the PORT module for each port. Not all pins within each port are implemented on a specific device.

### 24.1.2 Features

The PORT module has the following features:

- Pin interrupt on selected pins
  - Interrupt flag and enable registers for each pin
  - Support for edge sensitive (rising, falling, both) or level sensitive (low, high) configured per pin
  - Support for interrupt or DMA request configured per pin
  - Asynchronous wake-up in low-power modes
  - Pin interrupt is functional in all digital pin muxing modes
  - Peripheral trigger output (active high, low) configured per pin
- Port control
  - Individual pull control fields with pullup, pulldown, and pull-disable support
  - Individual slew rate field supporting fast and slow slew rates
  - Individual input passive filter field supporting enable and disable of the individual input passive filter on selected pins

- Individual open drain field supporting enable and disable of the individual open drain output
- Individual mux control field supporting analog or pin disabled, GPIO, and up to 6 chip-specific digital functions
- Pad configuration fields are functional in all digital pin muxing modes

### 24.1.3 Modes of operation

#### 24.1.3.1 Run mode

In Run mode, the PORT operates normally.

#### 24.1.3.2 Wait mode

In Wait mode, PORT continues to operate normally and may be configured to exit the Low-Power mode if an enabled interrupt is detected. DMA requests are still generated during the Wait mode, but do not cause an exit from the Low-Power mode.

#### 24.1.3.3 Stop mode

In Stop mode, the PORT can be configured to exit the Low-Power mode via an asynchronous wake-up signal if an enabled interrupt is detected.

In Stop mode, the PORT configuration is static.

#### 24.1.3.4 Debug mode

In Debug mode, PORT operates normally.

## 24.2 External signal description

The table found here describes the PORT external signal.

**Table 24-1. Signal properties**

| Name | Function | I/O | Reset | Pull |
|---|---|---|---|---|
| PORTx[31:0] | External interrupt | I/O | 0 | - |

**NOTE**

Not all pins within each port are implemented on each device.

### 24.2.1 Detailed signal description

The table found here contains the detailed signal description for the PORT interface.

**Table 24-2. PORT interface—detailed signal description**

| Signal | I/O | Description | |
|---|---|---|---|
| PORTx[31:0] | I/O | External interrupt. | |
| | | State meaning | Asserted—pin is logic 1.<br>Negated—pin is logic 0. |
| | | Timing | Assertion—may occur at any time and can assert asynchronously to the system clock.<br>Negation—may occur at any time and can assert asynchronously to the system clock. |

## 24.3 Memory map and register definition

Any read or write access to the PORT memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states.

### 24.3.1 PORTA register descriptions

#### 24.3.1.1 PORTA Memory map

PORTA base address: 4004_6000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Pin Control Register 0 (PCR0) | 32 | RW | 0000_0703h |
| 4h | Pin Control Register 1 (PCR1) | 32 | RW | 0000_0702h |
| 8h - 10h | Pin Control Register a (PCR2 - PCR4) | 32 | RW | 0000_0703h |
| 24h | Pin Control Register 9 (PCR9) | 32 | RW | 0000_0000h |
| 28h | Pin Control Register 10 (PCR10) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 38h | Pin Control Register 14 (PCR14) | 32 | RW | 0000_0000h |
| 3Ch | Pin Control Register 15 (PCR15) | 32 | RW | 0000_0000h |
| 44h | Pin Control Register 17 (PCR17) | 32 | RW | 0000_0000h |
| 48h | Pin Control Register 18 (PCR18) | 32 | RW | 0000_0000h |
| 4Ch | Pin Control Register 19 (PCR19) | 32 | RW | 0000_0000h |
| 50h | Pin Control Register 20 (PCR20) | 32 | RW | 0000_0000h |
| 54h | Pin Control Register 21 (PCR21) | 32 | RW | 0000_0000h |
| 58h | Pin Control Register 22 (PCR22) | 32 | RW | 0000_0000h |
| 5Ch | Pin Control Register 23 (PCR23) | 32 | RW | 0000_0000h |
| 60h | Pin Control Register 24 (PCR24) | 32 | RW | 0000_0000h |
| 64h | Pin Control Register 25 (PCR25) | 32 | RW | 0000_0000h |
| 68h | Pin Control Register 26 (PCR26) | 32 | RW | 0000_0000h |
| 6Ch | Pin Control Register 27 (PCR27) | 32 | RW | 0000_0000h |
| 70h | Pin Control Register 28 (PCR28) | 32 | RW | 0000_0000h |
| 78h | Pin Control Register 30 (PCR30) | 32 | RW | 0000_0000h |
| 7Ch | Pin Control Register 31 (PCR31) | 32 | RW | 0000_0000h |
| 80h | Global Pin Control Low Register (GPCLR) | 32 | WORZ | 0000_0000h |
| 84h | Global Pin Control High Register (GPCHR) | 32 | WORZ | 0000_0000h |
| 88h | Global Interrupt Control Low Register (GICLR) | 32 | WORZ | 0000_0000h |
| 8Ch | Global Interrupt Control High Register (GICHR) | 32 | WORZ | 0000_0000h |
| A0h | Interrupt Status Flag Register (ISFR) | 32 | W1C | 0000_0000h |

## 24.3.1.2 Pin Control Register 0 (PCR0)

### 24.3.1.2.1 Offset

| Register | Offset |
|---|---|
| PCR0 | 0h |

### 24.3.1.2.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | PFE | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 24.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>PFE | Passive Filter Enable<br>Passive filter configuration is valid in all digital pin muxing modes.<br>0b - Passive input filter is disabled on the corresponding pin. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. Refer to the device data sheet for filter characteristics. |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.1.3 Pin Control Register 1 (PCR1)

### 24.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PCR1 | 4h |

### 24.3.1.3.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

### 24.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2 | Slew Rate Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SRE | Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1 | Pull Enable |
| PE | Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0 | Pull Select |
| PS | Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.1.4 Pin Control Register a (PCR2 - PCR4)

### 24.3.1.4.1 Offset

For a = 2 to 4:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.1.4.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 24.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1 | Pull Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| PE | Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.1.5 Pin Control Register a (PCR9 - PCR31)

### 24.3.1.5.1 Offset

For a = 9 to 31:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.1.5.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1 | Pull Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| PE | Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.1.6 Global Pin Control Low Register (GPCLR)

### 24.3.1.6.1 Offset

| Register | Offset |
|---|---|
| GPCLR | 80h |

### 24.3.1.6.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.1.6.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| GPWE | Selects which Pin Control Registers (15 through 0) bits update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n). |

## 24.3.1.7 Global Pin Control High Register (GPCHR)

### 24.3.1.7.1 Offset

| Register | Offset |
|---|---|
| GPCHR | 84h |

### 24.3.1.7.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.1.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GPWE | Global Pin Write Enable<br>Selects which Pin Control Registers (31 through 16) update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n + 16). |

## 24.3.1.8 Global Interrupt Control Low Register (GICLR)

### 24.3.1.8.1 Offset

| Register | Offset |
|---|---|
| GICLR | 88h |

### 24.3.1.8.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.1.8.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GIWD | Global Interrupt Write Data<br>Write value that is written to upper 16-bits of the Pin Control Registers bits that are selected by GIWE. |
| 15-0<br>GIWE | Global Interrupt Write Enable<br>Selects which upper 16-bits of Pin Control Registers (15 through 0) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Control Register is updated with the value in GIWD. |

## 24.3.1.9 Global Interrupt Control High Register (GICHR)

### 24.3.1.9.1 Offset

| Register | Offset |
|---|---|
| GICHR | 8Ch |

### 24.3.1.9.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (31 through 16) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Register is updated with the value in GIWD. |

## 24.3.1.10 Interrupt Status Flag Register (ISFR)

#### 24.3.1.10.1 Offset

| Register | Offset |
|---|---|
| ISFR | A0h |

#### 24.3.1.10.2 Function

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

#### 24.3.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.1.10.4 Fields

| Field | Function |
|---|---|
| 31-0 | Interrupt Status Flag |
| ISF | Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br><br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 24.3.2 PORTB register descriptions

### 24.3.2.1 PORTB Memory map

PORTB base address: 4004_7000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Pin Control Register 0 (PCR0) | 32 | RW | 0000_0000h |
| 4h | Pin Control Register 1 (PCR1) | 32 | RW | 0000_0000h |
| 8h | Pin Control Register 2 (PCR2) | 32 | RW | 0000_0000h |
| Ch | Pin Control Register 3 (PCR3) | 32 | RW | 0000_0000h |
| 10h | Pin Control Register 4 (PCR4) | 32 | RW | 0000_0000h |
| 14h | Pin Control Register 5 (PCR5) | 32 | RW | 0000_0000h |
| 18h | Pin Control Register 6 (PCR6) | 32 | RW | 0000_0000h |
| 1Ch | Pin Control Register 7 (PCR7) | 32 | RW | 0000_0000h |
| 20h | Pin Control Register 8 (PCR8) | 32 | RW | 0000_0000h |
| 24h | Pin Control Register 9 (PCR9) | 32 | RW | 0000_0000h |
| 2Ch | Pin Control Register 11 (PCR11) | 32 | RW | 0000_0000h |
| 30h | Pin Control Register 12 (PCR12) | 32 | RW | 0000_0000h |
| 34h | Pin Control Register 13 (PCR13) | 32 | RW | 0000_0000h |
| 38h | Pin Control Register 14 (PCR14) | 32 | RW | 0000_0000h |
| 3Ch | Pin Control Register 15 (PCR15) | 32 | RW | 0000_0000h |
| 40h | Pin Control Register 16 (PCR16) | 32 | RW | 0000_0000h |
| 44h | Pin Control Register 17 (PCR17) | 32 | RW | 0000_0000h |
| 48h | Pin Control Register 18 (PCR18) | 32 | RW | 0000_0000h |
| 4Ch | Pin Control Register 19 (PCR19) | 32 | RW | 0000_0000h |
| 50h | Pin Control Register 20 (PCR20) | 32 | RW | 0000_0000h |
| 54h | Pin Control Register 21 (PCR21) | 32 | RW | 0000_0000h |
| 58h | Pin Control Register 22 (PCR22) | 32 | RW | 0000_0000h |
| 60h | Pin Control Register 24 (PCR24) | 32 | RW | 0000_0000h |
| 64h | Pin Control Register 25 (PCR25) | 32 | RW | 0000_0000h |
| 68h | Pin Control Register 26 (PCR26) | 32 | RW | 0000_0000h |
| 70h | Pin Control Register 28 (PCR28) | 32 | RW | 0000_0000h |
| 74h | Pin Control Register 29 (PCR29) | 32 | RW | 0000_0000h |
| 78h | Pin Control Register 30 (PCR30) | 32 | RW | 0000_0000h |
| 7Ch | Pin Control Register 31 (PCR31) | 32 | RW | 0000_0000h |
| 80h | Global Pin Control Low Register (GPCLR) | 32 | WORZ | 0000_0000h |
| 84h | Global Pin Control High Register (GPCHR) | 32 | WORZ | 0000_0000h |
| 88h | Global Interrupt Control Low Register (GICLR) | 32 | WORZ | 0000_0000h |
| 8Ch | Global Interrupt Control High Register (GICHR) | 32 | WORZ | 0000_0000h |
| A0h | Interrupt Status Flag Register (ISFR) | 32 | W1C | 0000_0000h |

## 24.3.2.2 Pin Control Register a (PCR0 - PCR31)

### 24.3.2.2.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.2.2.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 24.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8 | Pin Mux Control |

*Table continues on the next page...*

| Field | Function |
|---|---|
| MUX | Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.2.3 Global Pin Control Low Register (GPCLR)

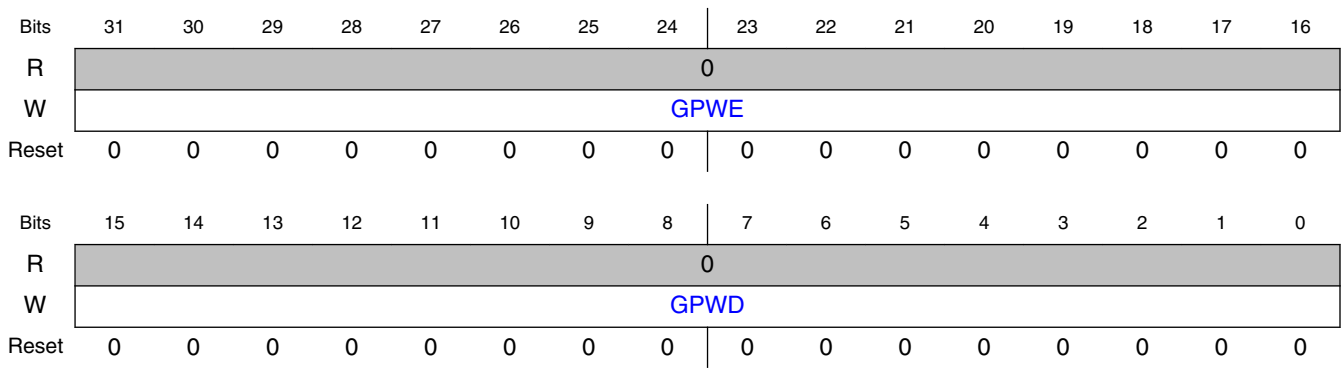
#### 24.3.2.3.1 Offset

| Register | Offset |
|---|---|
| GPCLR | 80h |

#### 24.3.2.3.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.2.3.3 Diagram




#### 24.3.2.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GPWE | Global Pin Write Enable<br>Selects which Pin Control Registers (15 through 0) bits update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n). |

### 24.3.2.4 Global Pin Control High Register (GPCHR)

#### 24.3.2.4.1 Offset

| Register | Offset |
|---|---|
| GPCHR | 84h |

#### 24.3.2.4.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.2.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.2.4.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |
| GPWE | Selects which Pin Control Registers (31 through 16) update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0 | Global Pin Write Data |
| GPWD | GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n + 16). |

### 24.3.2.5 Global Interrupt Control Low Register (GICLR)

#### 24.3.2.5.1 Offset

| Register | Offset |
|---|---|
| GICLR | 88h |

#### 24.3.2.5.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.2.5.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers bits that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (15 through 0) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Control Register is updated with the value in GIWD. |

### 24.3.2.6 Global Interrupt Control High Register (GICHR)

#### 24.3.2.6.1 Offset

| Register | Offset |
|---|---|
| GICHR | 8Ch |

#### 24.3.2.6.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.2.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.2.6.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (31 through 16) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Register is updated with the value in GIWD. |

## 24.3.2.7 Interrupt Status Flag Register (ISFR)

### 24.3.2.7.1 Offset

| Register | Offset |
|---|---|
| ISFR | A0h |

### 24.3.2.7.2 Function

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

#### 24.3.2.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.2.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ISF | Interrupt Status Flag<br>Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 24.3.3 PORTC register descriptions

### 24.3.3.1 PORTC Memory map

PORTC base address: 4004_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 4h | Pin Control Register a (PCR0 - PCR1) | 32 | RW | 0000_0000h |
| 1Ch - 30h | Pin Control Register a (PCR7 - PCR12) | 32 | RW | 0000_0000h |
| 68h - 78h | Pin Control Register a (PCR26 - PCR30) | 32 | RW | 0000_0000h |
| 80h | Global Pin Control Low Register (GPCLR) | 32 | WORZ | 0000_0000h |
| 84h | Global Pin Control High Register (GPCHR) | 32 | WORZ | 0000_0000h |
| 88h | Global Interrupt Control Low Register (GICLR) | 32 | WORZ | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8Ch | Global Interrupt Control High Register (GICHR) | 32 | WORZ | 0000_0000h |
| A0h | Interrupt Status Flag Register (ISFR) | 32 | W1C | 0000_0000h |

## 24.3.3.2 Pin Control Register a (PCR0 - PCR1)

### 24.3.3.2.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.3.2.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.3.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.3.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.3.3 Pin Control Register a (PCR7 - PCR12)

### 24.3.3.3.1 Offset

For a = 7 to 12:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.3.3.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.3.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | DSE | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.3.3.4 Fields

| Field | Function |
|---|---|
| 31 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>DSE | Drive Strength Enable<br>Drive strength configuration is valid in all digital pin muxing modes.<br>0b - Low drive strength is configured on the corresponding pin, if pin is configured as a digital output.<br>1b - High drive strength is configured on the corresponding pin, if pin is configured as a digital output. |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.3.4 Pin Control Register a (PCR26 - PCR30)

#### 24.3.3.4.1 Offset

For a = 26 to 30:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

#### 24.3.3.4.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

#### 24.3.3.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.3.4.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.3.5 Global Pin Control Low Register (GPCLR)

### 24.3.3.5.1 Offset

| Register | Offset |
|---|---|
| GPCLR | 80h |

#### 24.3.3.5.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.3.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.3.5.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |
| GPWE | Selects which Pin Control Registers (15 through 0) bits update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0 | Global Pin Write Data |
| GPWD | GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n). |

### 24.3.3.6 Global Pin Control High Register (GPCHR)

#### 24.3.3.6.1 Offset

| Register | Offset |
|---|---|
| GPCHR | 84h |

#### 24.3.3.6.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.3.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.3.6.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |
| GPWE | Selects which Pin Control Registers (31 through 16) update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0 | Global Pin Write Data |
| GPWD | GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n + 16). |

## 24.3.3.7 Global Interrupt Control Low Register (GICLR)

### 24.3.3.7.1 Offset

| Register | Offset |
|---|---|
| GICLR | 88h |

### 24.3.3.7.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.3.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.3.7.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers bits that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (15 through 0) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Control Register is updated with the value in GIWD. |

### 24.3.3.8 Global Interrupt Control High Register (GICHR)

#### 24.3.3.8.1 Offset

| Register | Offset |
|---|---|
| GICHR | 8Ch |

#### 24.3.3.8.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.3.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.3.8.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (31 through 16) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Register is updated with the value in GIWD. |

## 24.3.3.9 Interrupt Status Flag Register (ISFR)

### 24.3.3.9.1 Offset

| Register | Offset |
|---|---|
| ISFR | A0h |

### 24.3.3.9.2 Function

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

#### 24.3.3.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.3.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ISF | Interrupt Status Flag<br>Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 24.3.4 PORTD register descriptions

### 24.3.4.1 PORTD Memory map

PORTD base address: 4004_9000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 1Ch | Pin Control Register a (PCR0 - PCR7) | 32 | RW | 0000_0000h |
| 20h - 2Ch | Pin Control Register a (PCR8 - PCR11) | 32 | RW | 0000_0000h |
| 80h | Global Pin Control Low Register (GPCLR) | 32 | WORZ | 0000_0000h |
| 84h | Global Pin Control High Register (GPCHR) | 32 | WORZ | 0000_0000h |
| 88h | Global Interrupt Control Low Register (GICLR) | 32 | WORZ | 0000_0000h |
| 8Ch | Global Interrupt Control High Register (GICHR) | 32 | WORZ | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| A0h | Interrupt Status Flag Register (ISFR) | 32 | W1C | 0000_0000h |
| C0h | Digital Filter Enable Register (DFER) | 32 | RW | 0000_0000h |
| C4h | Digital Filter Clock Register (DFCR) | 32 | RW | 0000_0000h |
| C8h | Digital Filter Width Register (DFWR) | 32 | RW | 0000_0000h |

## 24.3.4.2 Pin Control Register a (PCR0 - PCR7)

### 24.3.4.2.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.4.2.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.4.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.4.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1 | Pull Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| PE | Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.4.3 Pin Control Register a (PCR8 - PCR11)

### 24.3.4.3.1 Offset

For a = 8 to 11:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.4.3.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

## 24.3.4.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | DSE | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 24.3.4.3.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>DSE | Drive Strength Enable<br>Drive strength configuration is valid in all digital pin muxing modes.<br>0b - Low drive strength is configured on the corresponding pin, if pin is configured as a digital output.<br>1b - High drive strength is configured on the corresponding pin, if pin is configured as a digital output. |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.4.4 Global Pin Control Low Register (GPCLR)

### 24.3.4.4.1 Offset

| Register | Offset |
|---|---|
| GPCLR | 80h |

### 24.3.4.4.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.4.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.4.4.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GPWE | Global Pin Write Enable<br>Selects which Pin Control Registers (15 through 0) bits update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n). |

### 24.3.4.5 Global Pin Control High Register (GPCHR)

#### 24.3.4.5.1 Offset

| Register | Offset |
|---|---|
| GPCHR | 84h |

#### 24.3.4.5.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.4.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.4.5.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| GPWE | Selects which Pin Control Registers (31 through 16) update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n + 16). |

## 24.3.4.6 Global Interrupt Control Low Register (GICLR)

### 24.3.4.6.1 Offset

| Register | Offset |
|---|---|
| GICLR | 88h |

### 24.3.4.6.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.4.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.4.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GIWD | Global Interrupt Write Data<br>Write value that is written to upper 16-bits of the Pin Control Registers bits that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |

| Field | Function |
|---|---|
| GIWE | Selects which upper 16-bits of Pin Control Registers (15 through 0) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Control Register is updated with the value in GIWD. |

## 24.3.4.7 Global Interrupt Control High Register (GICHR)

### 24.3.4.7.1 Offset

| Register | Offset |
|---|---|
| GICHR | 8Ch |

### 24.3.4.7.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.4.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.4.7.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (31 through 16) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Register is updated with the value in GIWD. |

## 24.3.4.8 Interrupt Status Flag Register (ISFR)

### 24.3.4.8.1 Offset

| Register | Offset |
|---|---|
| ISFR | A0h |

### 24.3.4.8.2 Function

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

### 24.3.4.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.4.8.4 Fields

| Field | Function |
|---|---|
| 31-0 | Interrupt Status Flag |
| ISF | Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br><br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 24.3.4.9 Digital Filter Enable Register (DFER)

### 24.3.4.9.1 Offset

| Register | Offset |
|---|---|
| DFER | C0h |

### 24.3.4.9.2 Function

The digital filter configuration is valid in all digital pin muxing modes.

### 24.3.4.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.4.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DFE | Digital Filter Enable<br>The digital filter configuration is valid in all digital pin muxing modes. The output of each digital filter is reset to zero at system reset and whenever the digital filter is disabled. Each bit in the field enables the digital filter of the same number as the field.<br>0b - Digital filter is disabled on the corresponding pin and output of the digital filter is reset to zero.<br>1b - Digital filter is enabled on the corresponding pin, if the pin is configured as a digital input. |

## 24.3.4.10 Digital Filter Clock Register (DFCR)

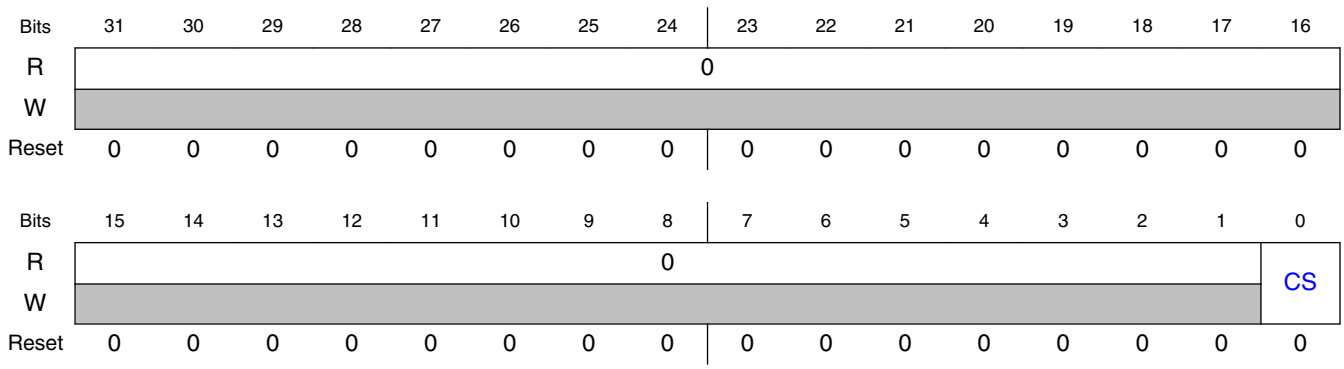
### 24.3.4.10.1 Offset

| Register | Offset |
|---|---|
| DFCR | C4h |

### 24.3.4.10.2 Function

The digital filter configuration is valid in all digital pin muxing modes.

### 24.3.4.10.3 Diagram




### 24.3.4.10.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>CS | Clock Source<br>The digital filter configuration is valid in all digital pin muxing modes. Configures the clock source for the digital input filters. Changing the filter clock source must be done only when all digital filters are disabled.<br>0b - Digital filters are clocked by the bus clock.<br>1b - Digital filters are clocked by the 8 clock. |

## 24.3.4.11 Digital Filter Width Register (DFWR)

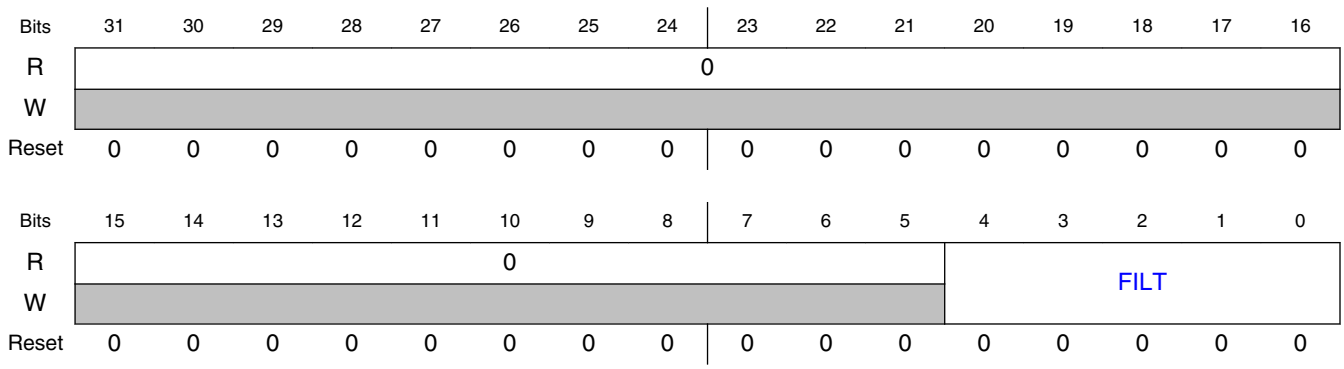
#### 24.3.4.11.1 Offset

| Register | Offset |
|---|---|
| DFWR | C8h |

#### 24.3.4.11.2 Function

The digital filter configuration is valid in all digital pin muxing modes.

#### 24.3.4.11.3 Diagram




#### 24.3.4.11.4 Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>FILT | Filter Length<br>The digital filter configuration is valid in all digital pin muxing modes. Configures the maximum size of the glitches, in clock cycles, that the digital filter absorbs for the enabled digital filters. Glitches that are longer than this register setting will pass through the digital filter, and glitches that are equal to or less than this register setting are filtered. Changing the filter length must be done only after all filters are disabled. |

## 24.3.5 PORTE register descriptions

### 24.3.5.1 PORTE Memory map

PORTE base address: 4103_7000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Pin Control Register 0 (PCR0) | 32 | RW | 0000_0000h |
| 4h | Pin Control Register 1 (PCR1) | 32 | RW | 0000_0000h |
| 8h | Pin Control Register 2 (PCR2) | 32 | RW | 0000_0000h |
| Ch | Pin Control Register 3 (PCR3) | 32 | RW | 0000_0000h |
| 10h | Pin Control Register 4 (PCR4) | 32 | RW | 0000_0000h |
| 14h | Pin Control Register 5 (PCR5) | 32 | RW | 0000_0000h |
| 20h | Pin Control Register 8 (PCR8) | 32 | RW | 0000_0000h |
| 24h | Pin Control Register 9 (PCR9) | 32 | RW | 0000_0000h |
| 28h - 2Ch | Pin Control Register a (PCR10 - PCR11) | 32 | RW | 0000_0000h |
| 30h | Pin Control Register 12 (PCR12) | 32 | RW | 0000_0000h |
| 34h | Pin Control Register 13 (PCR13) | 32 | RW | 0000_0000h |
| 38h | Pin Control Register 14 (PCR14) | 32 | RW | 0000_0000h |
| 3Ch | Pin Control Register 15 (PCR15) | 32 | RW | 0000_0000h |
| 40h | Pin Control Register 16 (PCR16) | 32 | RW | 0000_0000h |
| 44h | Pin Control Register 17 (PCR17) | 32 | RW | 0000_0000h |
| 48h | Pin Control Register 18 (PCR18) | 32 | RW | 0000_0000h |
| 4Ch | Pin Control Register 19 (PCR19) | 32 | RW | 0000_0000h |
| 54h | Pin Control Register 21 (PCR21) | 32 | RW | 0000_0000h |
| 58h | Pin Control Register 22 (PCR22) | 32 | RW | 0000_0000h |
| 6Ch | Pin Control Register 27 (PCR27) | 32 | RW | 0000_0000h |
| 70h | Pin Control Register 28 (PCR28) | 32 | RW | 0000_0000h |
| 74h | Pin Control Register 29 (PCR29) | 32 | RW | 0000_0000h |
| 78h | Pin Control Register 30 (PCR30) | 32 | RW | 0000_0000h |
| 80h | Global Pin Control Low Register (GPCLR) | 32 | WORZ | 0000_0000h |
| 84h | Global Pin Control High Register (GPCHR) | 32 | WORZ | 0000_0000h |
| 88h | Global Interrupt Control Low Register (GICLR) | 32 | WORZ | 0000_0000h |
| 8Ch | Global Interrupt Control High Register (GICHR) | 32 | WORZ | 0000_0000h |
| A0h | Interrupt Status Flag Register (ISFR) | 32 | W1C | 0000_0000h |

### 24.3.5.2 Pin Control Register a (PCR0 - PCR9)

#### 24.3.5.2.1 Offset

For a = 0 to 9:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

#### 24.3.5.2.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

#### 24.3.5.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.5.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23<br>— | Reserved. |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 6<br>— | Reserved |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.5.3 Pin Control Register a (PCR10 - PCR11)

### 24.3.5.3.1 Offset

For a = 10 to 11:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.5.3.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.5.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | DSE | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.5.3.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>DSE | Drive Strength Enable<br>Drive strength configuration is valid in all digital pin muxing modes.<br>0b - Low drive strength is configured on the corresponding pin, if pin is configured as a digital output. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - High drive strength is configured on the corresponding pin, if pin is configured as a digital output. |
| 5<br>ODE | Open Drain Enable<br>Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.5.4 Pin Control Register a (PCR12 - PCR30)

### 24.3.5.4.1 Offset

For a = 12 to 30:

| Register | Offset |
|---|---|
| PCRa | 0h + (a × 4h) |

### 24.3.5.4.2 Function

**NOTE**

See the Signal Multiplexing and Pin Assignment chapter for the reset values for the pins on this device.

See the Chip specific PORT information section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 24.3.5.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | ISF | 0 | 0 | | | IRQC | | | |
| W | | | | | | | | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | 0 | | | MUX | | | 0 | 0 | ODE | 0 | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.5.4.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-25<br>— | Reserved |
| 24<br>ISF | Interrupt Status Flag<br>The pin interrupt configuration is valid in all digital pin muxing modes.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |
| 23 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 22-20<br>— | Reserved |
| 19-16<br>IRQC | Interrupt Configuration<br>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt, DMA request or trigger as follows:<br>0000b - Interrupt Status Flag (ISF) is disabled.<br>0001b - ISF flag and DMA request on rising edge.<br>0010b - ISF flag and DMA request on falling edge.<br>0011b - ISF flag and DMA request on either edge.<br>0100b - Reserved.<br>0101b - Flag sets on rising edge.<br>0110b - Flag sets on falling edge.<br>0111b - Flag sets on either edge.<br>1000b - ISF flag and Interrupt when logic 0.<br>1001b - ISF flag and Interrupt on rising-edge.<br>1010b - ISF flag and Interrupt on falling-edge.<br>1011b - ISF flag and Interrupt on either edge.<br>1100b - ISF flag and Interrupt when logic 1.<br>1101b - Enable active high trigger output, flag is disabled. [The trigger output goes to the trigger mux, which allows pins to trigger other peripherals (configurable polarity; 1 pin per port; if multiple pins are configured, then they are ORed together to create the trigger)]<br>1110b - Enable active low trigger output, flag is disabled.<br>1111b - Reserved. |
| 15<br>LK | Lock Register<br>0b - Pin Control Register is not locked.<br>1b - Pin Control Register is locked and cannot be updated until the next system reset. |
| 14<br>— | Reserved. |
| 13-11<br>— | Reserved |
| 10-8<br>MUX | Pin Mux Control<br>Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot.<br>The corresponding pin is configured in the following pin muxing slot as follows:<br>000b - Pin disabled (Alternative 0) (analog).<br>001b - Alternative 1 (GPIO).<br>010b - Alternative 2 (chip-specific).<br>011b - Alternative 3 (chip-specific).<br>100b - Alternative 4 (chip-specific).<br>101b - Alternative 5 (chip-specific).<br>110b - Alternative 6 (chip-specific).<br>111b - Alternative 7 (chip-specific). |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5 | Open Drain Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ODE | Open drain configuration is valid in all digital pin muxing modes.<br>0b - Open drain output is disabled on the corresponding pin.<br>1b - Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SRE | Slew Rate Enable<br>Slew rate configuration is valid in all digital pin muxing modes.<br>0b - Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output.<br>1b - Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output. |
| 1<br>PE | Pull Enable<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pull resistor is not enabled on the corresponding pin.<br>1b - Internal pull resistor is enabled on the corresponding pin, if the pin is configured as a digital input. |
| 0<br>PS | Pull Select<br>Pull configuration is valid in all digital pin muxing modes.<br>0b - Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set.<br>1b - Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set. |

## 24.3.5.5 Global Pin Control Low Register (GPCLR)

### 24.3.5.5.1 Offset

| Register | Offset |
|---|---|
| GPCLR | 80h |

### 24.3.5.5.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.5.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.5.5.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Pin Write Enable |
| GPWE | Selects which Pin Control Registers (15 through 0) bits update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0 | Global Pin Write Data |
| GPWD | GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n). |

## 24.3.5.6 Global Pin Control High Register (GPCHR)

### 24.3.5.6.1 Offset

| Register | Offset |
|---|---|
| GPCHR | 84h |

### 24.3.5.6.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.5.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.5.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>GPWE | Global Pin Write Enable<br>Selects which Pin Control Registers (31 through 16) update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.<br>0b - Corresponding lower 16-bits of Pin Control Register is not updated with the value in GPWD.<br>1b - Corresponding lower 16-bits of Pin Control Register is updated with the value in GPWD. |
| 15-0<br>GPWD | Global Pin Write Data<br>GPWD field is written to PCRa[15:0] if GPWEn = 1 (a = n + 16). |

### 24.3.5.7 Global Interrupt Control Low Register (GICLR)

#### 24.3.5.7.1 Offset

| Register | Offset |
|---|---|
| GICLR | 88h |

#### 24.3.5.7.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

### 24.3.5.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.5.7.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers bits that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (15 through 0) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Control Register is updated with the value in GIWD. |

## 24.3.5.8 Global Interrupt Control High Register (GICHR)

### 24.3.5.8.1 Offset

| Register | Offset |
|---|---|
| GICHR | 8Ch |

### 24.3.5.8.2 Function

Only 32-bit writes are supported to this register, any 16-bit or 8-bit writes are ignored.

#### 24.3.5.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 24.3.5.8.4 Fields

| Field | Function |
|---|---|
| 31-16 | Global Interrupt Write Data |
| GIWD | Write value that is written to upper 16-bits of the Pin Control Registers that are selected by GIWE. |
| 15-0 | Global Interrupt Write Enable |
| GIWE | Selects which upper 16-bits of Pin Control Registers (31 through 16) update with the value in GIWD.<br>0b - Corresponding upper 16-bits of Pin Control Register is not updated with the value in GIWD.<br>1b - Corresponding upper 16-bits of Pin Register is updated with the value in GIWD. |

### 24.3.5.9 Interrupt Status Flag Register (ISFR)

#### 24.3.5.9.1 Offset

| Register | Offset |
|---|---|
| ISFR | A0h |

#### 24.3.5.9.2 Function

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

### 24.3.5.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 24.3.5.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ISF | Interrupt Status Flag<br>Each bit in the field indicates the detection of the configured interrupt of the same number as the field.<br>0b - Configured interrupt is not detected.<br>1b - Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared. |

## 24.4 Functional description

### 24.4.1 Pin control

Each port pin has a corresponding Pin Control register, PORT_PCRn, associated with it. The Pin Control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable
- Slew rate configuration
- Open drain enable
- Passive input filter enable on selected pins
- Software configuration lock
- Pin Muxing mode

The functions apply across all digital pin muxing modes and individual peripherals do not override the configuration in the Pin Control register. For example, if an $I^2C$ function is enabled on a pin, that does not override the pullup or open drain configuration for that pin.

When the Pin Muxing mode is configured for analog or is disabled, all the digital functions on that pin are disabled. This includes the pullup and pulldown enables, output buffer enable, input buffer enable, and passive filter enable.

The LK bit (bit 15 of Pin Control Register PCR*n*) allows the configuration for each pin to be locked until the next system reset. When locked, writes to the lower half of that pin control register are ignored, although a bus error is not generated on an attempted write to a locked register.

The configuration of each Pin Control register is retained when the PORT module is disabled.

Whenever a pin is configured in any digital pin muxing mode, the input buffer for that pin is enabled allowing the pin state to be read via the corresponding GPIO Port Data Input Register (GPIO_PDIR) or allowing a pin interrupt or DMA request to be generated. If a pin is ever floating when its input buffer is enabled, then this can cause an increase in power consumption and must be avoided. A pin can be floating due to an input pin that is not connected or an output pin that has tri-stated (output buffer is disabled).

Enabling the internal pull resistor (or implementing an external pull resistor) will ensure a pin does not float when its input buffer is enabled; note that the internal pull resistor is automatically disabled whenever the output buffer is enabled allowing the Pull Enable bit to remain set. Configuring the Pin Muxing mode to disabled or analog will disable the pin's input buffer and results in the lowest power consumption.

## 24.4.2 Global pin control

The two global pin control registers allow a single register write to update the lower 16-bits of the Pin Control Register for up to 16 pins, all with the same value. Registers that are locked cannot be written using the global pin control registers.

The global pin control registers are designed to enable software to quickly configure multiple pins within the same port with the same peripheral function.

The global pin control registers are write-only registers, that always read as 0.

### 24.4.3 Global interrupt control

The two global interrupt control registers allow a single register write to update the upper 16-bits of the Pin Control Register for up to 16 pins, all with the same value.

The global interrupt control registers are designed to enable software to quickly configure multiple pins within the same port with the same interrupt configuration.

The global interrupt control registers are write-only registers and always read as 0.

### 24.4.4 External interrupts

The external interrupt capability of the PORT module is available in all digital pin muxing modes provided the PORT module is enabled.

Each pin can be individually configured for any of the following external interrupt modes:

- Interrupt disabled, default out of reset
- Rising edge flag (for software polling)
- Falling edge flag (for software polling)
- Rising and falling edge flag (for software polling)
- Active high level peripheral trigger (status flag disabled)
- Active low level peripheral trigger (status flag disabled)
- Active high level sensitive interrupt
- Active low level sensitive interrupt
- Rising edge sensitive interrupt
- Falling edge sensitive interrupt
- Rising and falling edge sensitive interrupt
- Rising edge sensitive DMA request
- Falling edge sensitive DMA request
- Rising and falling edge sensitive DMA request

The interrupt status flag is set when the configured edge or level is detected on the pin . When not in Stop mode, the input is first synchronized to the bus clock to detect the configured level or edge transition.

The PORT module generates a single pin interrupt that asserts when the interrupt status flag is set for any enabled interrupt for that port. The interrupt negates after the interrupt status flags for all enabled interrupts have been cleared by writing a logic 1 to the ISF flag in either the PORT_ISFR or PORT_PCRn registers.

The PORT module generates a single DMA request that asserts when the interrupt status flag is set for any enabled DMA request in that port. The DMA request negates after the DMA transfer is completed, because that clears the interrupt status flags for all enabled DMA requests.

During Stop mode, the interrupt status flag for any enabled interrupt is asynchronously set if the required level or edge is detected. This also generates an asynchronous wake-up signal to exit the Low-Power mode.

The PORT module generates a single peripheral trigger output that asserts if any pin configured for active high trigger is logic one, or any pin triggered for active low trigger is logic zero. The peripheral trigger output asynchronously updates from the value on the configured pins.

# Chapter 25
# General-Purpose Input/Output (GPIO)

## 25.1 Chip-specific GPIO information

### 25.1.1 GPIO signals

See Pin properties for GPIO signal properties.

## 25.2 Introduction

The general-purpose input and output (GPIO) module communicates to the processor core via a zero wait state interface (IOPORT) for maximum pin performance.The general-purpose input and output (GPIO) module is also accessible via the peripheral bus. The GPIO registers support 8-bit, 16-bit or 32-bit accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function. The GPIO input data register displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt module for that pin is enabled.

Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.

### 25.2.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers

- Port Data Direction register
- Zero wait state access to GPIO registers through IOPORT

**NOTE**

The GPIO module is clocked by system clock.

## 25.2.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

**Table 25-1. Modes of operation**

| Modes of operation | Description |
|---|---|
| Run | The GPIO module operates normally. |
| Wait | The GPIO module operates normally. |
| Stop | The GPIO module is disabled. |
| Debug | The GPIO module operates normally. |

## 25.2.3 GPIO signal descriptions

**Table 25-2. GPIO signal descriptions**

| GPIO signal descriptions | Description | I/O |
|---|---|---|
| PORTA31–PORTA0 | General-purpose input/output | I/O |
| PORTB31–PORTB0 | General-purpose input/output | I/O |
| PORTC31–PORTC0 | General-purpose input/output | I/O |
| PORTD31–PORTD0 | General-purpose input/output | I/O |
| PORTE31–PORTE0 | General-purpose input/output | I/O |

**NOTE**

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

### 25.2.3.1 Detailed signal description

**Table 25-3. GPIO interface-detailed signal descriptions**

| Signal | I/O | Description |
|---|---|---|
| PORTA31–PORTA0 | I/O | General-purpose input/output |

*Table continues on the next page...*

**Table 25-3. GPIO interface-detailed signal descriptions (continued)**

| Signal | I/O | Description | |
|---|---|---|---|
| PORTB31–PORTB0<br>PORTC31–PORTC0<br>PORTD31–PORTD0<br>PORTE31–PORTE0 | | State meaning | Asserted: The pin is logic 1.<br>Deasserted: The pin is logic 0. |
| | | Timing | Assertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.<br>Deassertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock. |

**NOTE**

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

## 25.3 Memory map and register definition

The registers for each GPIO port occupy 64-byte of the memory map. Any read or write access to the GPIO slot outside this space results in a bus error.

**NOTE**

For simplicity, each GPIO port's registers appear with the same width of 32 bits, corresponding to 32 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is chip-specific. Refer to the chip-specific GPIO information to see the exact control bits for each port.

### 25.3.1 GPIO register descriptions

### 25.3.1.1 GPIO Memory map

GPIOA base address: 4802_0000h

GPIOB base address: 4802_0040h

GPIOC base address: 4802_0080h

GPIOD base address: 4802_00C0h

GPIOE base address: 4100_F000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Port Data Output Register (PDOR) | 32 | RW | 0000_0000h |
| 4h | Port Set Output Register (PSOR) | 32 | WORZ | 0000_0000h |
| 8h | Port Clear Output Register (PCOR) | 32 | WORZ | 0000_0000h |
| Ch | Port Toggle Output Register (PTOR) | 32 | WORZ | 0000_0000h |
| 10h | Port Data Input Register (PDIR) | 32 | RO | 0000_0000h |
| 14h | Port Data Direction Register (PDDR) | 32 | RW | 0000_0000h |

### 25.3.1.2 Port Data Output Register (PDOR)

#### 25.3.1.2.1 Offset

| Register | Offset |
|---|---|
| PDOR | 0h |

#### 25.3.1.2.2 Function

This register configures the logic levels that are driven on each general-purpose output pin.

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

#### 25.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDO | Port Data Output<br>Register bits for unbonded pins return an undefined value when read.<br>0b - Logic level 0 is driven on pin, provided pin is configured for general-purpose output.<br>1b - Logic level 1 is driven on pin, provided pin is configured for general-purpose output. |

### 25.3.1.3 Port Set Output Register (PSOR)

#### 25.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PSOR | 4h |

#### 25.3.1.3.2 Function

This register configures whether to set the fields of the PDOR.

### 25.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTSO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTSO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 25.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTSO | Port Set Output<br>Writing to this register updates the contents of the corresponding bit in the PDOR as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is set to logic 1. |

## 25.3.1.4 Port Clear Output Register (PCOR)

### 25.3.1.4.1 Offset

| Register | Offset |
|---|---|
| PCOR | 8h |

### 25.3.1.4.2 Function

This register configures whether to clear the fields of PDOR.

#### 25.3.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTCO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTCO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTCO | Port Clear Output<br>Writing to this register updates the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is cleared to logic 0. |

### 25.3.1.5 Port Toggle Output Register (PTOR)

#### 25.3.1.5.1 Offset

| Register | Offset |
|---|---|
| PTOR | Ch |

#### 25.3.1.5.2 Function

This register toggles the logic levels that are driven on each general-purpose output pin.

### 25.3.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 25.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTTO | Port Toggle Output<br>Writing to this register updates the contents of the corresponding bit in the PDOR as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is set to the inverse of its existing logic state. |

## 25.3.1.6 Port Data Input Register (PDIR)

### 25.3.1.6.1 Offset

| Register | Offset |
|---|---|
| PDIR | 10h |

### 25.3.1.6.2 Function

This register captures the logic levels that are driven into each general-purpose input pin.

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

#### 25.3.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.3.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDI | Port Data Input<br>Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update.<br>0b - Pin logic level is logic 0, or is not configured for use by digital function.<br>1b - Pin logic level is logic 1. |

### 25.3.1.7 Port Data Direction Register (PDDR)

#### 25.3.1.7.1 Offset

| Register | Offset |
|---|---|
| PDDR | 14h |

#### 25.3.1.7.2 Function

The PDDR configures the individual port pins for input or output.

#### 25.3.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.3.1.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDD | Port Data Direction<br>Configures individual port pins for input or output.<br>0b - Pin is configured as general-purpose input, for the GPIO function.<br>1b - Pin is configured as general-purpose output, for the GPIO function. |

## 25.4 FGPIO memory map and register definition

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000.

Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.

Any read or write access to the FGPIO memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states, except error accesses which complete with one wait state.

**NOTE**

For simplicity, each FGPIO port's registers appear with the same width of 32 bits, corresponding to 32 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is chip-specific. Refer to the Chip Configuration chapter to see the exact control bits for the non-identical port instance.

## 25.4.1 FGPIOE register descriptions

### 25.4.1.1 FGPIOE Memory map

FGPIOE base address: F800_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Port Data Output Register (PDOR) | 32 | RW | 0000_0000h |
| 4h | Port Set Output Register (PSOR) | 32 | WORZ | 0000_0000h |
| 8h | Port Clear Output Register (PCOR) | 32 | WORZ | 0000_0000h |
| Ch | Port Toggle Output Register (PTOR) | 32 | WORZ | 0000_0000h |
| 10h | Port Data Input Register (PDIR) | 32 | RO | 0000_0000h |
| 14h | Port Data Direction Register (PDDR) | 32 | RW | 0000_0000h |

### 25.4.1.2 Port Data Output Register (PDOR)

#### 25.4.1.2.1 Offset

| Register | Offset |
|---|---|
| PDOR | 0h |

#### 25.4.1.2.2 Function

This register configures the logic levels that are driven on each general-purpose output pin.

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

#### 25.4.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDO | Port Data Output<br>Register bits for unbonded pins return an undefined value when read.<br>0b - Logic level 0 is driven on pin, provided pin is configured for general-purpose output.<br>1b - Logic level 1 is driven on pin, provided pin is configured for general-purpose output. |

### 25.4.1.3 Port Set Output Register (PSOR)

#### 25.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PSOR | 4h |

#### 25.4.1.3.2 Function

This register configures whether to set the fields of the PDOR.

#### 25.4.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTSO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTSO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTSO | Port Set Output<br>Writing to this register updates the contents of the corresponding bit in the PDOR as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is set to logic 1. |

### 25.4.1.4 Port Clear Output Register (PCOR)

#### 25.4.1.4.1 Offset

| Register | Offset |
|---|---|
| PCOR | 8h |

#### 25.4.1.4.2 Function

This register configures whether to clear the fields of PDOR.

#### 25.4.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTCO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTCO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTCO | Port Clear Output<br>Writing to this register updates the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is cleared to logic 0. |

### 25.4.1.5 Port Toggle Output Register (PTOR)

#### 25.4.1.5.1 Offset

| Register | Offset |
|---|---|
| PTOR | Ch |

#### 25.4.1.5.2 Function

This register toggles the logic levels that are driven on each general-purpose output pin.

#### 25.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PTTO | Port Toggle Output<br>Writing to this register updates the contents of the corresponding bit in the PDOR as follows:<br>0b - Corresponding bit in PDORn does not change.<br>1b - Corresponding bit in PDORn is set to the inverse of its existing logic state. |

### 25.4.1.6 Port Data Input Register (PDIR)

#### 25.4.1.6.1 Offset

| Register | Offset |
|---|---|
| PDIR | 10h |

#### 25.4.1.6.2 Function

This register captures the logic levels that are driven into each general-purpose input pin.

**NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

### 25.4.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 25.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDI | Port Data Input<br>Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update.<br>0b - Pin logic level is logic 0, or is not configured for use by digital function.<br>1b - Pin logic level is logic 1. |

## 25.4.1.7 Port Data Direction Register (PDDR)

### 25.4.1.7.1 Offset

| Register | Offset |
|---|---|
| PDDR | 14h |

### 25.4.1.7.2 Function

The PDDR configures the individual port pins for input or output.

#### 25.4.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 25.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PDD | Port Data Direction<br>Configures individual port pins for input or output.<br>0b - Pin is configured as general-purpose input, for the GPIO function.<br>1b - Pin is configured as general-purpose output, for the GPIO function. |

## 25.5 Functional description

### 25.5.1 General-purpose input

The logic state of each pin is available via the Port Data Input registers, provided the pin is configured for a digital function and the corresponding Port Control and Interrupt module is enabled.

The input pin synchronizers are shared with the Port Control and Interrupt module, so that if the corresponding Port Control and Interrupt module is disabled, then synchronizers are also disabled. This reduces power consumption when a port is not required for general-purpose input functionality.

### 25.5.2 General-purpose output

The logic state of each pin can be controlled via the port data output registers and port data direction registers, provided the pin is configured for the GPIO function. The following table depicts the conditions for a pin to be configured as input/output.

| If | Then |
|---|---|
| A pin is configured for the GPIO function and the corresponding port data direction register bit is clear. | The pin is configured as an input. |
| A pin is configured for the GPIO function and the corresponding port data direction register bit is set. | The pin is configured as an output and and the logic state of the pin is equal to the corresponding port data output register. |

To facilitate efficient bit manipulation on the general-purpose outputs, pin data set, pin data clear, and pin data toggle registers exist to allow one or more outputs within one port to be set, cleared, or toggled from a single register write.

The corresponding Port Control and Interrupt module does not need to be enabled to update the state of the port data direction registers and port data output registers including the set/clear/toggle registers.

### 25.5.3 IOPORT

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. If another bus master attempts to access the GPIO registers on the same cycle as an IOPORT access, then the DMA access will stall until any IOPORT accesses have completed.

# Chapter 26
# Reset and Boot

## 26.1 Introduction

The following reset sources exist in this MCU:

**Table 26-1. Reset sources**

| Reset Type | Description |
|---|---|
| Voltage resets | • System Power-On Reset (POR)<br>• System High-Voltage Detect (HVD)<br>• System Low-Voltage Detect (LVD)<br>• Battery Power-On Reset (VBATPOR) |
| External resets | • External pin reset (RESET_b)<br>• Low Leakage Wake-Up (LLWU) |
| Internal resets | • Watchdog (WDOG)<br>• Loss-of-Clock (LOC)<br>• Software (SW)<br>• Stop mode ACKnowledge ERRor (SACKERR)<br>• Lockup (LOCKUP)<br>• MDM-AP system reset request |
| Debug reset | • JTAG IR commands<br>• Resetting the debug sub-system |

Each system reset source has a corresponding bit in the System Reset Status (MSMC_SRS) register. This information can be used to take appropriate action when coming out of a reset. See the Multi System Mode Controller (MSMC) section for detailed information.

The microcontroller has 4 CPUs: Core0 (CM4F), Core1 (CM0+), Core2 (RI5CY), and Core3 (ZERO_RISCY).

- Resets initiated externally from the Cores (like External Global System resets) force a reset to all Cores.
- On exit from any reset, the boot core defined by the FTFE_FOPT[BOOT_CORE], FTFE_FOPT[CM4_EN], FTFE_FOPT[RI5CY_DIS], FTFE_FOPT[CM0+_EN] and FTFE_FOPT[ZERO_RISCY_DIS] is enabled and starts to execute. All other cores

are hold in reset until they are enabled by software. By default, the Cortex M4 is the boot core.

The microcontroller can exit and reset in functional mode where the Core is executing code (default) or the Core is in a debug halted state. There are several boot options that can be configured. See the Boot section for more information.

## 26.2 Reset

Asserting a reset to the device, a core, or a peripheral provides a way to start processing from a known set of initial conditions. On de-assertion of a system level reset source, the on-chip regulator is in full regulation and system clock generation is from an internal source. When the device exits reset, the boot core performs the following actions:

- ARM cores: Reads the initial Stack Pointer (SP) from vector-table at offset 0x0000_0000
- ARM cores: Reads the initial Program Counter (PC) from vector-table at offset 0x0000_0004
- ARM cores: Sets the Link Register (LR) to 0xFFFF_FFFF

After a system level reset, the on-chip peripherals are disabled, the non-analog I/O pins return to their default disabled configuration, and the analog I/O pins return to their default analog function configuration.

On the assertion of a system level reset, the debug pins return to their default enabled configuration as:

- JTAG_TCK/SWD_CLK with pull-down (PD) enabled
- JTAG_TMS with pull-up (PU) enabled
- JTAG_TDI/SWD_DIO with pull-up (PU) enabled
- JTAG_TDO with pull-up (PU) and pull-down (PD) disbaled

Some modules that cause resets can be configured to cause interrupts instead. Consult the individual peripheral chapters for more information.

### 26.2.1 Voltage monitor reset sources

To ensure predictable operation of the device, the supply voltages must remain in their specified ranges during execution. Because this is not possible during power-up and power-down, the device implements multiple voltage monitors to detect when the supply is below an acceptable level. When this occurs, the voltage monitor asserts a reset to halt execution and ensure that unexpected events do not occur.

#### 26.2.1.1 System Power-On Reset (POR)

The POR voltage monitor is used to ensure that the voltage applied to the system is high enough for other analog modules such as the bias circuits and low voltage monitors to operate correctly. When voltage is initially applied to the device or when the supply voltage drops below the Power-On Reset re-arm voltage ($V_{POR}$), the POR monitor asserts POR to the device.

The POR will reset everything in the device that is not in the battery voltage ($V_{BAT}$) domain. The POR field in the System Reset Status register (MSMC_SRS[POR]) is set following a POR.

#### 26.2.1.2 High-voltage detect (HVD)

The chip includes a system to guard against high-voltage conditions. See:

- High-voltage detect (HVD) system
- HVD reset operation
- HVD interrupt operation

#### 26.2.1.3 Low-Voltage Detect (LVD)

The LVD voltage monitor, which including VDDIO1 and CORE_VDD voltage monitor, is used to ensure that the core voltage is high enough for the digital logic and memories to operate correctly. When voltage is initially applied to the device, the POR voltage monitor will hold the LVD voltage monitor in reset until the voltage is high enough. The LVD monitor will continue to hold the device in reset until the core voltage has risen above the default LVD Low threshold ($V_{LVDL}$).

The VDDIO1 LVD monitor can be configured to generate a reset when the voltage drops below the VLVDL threshold. The voltage threshold at which the reset will assert is user programmable. This ensures that the device will halt operation and remain in reset while the core voltage is not within the desired range. The LVD Low threshold ($V_{LVDL}$) can be configured by programming the LVDV field of the Low Voltage Detect Status and Control 1 register (SPM_LVDSC1).

The CORE_VDD LVD monitor is enabled in the RUN and WAIT modes. The CORE_VDD LVD monitor can be configured to be enabled or disabled in STOP, VLPR, VLPW, VLPS, LLS, and VLLS modes. Additionally, the CORE_VDD LVD monitor can be configured not to generate resets by writing a 0 to the SPM_LVDSC1[COREVDD_LVDRE] field.

The LVD will reset everything in the device that is not in the battery voltage ($V_{BAT}$) domain except the Low-Power Timers (LPTMR), the contents of the System Register File (SYSREG), portions of the System Power Manager (SPM), and the debug sub-systems. The LVD field in the System Reset Status register (SRS[LVD] of MSMC) is set following a LVD reset.

#### 26.2.1.4 Battery Power-On Reset (VBATPOR)

The VBATPOR voltage monitor is used to ensure that the voltage applied to the battery domain is high enough for the $V_{BAT}$ modules to operate correctly. When voltage is initially applied to the $V_{BAT}$ domain or when the voltage drops below an acceptable level, the VBATPOR monitor asserts a POR to the $V_{BAT}$ domain.

### 26.2.2 External system reset sources

External system resets are provided so that the device can be reset or awakened from very low power states. This allows the user to start the device at the correct time from a known state. The external system resets available in this device are described in the following sections.

#### 26.2.2.1 External pin reset (RESET_b)

The RESET_b is a dedicated pin on this device. This pin is open drain and has an internal pull-up. Asserting RESET_b wakes the device from any mode.

##### 26.2.2.1.1 RESET_b filter functionality

The RESET_b pin filter supports filtering from both the 1kHz LPO clock and the bus clock. The LPOFEN, FILTEN and FILTCFG fields of the Reset Pin Control (MSMC_RPC) register controls this functionality. For detailed information on these features, please refer to the Multi System Mode Controller (MSMC) chapter. The filters are asynchronously reset by the system POR. The reset value for each filter assumes the RESET_b pin is negated.

For all stop modes where LPO clock is still active (STOP, VLPS, LLS, and VLLS), the only filtering option is the LPO-based digital filter. The filtering logic either switches to bypass operation or has continued filtering operation depending on the filtering mode selected. When entering VLLS0, the RESET_b pin filter is disabled and bypassed.

The LPO filter has a fixed filter value of 3. Due to a synchronizer on the input data, there is also an associated latency of 2 cycles. As a result, 5 cycles are required to complete a transition from low-to-high or high-to-low.

#### 26.2.2.1.2 RESET_b output functionality

For all reset sources except a VLLS Wakeup that does not occur via the RESET_b pin, the RESET_b pin is driven low by the MCU for at least 128 bus clock cycles and until flash initialization has completed.

After flash initialization has completed, the RESET_b pin is released, and the internal Chip Reset negates after the RESET_b pin is pulled high. Keeping the RESET_b pin asserted externally delays the negation of the internal Chip Reset.

### 26.2.2.2 Low Leakage Wake-Up (LLWU)

The LLWU module provides the means for a number of external pins and a number of internal peripherals to wake the device from low leakage power modes, LLS and VLLS. It also allows for asynchronous DMA operation to wake-up from LLS when transfers to or from certain peripherals complete.

The LLWU can enable the external input pin and filter detection in all power modes (not only LLS/VLLS). In LLS_mode, only the RESET_b pin can generate a system reset through the LLWU. In VLLS modes, all enabled inputs to the LLWU can generate a system reset. After a system reset, the LLWU retains the flags indicating the input source of the last wakeup until the user clears them. The WAKEUP field in the System Reset Status register (MSMC_SRS[WAKEUP]) is set following a LLWU reset.

## 26.2.3 Internal system reset sources

Internal system resets are provided so that the device can be reset when certain erroneous conditions are detected. This allows the device to reset a portion of the device or the entire device to recover from the erroneous conditions. The internal system resets available in this device are described in the following sections.

#### 26.2.3.1 Watchdog (WDOG) timer

The WDOG monitors the operation of the system by receiving periodic communication from the software. This communication is generally known as servicing or refreshing the Watchdog. If this periodic servicing does not occur, then the Watchdog issues a system reset.

The WDOG will reset everything in the device that is not in the battery voltage ($V_{BAT}$) domain except the Low-Power Timers (LPTMR), the contents of the System Register File (SYSREG), System Power Manager (SPM), and the debug sub-systems. The WDOG field in the System Reset Status register (MSMC_SRS[WDOG]) is set following a WDOG reset.

In multi-core devices, the WDOG can be configured to generate an interrupt to the boot core in the device rather than assert an interrupt. This allows the boot core to run an Interrupt Service Routine (ISR) and make a determination on how to handle the WDOG interrupt. The boot core could decide to reset the core domain that generated the WDOG interrupt through MUA/B_CCR[RSTH] or reset the entire device.

#### 26.2.3.2 Loss-of-Clock (LOC)

The System Clock Generator (SCG) module contains a clock monitor with reset and interrupt request capability for external clocks. The LOC indicates a reset has been caused by a loss of external clock. The SCG clock monitor must be enabled for a loss of clock to be detected. For details on enabling the clock monitor, please refer to the SCG chapter.

**NOTE**

To prevent unexpected reset events, all clock monitors must be disabled before entering any low-power modes, including VLPR and VLPW.

#### 26.2.3.3 Software (SW)

During execution, if the software detects erroneous operation, the software can initiate a reset to restart the device from a known state. The software can write the SYSRESETREQ field in the Nested Vectored Interrupt Controller - Application Interrupt and Reset Control Register (NVIC_AIRCR) to force a SW reset to the core. (See ARM's NVIC documentation for the full description of the register fields, especially the VECTKEY field requirements.)

The SW SYSRESETREQ request will reset all of the major components of the core except for the debug sub-systems. The SW field in the System Reset Status register (MSMC_SRS[SW]) is set following a SW reset.

In multi-core devices, the SW can be configured to generate an interrupt to the master core in the device rather than assert an interrupt. This allows the master core to run an Interrupt Service Routine (ISR) and make a determination on how to handle the SW interrupt. The master core could decide to reset the core domain that generated the SW interrupt or reset the entire device.

The SW system reset will reset everything in the device that is not in the battery voltage ($V_{BAT}$) domain except the Low-Power Timers (LPTMR), the contents of the System Register File (SYSREG), portions of the System Power Manager (SPM), and the debug sub-systems. The PIN and SW fields in the System Reset Status (MSMC_SRS) register are set following a SW system reset.

#### 26.2.3.4 Stop mode ACKnowledge ERRor (SACKERR)

The SACKERR reset is generated if a core attempts to enter STOP mode or Compute Operation and one or more of the modules in the core's domain fails to acknowledge Stop mode within 1025 cycles of the 1 kHz LPO clock.

A module might not acknowledge the entry to STOP mode if an error condition occurs. The error can be caused by a failure of an external clock input to a module.

The SACKERR condition will reset everything in the device that is not in the battery voltage (VBAT) domain except the Low-Power Timers (LPTMR), the contents of the System Register File (SYSREG), portions of the System Power Manager (SPM), and the debug sub-systems. The SACKERR field in the System Reset Status register (MSMC_SRS[SACKERR]) is set following a SACKERR reset.

#### 26.2.3.5 Lockup (LOCKUP)

The assertion of the LOCKUP condition gives an immediate indication of seriously errant kernel software. This is the result of the core being locked, because of an unrecoverable exception, following the activation of the processor's built-in system state protection hardware.

In multi-core devices, the LOCKUP can be configured to generate an interrupt to the master core in the device rather than assert an interrupt. This allows the master core to run an Interrupt Service Routine (ISR) and make a determination on how to handle the LOCKUP interrupt. The master core could decide to reset the core domain that generated the LOCKUP interrupt or reset the entire device.

#### 26.2.3.6 MDM-AP system reset request

To initiate a system reset, set the System Reset Request field in the MDM-AP control register. This is the primary method for resets via the debug interfaces. The system reset is held until this field is cleared.

To hold the core in reset as the rest of the chip comes out of system reset, set the Core Hold Reset field in the MDM-AP control register.

### 26.2.4 Debug resets

The following sections detail the debug resets available on the device.

#### 26.2.4.1 JTAG Instruction Register (IR) commands

Some JTAG IR commands will assert reset to the system. This is required to ensure proper behavior in the JTAG TAP modes. This reset acts the same as a pin reset to the system. For more information on the JTAG IR commands, please refer to the JTAG chapter.

#### 26.2.4.2 Resetting the debug sub-system

Use the CDBGRSTREQ bit within the SWJ-DP CTRL/STAT register to reset the debug modules. However, as explained below, using the CDBGRSTREQ bit does not reset all debug-related registers.

CDBGRSTREQ resets the debug-related registers within the following modules:

- SWJ-DP
- AHB-AP
- ETM
- TPIU
- MDM-AP (MDM control and status registers)

CDBGRSTREQ does not reset the debug-related registers within the following modules:

- CM0+ core
- CM4F core
- BPU
- FPB
- DWT
- ITM
- NVIC
- Crossbar bus switch[1]
- AHB-AP[1]
- Private peripheral bus[1]

## 26.3 Boot

This section describes the boot sources, boot options, and boot sequence.

The various boot options apply to the Boot Core, which is the Core on exit from reset. By default, Core0 is the Boot Core and Core1, Core2, and Core 3 are disabled. If the user sets the appropriate bits in the boot options field, Core1 or/and Core3 will be enabled as the boot core, and Core0 or/and Core2 will be disabled on exit from reset.

Some configuration information such as clock trim values stored in factory programmed flash locations is autoloaded.

### 26.3.1 Boot sources

Based on the source selections in the Flash Option (FOPT) register, the device will boot from the internal Flash or the internal Boot ROM. If the device boots from the Boot ROM, the device executes in boot loader mode.

### 26.3.2 Flash Option (FOPT) customization

The Flash Option (FOPT) register in the Flash Memory module (FTFE_FOPT) allows the user to customize the operation of the MCU at boot time. The register contains read-only bits that are loaded from the NVM's flash configuration field. The default setting for all values in the FTFE_FOPT register is logic 1 since it is copied from the option byte

1. CDBGRSTREQ does not affect AHB resources so that debug resources on the private peripheral bus are available during System Reset.

residing in flash, which has all bits as logic 1 in the flash erased state. To configure for alternate settings, program the appropriate bits in the NVM option byte. The new settings will take effect on subsequent POR and any system reset. For more details on programming the option byte, see the flash memory chapter.

The MCU uses the FTFE_FOPT to configure the device at reset as shown in the following table.

**NOTE**

An FTFE_FOPT value of 0x0000_0000 is invalid and will be ignored. The FOPT register is written to 0xFFFF_FFFF if the contents of the Flash non-volatile option are 0x0000_0000.

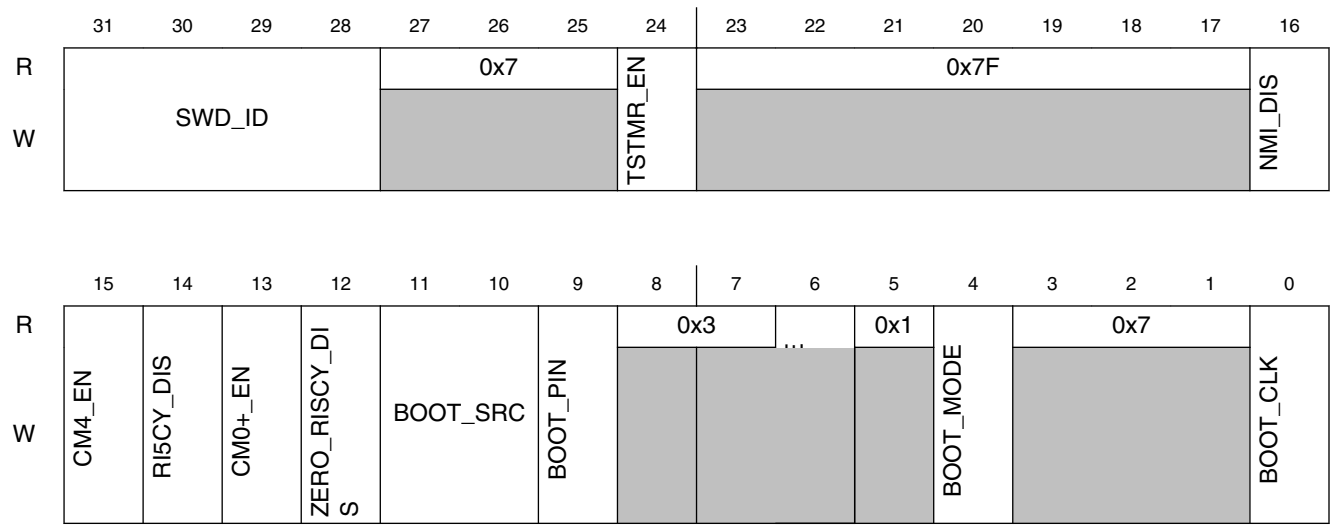



**Table 26-4. Flash Option Register Bit Definitions**

| Bit Num | Field | Definition |
|---|---|---|
| 31-28 | SWD_ID | **Serial Wire Debug Identification** - The SWD_ID sets the Instance ID of SWD modules that support multidrop. The Instance ID is used to select a specific SWD instance in systems with a single SWD interface port, but multiple devices with SWD interfaces. |
| 27-25 | Reserved | This field is reserved. |
| 24 | TSTMR_EN | **Time Stamp Timer Enable** - The TSTMR_EN bit enables the Time Stamp Timer (TSTMR) when the the Slow-Speed IRC is enabled.<br>• 0 - The TSTMR is disabled and cannot be enabled through any hardware or software operations.<br>• 1 - The TSTMR is enabled and begins counting whenever the Slow-Speed IRC is enabled. |
| 23-17 | Reserved | This field is reserved. |
| 16 | NMI_DIS | **Non-Maskable Interrupt (NMI) Disable** - The NMI_DIS bit disables the NMI signal. |

*Table continues on the next page...*

## Table 26-4. Flash Option Register Bit Definitions (continued)

| Bit Num | Field | Definition |
|---|---|---|
| | | • 0 - NMI pin and interrupts are enabled.<br>• 1 - NMI interrupts are always blocked. The associated pin continues to default to NMI pin controls with internal pullup enabled. |
| 15 | CM4_EN | **ARM CM4F core Enable** - The CM4_EN enables CM4F core.<br>• 0 - CM4F core is disabled.<br>• 1 - CM4F core is enabled. |
| 14 | RI5CY_DIS | **RI5CY core Disable** - The RI5CY_DIS bit disables RI5CY core.<br>• 0 - RI5CY core is enabled.<br>• 1 - RI5CY core is disabled. |
| 13 | CM0+_EN | **ARM CM0+ core Enable** - The CM0+_EN enables CM4F core.<br>• 0 - CM0+ core is disabled.<br>• 1 - CM0+ core is enabled. |
| 12 | ZERO_RISCY_DIS | **ZERO_RISCY core Disable** - The ZERO_RISCY_DIS bit disables ZERO_RISCY core.<br>• 0 - ZERO_RISCY core is enabled.<br>• 1 - ZERO_RISCY core is disabled. |
| 11-10 | BOOT_SRC | **Boot Source** - The BOOT_SRC field selects the boot source if the Boot Pin Configuration bit (BOOT_PIN) is set to 1.<br>• 00 - Boot from the internal flash memory.<br>• 01 - Reserved.<br>• 10 - Boot from the Boot ROM, configure communications peripheral(s), and enter boot loader mode. See Boot sequence for more information<br>• 11 - Boot from the Boot ROM, and enter boot loader mode. See Boot sequence for more information. |
| 9 | BOOT_PIN | **Boot Pin Configuration** - The BOOT_PIN bit controls the behavior of the boot pin selection.<br>• 0 - Force Boot from ROM with update if BOOTCFG0 asserted, where BOOTCFG0 is the boot configuration function which is muxed with NMI pin. The RESET_b pin must be enabled when this option is selected.<br>• 1 - Boot source configured by Boot Source (BOOT_SRC) field. |
| 8-7 | Reserved | This field is reserved. |
| 6 | BOOT_CORE | **Boot Core** - The BOOT_CORE bit selects which core will start executing code at the completion of initialization. Additionally, the Non-Maskable Interrupt (NMI) is routed to this core if the NMI_DIS bit is set to 0.<br>• 0 - CPU1 (Cortex-M0+) or/and CPU3 (ZERO-RISCY), which depends on [CM0+_EN] and [ZERO_RISCY_DIS] settings, will start executing code on completion of initialization.<br>• 1 - CPU0 (Cortex-M4) or/and CPU2 (RI5CY), which depends on [CM4_EN] and [RI5CY_DIS] settings,will start executing code on completion of initialization. |
| 5 | Reserved | This field is reserved for future expansion. |
| 4 | BOOT_MODE | **Boot Mode** - The BOOT_MODE bit selects the operational mode that will be entered at the completion of initialization.<br>• 0 - The device will exit initialization in VLPR mode.<br>• 1 - The device will exit initialization in RUN mode. |
| 3-1 | Reserved | This field is reserved. |
| 0 | BOOT_CLK | **Boot Clock** - The BOOT_CLK field controls the setting of the Core Clock Divide Ratio (DIVCORE) field. Depending on the value of the BOOT_MODE bit, this field will be set in either the SCG_RCCR register or the SCG_VCCR register.<br>BOOT_MODE = 1 (Exit initialization in RUN mode.) |

**Table 26-4. Flash Option Register Bit Definitions**

| Bit Num | Field | Definition |
|---|---|---|
| | | • 1 - SCG_RCCR[DIVCORE] set to 0000 - Divide-by-1<br>• 0 - SCG_RCCR[DIVCORE] set to 0001 - Divide-by-2<br><br>BOOT_MODE = 0 (Exit initialization in VLPR mode.)<br>• 1 - SCG_VCCR[DIVCORE] set to 0011 - Divide-by-4<br>• 0 - SCG_VCCR[DIVCORE] set to 0111 - Divide-by-8 |

## 26.3.3 Boot sequence

At power up, the on-chip POR voltage monitor holds the device in a POR state until the input supply exceeds the POR threshold, VPOR. The system continues to be held in this static state until the internal supplies have reached a safe operating voltage as determined by the LVD. The Multi System Mode Controller (MSMC) logic then completes a sequence of operations to exit reset.

1. The MSMC continues to assert reset to the internal logic, the RESET_b pin is driven low, and the SCG is enabled in its default clocking mode, (FIRC clock source selected).
2. The SCG enables the required clocks (system clock, flash clock, and any bus clocks).
3. The MSMC de-asserts reset to the Flash Memory Controller (FMC) while still asserting reset to the rest of the internal logic and driving the RESET_b pin low.
4. The FMC then configures the flash and reads the flash option byte. The data is stored in the Flash Memory Module (FTFE) in the Flash Option (FTFE_FOPT) register.
5. The flash option configurations from the FTFE_FOPT register are applied to the system to adjust the clock frequency, the clock divider, and other associated functions.
6. The FMC then reads the core option byte. The data is stored in the FTFE in the Core Option (FTFE_FOPT) register.
7. The core option configurations from the FTFE_FOPT register are applied to the system to select which core will be used as the boot core and which one remains power gated.
8. When flash initialization completes, the MSMC stops driving the RESET_b pin low. If RESET_b continues to be asserted (an indication of a slow rise time on the RESET_b pin or an external device driving the pin low), the device continues to be held in reset. Once the RESET_b pin is detected high, the boot core clock is enabled and the device is released from reset.
9. When the device exits reset, the boot core sets up the stack, program counter (PC), and link register (LR). The processor reads the start SP (SP_main) from vector-table

offset 0. The core reads the start PC from vector-table offset 4. LR is set to 0xFFFF_FFFF. The next sequence of events depends on the FTFE_FOPT[BOOT_PIN] and FTFE_FOPT[BOOT_SRC].

- If FTFE_FOPT[BOOT_PIN] = 0:
  - If NMI=0, boot from ROM
  - If NMI=1, boot option base on FTFE_FOPT[BOOT_SRC]
- If FTFE_FOPT[BOOT_PIN] = 1, boot option base on FTFE_FOPT[BOOT_SRC]

FTFE_FOPT[BOOTSRC]:

- If FTFE_FOPT[BOOT_SRC] == 00, boot from flash
- If FTFE_FOPT[BOOT_SRC] == 11/10, boot from ROM
- If FTFE_FOPT[BOOT_SRC] == 01, reserved

**NOTE**

If the NMFI function is not required, either for an interrupt or wake up source, it is recommended that the NMI function be disabled by setting the FTFE_FOPT[NMI_DIS].

Subsequent system resets follow this same reset flow.

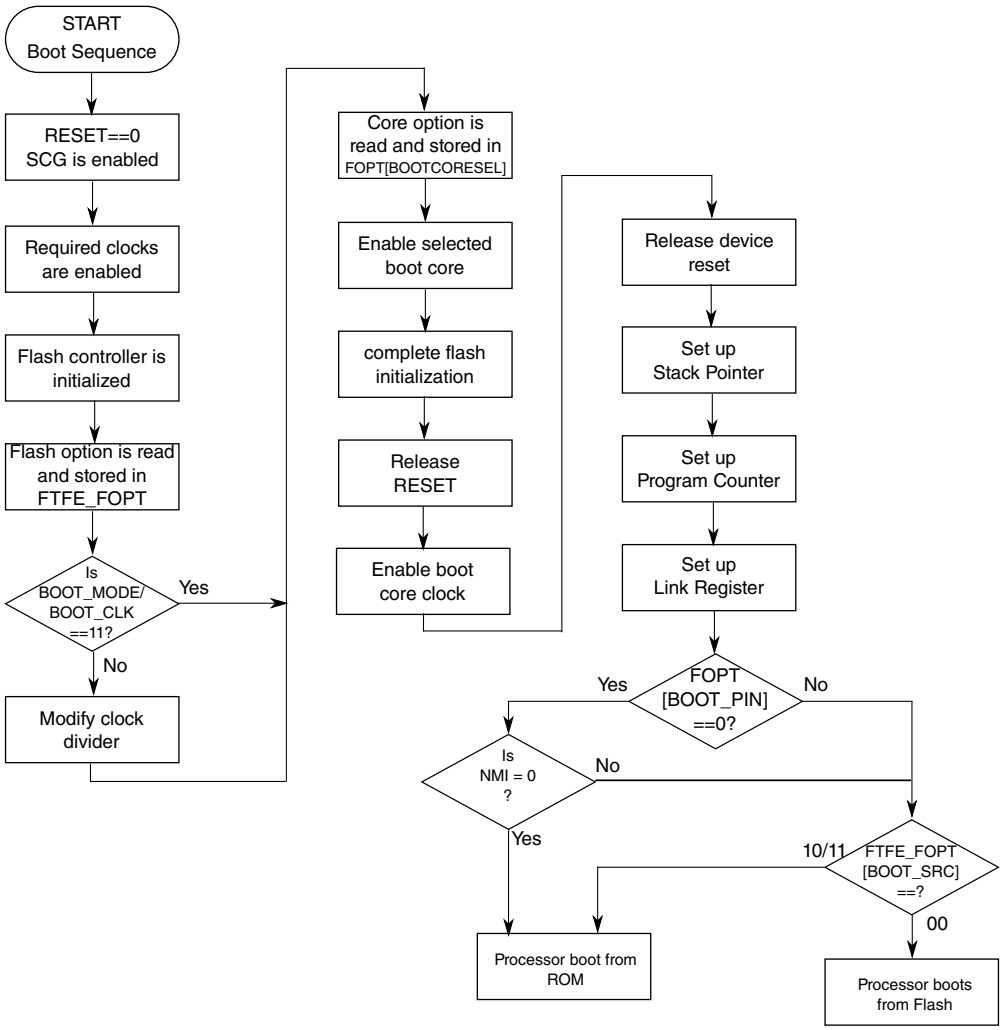



**Figure 26-1. Boot Sequence**

# Chapter 27
# Kinetis ROM Bootloader

## 27.1 Chip-specific information

This chapter is for ARM core only.

### 27.1.1 Also see the Kinetis Bootloader Reference Manual

This chapter describes only part of the information for your device's specific ROM Bootloader. For more detailed information and explanations of how the ROM Bootloader works, see the Kinetis Bootloader v2.0.0 Reference Manual, which can be found on https://www.nxp.com.

### 27.1.2 Kinetis bootloader peripheral pinmux

This device has various peripherals (LPUART, LPI2C, LPSPI , USB) supported by the Kinetis ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in The Kinetis Bootloader Configuration Area (BCA). If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default. The next table shows the pads used by the Kinetis ROM Bootloader.

**Table 27-1. Kinetis Bootloader Peripheral Pinmux**

| Peripheral | Instance | Alt Mode | Pins |
|---|---|---|---|
| LPUART | 0 | 3 | PTC7, LPUART0_RX |
| | | | PTC8, LPUART0_TX |
| I2C | 0 | 4 | PTC10, LPI2C0_SCL |
| | | | PTC9, LPI2C0_SDA |
| SPI | 0 | 2 | PTC12, LPSPI0_PCS0 |
| | | | PTC8, LPSPI0_SCK |

*Table continues on the next page...*

**Table 27-1. Kinetis Bootloader Peripheral Pinmux (continued)**

| Peripheral | Instance | Alt Mode | Pins |
|---|---|---|---|
| | | | PTC11, LPSPI0_SIN |
| | | | PTC9, LPSPI0_SOUT |
| USB | 0 | 0 | USB0_DP |
| | | | USB0_DM |

### 27.1.3 Bootloader Memory Access

While executing, the Kinetis Bootloader uses the following address ranges in ROM and RAM memory:

- 0x0880_0000 – 0x0880_BFFF (Boot ROM, 48 KB)
- 0x0800_0000 – 0x0800_FFFF (Cortex M4 ITCM SRAM, 64 KB)

## 27.2 Introduction

The Kinetis bootloader is the program residing in the on-chip read-only memory (ROM) of a Kinetis microcontroller device. There is hardware logic in place at boot time that either starts execution of an embedded image available on the internal flash memory, or starts the execution of the Kinetis Bootloader from on-chip ROM.

The Kinetis Bootloader's main task is to provision the internal flash memory with an embedded firmware image during manufacturing, or at any time during the life of the Kinetis device. The Kinetis Bootloader does the provisioning by acting as a slave device, and listening to various peripheral ports where a master can start communication.

For the Kinetis device, the Kinetis Bootloader can interface with USB, LPI2C, LPSPI, and LPUART peripherals in slave mode and respond to the commands sent by a master (or host) communicating on one of those ports. The host/master can be a firmware-download application running on a PC or an embedded host communicating with the Kinetis Bootloader. Regardless of the host/master (PC or embedded host), the Kinetis Bootloader always uses a command protocol to communicate with that host/master. Commands are provided to write to memory (internal flash or RAM), erase flash, and get/set bootloader options and property values. The host application can query the set of available commands.

On start-up, the bootloader reads optional configuration parameters from a fixed area on flash called the bootloader configuration area (BCA). These parameters can be modified by the write memory command or by downloaded flash image. BCA parameters include configuration data such as enabled peripherals, peripheral-specific settings, etc.

This chapter describes Kinetis Bootloader features, functionality, command structure and which peripherals are supported.

Features supported by the Kinetis Bootloader in Kinetis ROM:

- Supports USB, LPI2C, LPSPI, and LPUART peripheral interfaces
- Automatic detection of the active peripheral
- Ability to disable any peripheral
- LPUART peripheral implements autobaud
- Common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Flash-resident configuration options
- Fully supports internal flash security, including ability to mass erase or unlock security via the backdoor key
- Protection of RAM used by the bootloader while it is running
- Provides command to read properties of the device, such as flash and RAM size
- Multiple options for executing the bootloader either at system start-up or under application control at runtime
- Supports internal flash
- Supports encrypted image download
- Supports application image authentication
- ROM boots from either M4 (Default) or M0+ by configuring to FOPT IFR (record Index 0x84)

**Table 27-2. Commands supported by the Kinetis Bootloader in ROM**

| Command | Description | When flash security is enabled, then this command is |
|---|---|---|
| Call | Runs user application code and returns control to bootloader | Not supported |
| Execute | Run user application code that never returns control to the bootloader | Not supported |
| FillMemory | Fill a range of bytes in flash with a word pattern | Not supported |
| FlashEraseAll | Erase the entire flash array | Not supported |
| FlashEraseRegion | Erase a range of sectors in flash | Not supported |
| WriteMemory | Write data to memory | Not supported |
| ReadMemory | Read data from memory | Not supported |
| FlashProgramOnce | Writes data provided in a command packet to a specified range of bytes in the program once field | Not supported |

*Table continues on the next page...*

**Table 27-2. Commands supported by the Kinetis Bootloader in ROM (continued)**

| Command | Description | When flash security is enabled, then this command is |
|---|---|---|
| FlashReadOnce | Returns the contents of the program once field by given index and byte count | Not supported |
| FlashReadResource | Returns the contents of the IFR field or Flash Version ID, by given offset, byte count and option | Not supported |
| ReliableUpdate | Copy backup application from address to main application region, or swap flash using an indicator address | Not supported |
| | | |
| FlashSecurityDisable | Attempt to unlock flash security using the backdoor key | Supported |
| GetProperty | Get the current value of a property | Supported |
| ReceiveSbFile | Receive an SB file (which is generated by the `elftosb` tool) | Supported if the SB file is encrypted. See the SB File Decryption Support section for more details. |
| Reset | Reset the chip | Supported |
| SetProperty | Attempt to modify a writable property | Supported |
| FlashEraseAllUnsecure | Erase the entire flash array, including protected sectors | Supported |

## 27.3 Functional Description

The following sub-sections describe the Kinetis Bootloader in ROM functionality.

### 27.3.1 The Kinetis Bootloader Configuration Area (BCA)

The Kinetis Bootloader reads data from the Bootloader Configuration Area (BCA) to configure various features of the bootloader. The BCA resides in flash memory at offset 0x3C0 when device boots from CM4, otherwise the BCA location is 0x010003C0 when device boots from CM0+, and provides all of the parameters needed to configure the Kinetis Bootloader operation. For uninitialized flash, the Kinetis Bootloader uses a predefined default configuration. A host application can use the Kinetis Bootloader to program the BCA for use during subsequent initializations of the bootloader.

**Table 27-3. Configuration Fields for the Kinetis Bootloader**

| Offset | Size (bytes) | Configuration Field | Description |
|---|---|---|---|
| 0x00 - 0x03 | 4 | tag | Magic number to verify bootloader configuration is valid. Must be set to 'kcfg'. |

*Table continues on the next page...*

## Table 27-3. Configuration Fields for the Kinetis Bootloader (continued)

| Offset | Size (bytes) | Configuration Field | Description |
|---|---|---|---|
| 0x04 - 0x07 | 4 | crcStartAddress | Start address for application image CRC check. To generate the CRC, refer to the CRC chapter. If the bits are all set then Kinetis bootloader by default will not perform any CRC check. |
| 0x08 - 0x0B | 4 | crcByteCount | Byte count for application image CRC check. If the bits are all set then Kinetis bootloader by default will not perform any CRC check. |
| 0x0C - 0x0F | 4 | crcExpectedValue | Expected CRC value for application CRC check. If the bits are all set then Kinetis bootloader by default will not perform any CRC check. |
| 0x10 | 1 | enabledPeripherals | Bitfield of peripherals to enable.<br>bit 0 LPUART<br>bit 1 LPI2C<br>bit 2 LPSPI<br>bit 4 USB<br>Kinetis bootloader will enable the peripheral if corresponding bit is set to 1. |
| 0x11 | 1 | i2cSlaveAddress | If not 0xFF, used as the 7-bit I2C slave address. If 0xFF, defaults to 0x10 for I2C slave address |
| 0x12 - 0x13 | 2 | peripheralDetectionTimeout | Timeout in milliseconds for active peripheral detection. If 0xFFFF, defaults to 5 seconds. |
| 0x14 - 0x15 | 2 | usbVid | Sets the USB Vendor ID reported by the device during enumeration. If 0xFFFF, it defaults to 0x15A2. |
| 0x16 - 0x17 | 2 | usbPid | Sets the USB Product ID reported by the device during enumeration. |
| 0x18 - 0x1B | 4 | usbStringsPointer | Sets the USB Strings reported by the device during enumeration. Default string values are described in the USB peripheral section. |
| 0x1C | 1 | Reserved | - |
| 0x1D | 1 | Reserved | - |
| 0x1F | 1 | Reserved | - |
| 0x20 | 4 | Reserved | - |
| 0x24 - 0x27 | 8 | Reserved | - |
| 0x29 - 0x2F | 7 | Reserved | - |
| 0x28 | 1 | Reserved | - |
| 0x30 | 4 | Reserved | - |
| 0x34 | 12 | Reserved | - |

**NOTE**

The flash sector containing the BCA should not be located in the execute-only region, because the Kinetis bootloader cannot read an execute-only region.

The first configuration field 'tag' is a tag value or magic number. The tag value must be set to 'kcfg' for the bootloader configuration data to be recognized as valid. If tag-field verification fails, then the Kinetis Bootloader assumes that the flash is not initialized and uses a predefined default configuration. The tag value is treated as a character string, so bytes 0-3 must be set as shown in the table.

**Table 27-4. tag Configuration Field**

| Offset | tag Byte Value |
|---|---|
| 0 | 'k' (0x6B) |
| 1 | 'c' (0x63) |
| 2 | 'f' (0x66) |
| 3 | 'g' (0x67) |

## 27.3.2 Start-up Process

Any of the following conditions will force the hardware to start the Kinetis Bootloader:

- FOPT [11] is set to 1. This forces the ROM to run out of reset.
- The BOOTCFG0 pin is asserted. The pin must be configured as BOOTCFG0 by setting the BOOTPIN_OPT bit of FOPT to 0.
- A user applications running on flash or RAM calls into the Kinetis Bootloader entry point address in ROM, to start Kinetis Bootloader execution.

The FOPT[BOOT_SRC] determines the boot source. The FOPT register is located in flash erasable IFR field at index 0x84 in the flash IFR memory array. For a complete list of options, see the Boot options section in the Reset and Boot chapter. If FOPT [11] is set to 1, then the device will boot to ROM out of reset. Flash memory defaults to all 1s when erased, so a blank chip will automatically boot to ROM.

The BOOTCFG0 pin is shared with the NMI pin, with NMI being the default usage. Regardless of whether the NMI pin is enabled or not, the NMI functionality is disabled if the ROM is executed out of reset, for as long as the ROM is running.

When the ROM is executed out of reset, vector fetches from the CPU are redirected to the ROM's vector table in ROM memory at offset 0x0880_0000. This ensures that any exceptions will be handled by the ROM.

After the Kinetis Bootloader has started, the following procedure starts bootloader operations:

1. The FOPT[BOOT_CORE] determines which core is currently booting from ROM.
2. The MSMC_MR[BOOTCFG] bit is set, so that the device will reboot back into the ROM if/when the device is reset.
3. Initializes the bootloader's .data and .bss sections.
4. Reads bootloader configuration data from flash at address 0x3C0 or 0x010003C0. The configuration data is only used if the tag field is set to the expected 'kcfg' value. If the tag is incorrect, then the configuration values are set to default, as if the data was all 0xFF bytes.
5. Enabled peripherals are initialized.
6. After coming out of reset, the ROM bootloader initializes all peripherals, looks to see if a valid application is in flash, then auto-detects any peripheral activities within a peripheral detect timeout period.

   ```
   #define BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT 5000   // 5s
   ```

   Before the BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT expires and even if a valid application exists in flash, the ROM bootloader still runs the automatic peripheral detection. After the BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT time period expires:
   - If no active peripherals are detected but there is a valid application in flash, then the ROM bootloader jumps to the application.
   - If no active peripherals are detected and there is no valid application in flash, then the ROM bootloader continues to auto-detect peripherals activities, until the hit time (BL_DEFAULT_POWERDOWN_TIMEOUT) expires; after that, the ROM bootloader puts the chip into vlls1 mode (enter_vlls1();).

   ```
   #define BL_DEFAULT_POWERDOWN_TIMEOUT 600000 //10mins)
   ```

**NOTE**

The flash sector containing the vector table should not be located in the execute-only region, because the Kinetis bootloader cannot read the PC and SP addresses in an execute-only region.

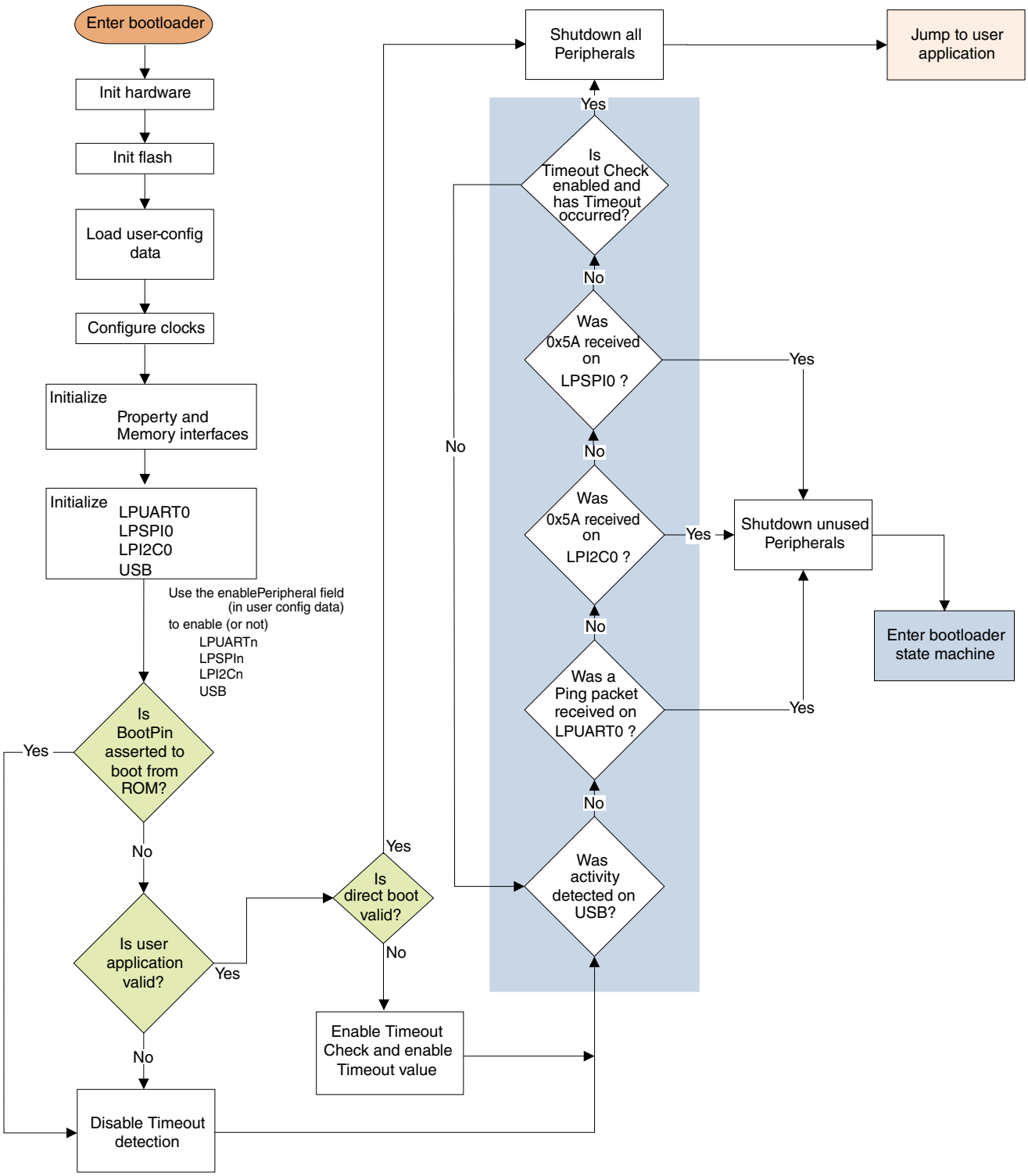



**Figure 27-1. Kinetis Bootloader Start-up Flowchart**

### 27.3.3 Bootloader Entry Point / API Tree

To run the Kinetis Bootloader, a user application simply calls the runBootloader function. To get the address of the entry point, the user application reads the word containing the pointer to the bootloader API tree at offset 0x1C of the bootloader's vector table. The vector table is placed at the base of the bootloader's address range, which for the ROM is 0x0880_0000; the API tree pointer is at address 0x0880_001C.

The bootloader API tree is a structure that contains pointers to other structures, which have the function and data addresses for the bootloader. The bootloader entry point is always the 1st word of the API tree.

```
typedef struct BootloaderTree
{
    void (*runBootloader)(void *arg);            //!< Function to start the bootloader
executing.
    standard_version_t version;                  //!< Bootloader version number.
    const char *copyright;                       //!< Copyright string.
    const bootloader_context_t *runtimeContext;  //!< Pointer to the bootloader's runtime
context.
    const flash_driver_interface_t *flashDriver; //!< Flash driver API.
    const aes_driver_interface_t *aesDriver;     //!< AES driver API.
    const kb_interface_t *kbApi;                 //!< Bootloader API.
} bootloader_tree_t;
```

The prototype of the entry point is:

```
void run_bootloader(void * arg);
```

The `arg` parameter is currently unused, and is intended for future expansion (for example, passing options to the bootloader). To ensure future compatibility, a value of NULL should be passed for `arg`.

***Example:*** code to get the entry pointer address from the ROM and start the bootloader.

**NOTE**

This entry must be called in supervisor (privileged) mode.

```
// Variables

uint32_t runBootloaderAddress;

void (*runBootloader)(void * arg);


// Read the function address from the ROM API tree.

runBootloaderAddress = **(uint32_t **)(0x0880001c);

runBootloader = (void (*)(void * arg))runBootloaderAddress;


// Start the bootloader.

runBootloader(NULL);
```

### 27.3.4 Bootloader Protocol

This section explains the general protocol for the packet transfers between the host and the Kinetis Bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase:

- If the data phase is **incoming** (from host to bootloader ), then the data phase is part of the **original command**.
- If the data phase is **outgoing** (from bootloader to host), then the data phase is part of the **response command**.

**NOTE**

In all protocols (described in the next subsections), the Ack sent in response to a Command or Data packet can arrive at any time *before, during, or after* the Command/Data packet has processed.

#### 27.3.4.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from host)
- Generic response command packet (to host)

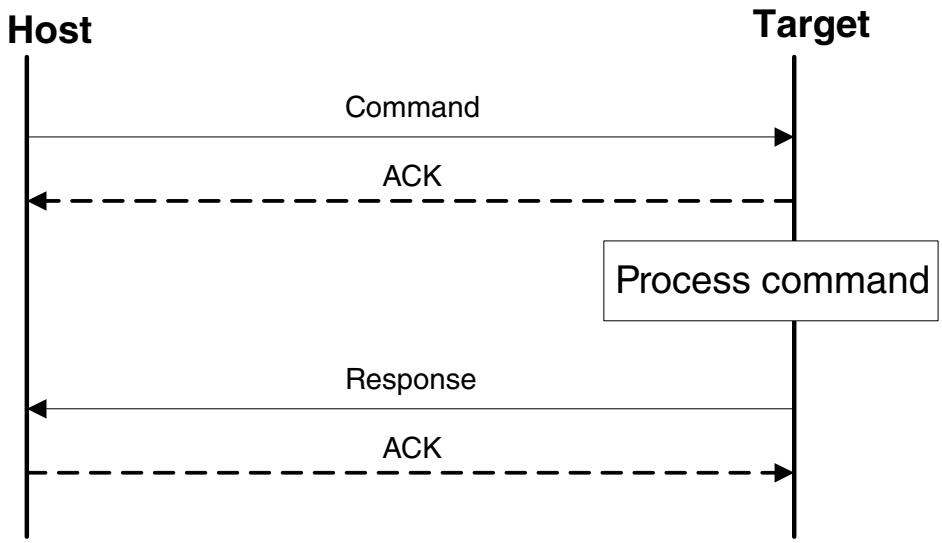



**Figure 27-2. Command with No Data Phase**

### 27.3.4.2 Command with incoming data phase

The protocol for a command with an incoming data phase contains:

- Command packet (from host)
- Generic response command packet (to host)
- Incoming data packets (from host)
- Generic response command packet (to host)

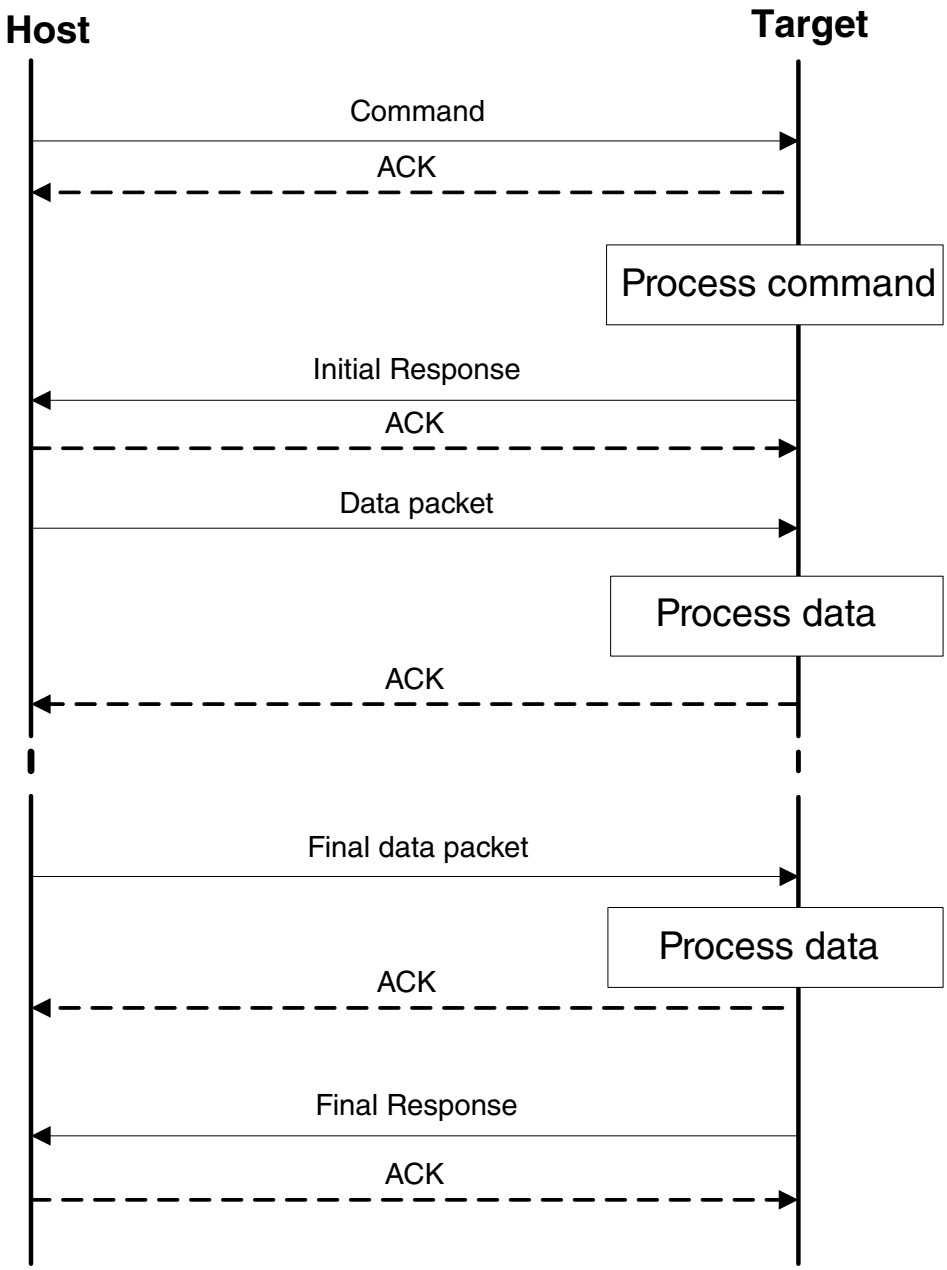



**Figure 27-3. Command with incoming data phase**

## NOTE

- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the Generic Response packet prior to the start of the data phase does not have a status of kStatus_Success, then the data phase is aborted.
- Data phases may be aborted by the receiving side by sending the final Generic Response early with a status of

kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

#### 27.3.4.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:
- Command packet (from host)
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set)
- Outgoing data packets (to host)
- Generic response command packet (to host)

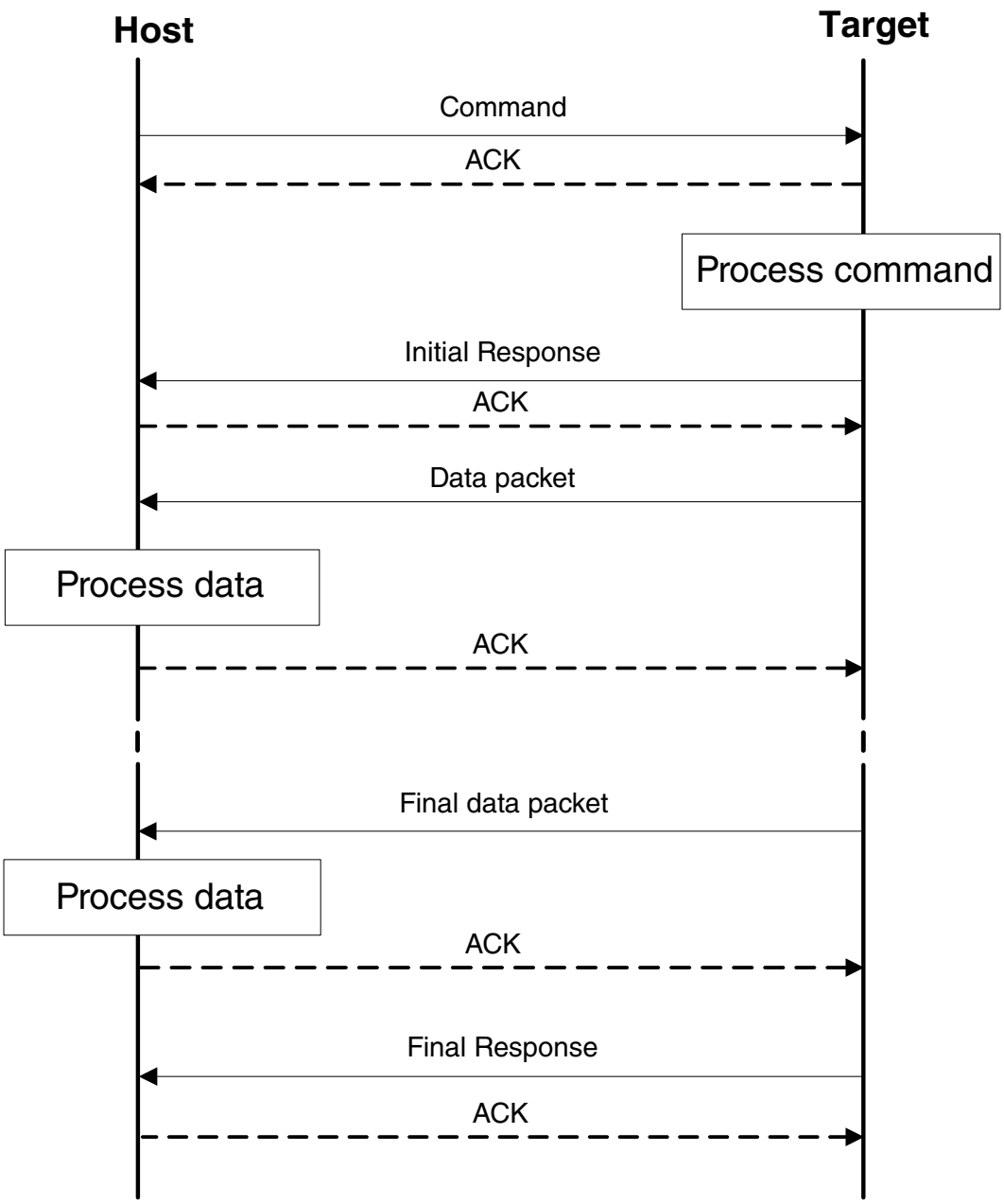



**Figure 27-4. Command with outgoing data phase**

## NOTE

- For the outgoing data phase sequence above, the data phase is really considered part of the response command.
- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the ReadMemory Response command packet prior to the start of the data phase does not contain the kCommandFlag_HasDataPhase flag, then the data phase is aborted.

- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

## 27.3.5 Bootloader Packet Types

The Kinetis Bootloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host . The Kinetis Bootloader device is the target, which receives a command or data packet. All data communication between host and target is packetized.

**NOTE**

The term "target" refers to the "Kinetis Bootloader device."

There are 6 types of packets used in the device:

- Ping packet
- Ping Response packet
- Framing packet
- Command packet
- Data packet
- Response packet

All fields in the packets are in little-endian byte order.

### 27.3.5.1 Ping packet

The Ping packet is the first packet sent from a host to the target (Kinetis Bootloader), to establish a connection on a selected peripheral. For a LPUART peripheral, the Ping packet is used to determine the baudrate. A Ping packet must be sent before any other communications. In response to a Ping packet, the target sends a Ping Response packet.

**Table 27-5. Ping Packet Format**

| Byte # | Value | Name |
|---|---|---|
| 0 | 0x5A | start byte |
| 1 | 0xA6 | ping |

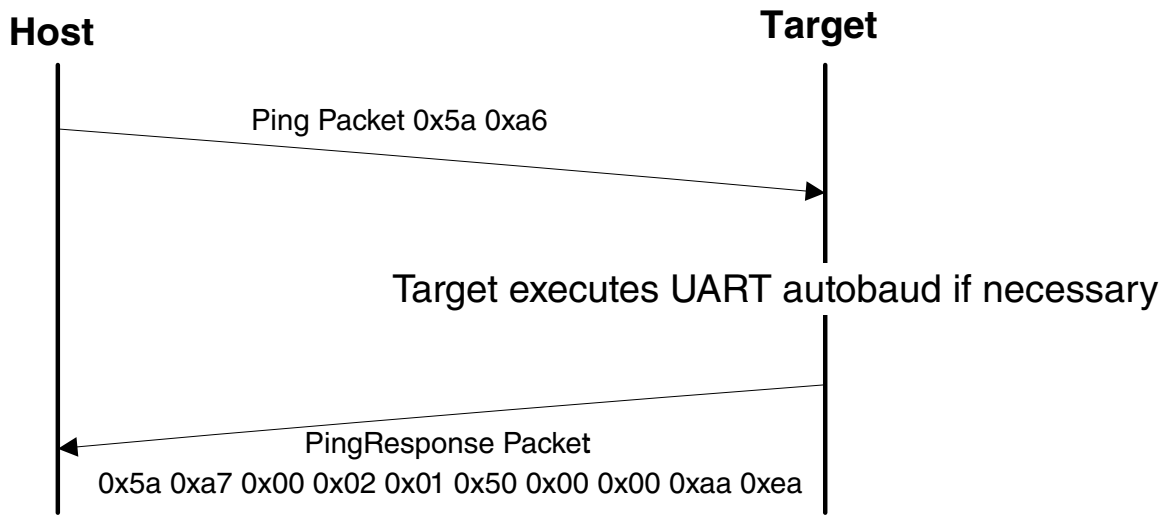



**Figure 27-5. Ping Packet Protocol Sequence**

### 27.3.5.2 Ping Response Packet

The target (Kinetis Bootloader) sends a Ping Response packet back to the host after receiving a Ping packet. If communication is over a LPUART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping Response packet. Once the Ping Response packet is received by the host, the connection is established, and the host starts sending commands to the target (Kinetis Bootloader).

**Table 27-6. Ping Response Packet Format**

| Byte # | Value | Parameter |
|---|---|---|
| 0 | 0x5A | start byte |
| 1 | 0xA7 | Ping response code |
| 2 | | Protocol bugfix |
| 3 | | Protocol minor |
| 4 | | Protocol major |
| 5 | | Protocol name = 'P' (0x50) |
| 6 | | Options low |
| 7 | | Options high |
| 8 | | CRC16 low |
| 9 | | CRC16 high |

### 27.3.5.3 Framing Packet

The framing packet is used for flow control and error detection, and it (the framing packet) wraps command and data packets as well.

The framing packet described in this section is used for serial peripherals including LPUART, LPI2C and LPSPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used. Please refer to the USB peripheral section for details.

**Table 27-7. Framing Packet Format**

| Byte # | Value | Parameter | |
|---|---|---|---|
| 0 | 0x5A | start byte | |
| 1 | | packetType | |
| 2 | | length_low | Length is a 16-bit field that specifies the entire command or data packet size in bytes. |
| 3 | | length_high | |
| 4 | | crc16_low | This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See the CRC16 algorithm after this table. |
| 5 | | crc16_high | |
| 6 . . .n | | Command or Data packet payload | |

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

**Table 27-8. Special Framing Packet Format**

| Byte # | Value | Parameter |
|---|---|---|
| 0 | 0x5A | start byte |
| 1 | 0xA*n* | packetType |

The Packet Type field specifies the type of the packet from one of the defined types (below):

**Table 27-9. packetType Field**

| packetType | Name | Description |
|---|---|---|
| 0xA1 | kFramingPacketType_Ack | The previous packet was received successfully; the sending of more packets is allowed. |
| 0xA2 | kFramingPacketType_Nak | The previous packet was corrupted and must be re-sent. |
| 0xA3 | kFramingPacketType_AckAbort | Data phase is being aborted. |
| 0xA4 | kFramingPacketType_Command | The framing packet contains a command packet payload. |
| 0xA5 | kFramingPacketType_Data | The framing packet contains a data packet payload. |

*Table continues on the next page...*

**Table 27-9. packetType Field (continued)**

| packetType | Name | Description |
|---|---|---|
| 0xA6 | kFramingPacketType_Ping | Sent to verify the other side is alive. Also used for UART autobaud. |
| 0xA7 | kFramingPacketType_PingResponse | A response to Ping; contains the framing protocol version number and options. |

## CRC16 algorithm:

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
        uint32_t byte = src[j];
        crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
            uint32_t temp = crc << 1;
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
            }
            crc = temp;
        }
    }
    return crc;
}
```

### 27.3.5.4 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

**Table 27-10. Command Packet Format**

| Command Packet Format (32 bytes) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Command Header (4 bytes) | | | | 28 bytes for Parameters (Max 7 parameters) | | | | | | |
| Tag | Flags | Rsvd | Param Count | Param1 (32-bit) | Param2 (32-bit) | Param3 (32-bit) | Param4 (32-bit) | Param5 (32-bit) | Param6 (32-bit) | Param7 (32-bit) |
| byte 0 | byte 1 | byte 2 | byte 3 | | | | | | | |

**Table 27-11. Command Header Format**

| Byte # | Command Header Field | |
|---|---|---|
| 0 | Command or Response tag | The command header is 4 bytes long, with these fields. |
| 1 | Flags | |
| 2 | Reserved. Should be 0x00. | |
| 3 | ParameterCount | |

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only 7 parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

**Table 27-12. Commands that are supported**

| Command | Name |
|---|---|
| 0x01 | FlashEraseAll |
| 0x02 | FlashEraseRegion |
| 0x03 | ReadMemory |
| 0x04 | WriteMemory |
| 0x05 | FillMemory |
| 0x06 | FlashSecurityDisable |
| 0x07 | GetProperty |
| 0x08 | ReceiveSBFile |
| 0x09 | Execute |
| 0x0A | Call |
| 0x0B | Reset |
| 0x0C | SetProperty |
| 0x0D | FlashEraseAllUnsecure |

*Table continues on the next page...*

**Table 27-12. Commands that are supported (continued)**

| Command | Name |
|---|---|
| 0x0E | FlashProgramOnce |
| 0x0F | FlashReadOnce |
| 0x10 | FlashReadResource |
| 0x11 | Reserved |
| 0x12 | ReliableUpdate |

**Table 27-13. Responses that are supported**

| Response | Name |
|---|---|
| 0xA0 | GenericResponse |
| 0xA3 | ReadMemoryResponse (used for sending responses to ReadMemory command only) |
| 0xA7 | GetPropertyResponse (used for sending responses to GetProperty command only) |
| 0xAF | FlashReadOnceResponse (used for sending responses to FlashReadOnce command only) |
| 0xB0 | FlashReadResourceResponse (used for sending responses to FlashReadResource command only) |

**Flags:** Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets will follow in the command sequence. The number of bytes that will be transferred in the data phase is determined by a command-specific parameter in the parameters array.

**ParameterCount:** The number of parameters included in the command packet.

**Parameters:** The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to 7 parameters.

### 27.3.5.5 Data packet

The data packet carries just the data, either host sending data to target, or target sending data to host. The data transfer direction is determined by the last command sent from the host. The data packet is also wrapped within a framing packet, to ensure the correct packet data is received.

The contents of a data packet are simply the data itself. There are no other fields, so that the most data per packet can be transferred. Framing packets are responsible for ensuring that the correct packet data is received.

### 27.3.5.6 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- FlashReadResourceResponse

**GenericResponse:** After the Kinetis Bootloader has processed a command, the bootloader will send a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

**Table 27-14. GenericResponse Parameters**

| Byte # | Parameter | Descripton |
|---|---|---|
| 0 - 3 | Status code | The Status codes are errors encountered during the execution of a command by the target (Kinetis Bootloader). If a command succeeds, then a kStatus_Success code is returned. Table 27-99, Kinetis Bootloader Status Error Codes, lists the status codes returned to the host by the Kinetis Bootloader for ROM. |
| 4 - 7 | Command tag | The Command tag parameter identifies the response to the command sent by the host. |

**GetPropertyResponse:** The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

**Table 27-15. GetPropertyResponse Parameters**

| Byte # | Value | Parameter |
|---|---|---|
| 0 - 3 | | Status code |
| 4 - 7 | | Property value |

*Table continues on the next page...*

**Table 27-15. GetPropertyResponse Parameters (continued)**

| Byte # | Value | Parameter |
|---|---|---|
| . . . | | . . . |
| | | Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type. |

**ReadMemoryResponse:** The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to 2 for the status code and the data byte count parameters shown below.

**Table 27-16. ReadMemoryResponse Parameters**

| Byte # | Parameter | Descripton |
|---|---|---|
| 0 - 3 | Status code | The status of the associated Read Memory command. |
| 4 - 7 | Data byte count | The number of bytes sent in the data phase. |

**FlashReadOnceResponse:**The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to 2 plus *the number of words* requested to be read in the FlashReadOnceCommand.

**Table 27-17. FlashReadOnceResponse Parameters**

| Byte # | Value | Parameter |
|---|---|---|
| 0 – 3 | | Status Code |
| 4 – 7 | | Byte count to read |
| … | | … |
| | | Can be up to 20 bytes of requested read data. |

The FlashReadResourceResponse packet is sent by the target in response to the host sending a FlashReadResource command. The FlashReadResourceResponse packet contains the framing packet data and command packet data, with the command/response tag set to a FlashReadResourceResponse tag value (0xB0), and the flags field set to kCommandFlag_HasDataPhase (1).

**Table 27-18. FlashReadResourceResponse Parameters**

| Byte # | Value | Parameter |
|---|---|---|
| 0 – 3 | | Status Code |
| 4 – 7 | | Data byte count |

## 27.3.6 Bootloader Command API

All Kinetis Bootloader command APIs follow the command packet format that is wrapped by the framing packet, as explained in previous sections.

- For a list of commands supported by the Kinetis Bootloader in ROM, see Table 27-2, Commands supported.
- For a list of status codes returned by the Kinetis Bootloader in ROM, see Table 27-99, Kinetis Bootloader Status Error Codes.

**NOTE**

All the examples in this section depict byte traffic on serial peripherals that use framing packets. USB HID transactions use the USB HID report packets instead of the serial framing packets shown in this section. Please refer to the HID reports section for details of the USB HID packet structure.

### 27.3.6.1 ReliableUpdate command

The ReliableUpdate command performs the reliable update operation.

- **For a software implementation:** the backup application address is the parameter that is required for the ReliableUpdate command. If the *backup address* is set to 0, then the bootloader uses the *predefined address*.
- **For a hardware implementation:** the swap indicator address is the parameter that is required for the ReliableUpdate command.

- If the flash swap system is **uninitialized**, then the swap indicator address can be arbitrarily specified.
- If the flash swap system is **initialized**, then the swap indicator must be aligned with the swap system.

**Table 27-19. Parameters for ReliableUpdate command**

| Byte number | Command |
|---|---|
| 0 - 3 | • **For a software implementation:** the value is the backup application address.<br>• **For a hardware implementation:** the value is the swap indicator address. |

**Response:** The target returns a GenericResponse packet with a status code either set to kStatus_Success upon a successful execution of the command, or set to an appropriate error status code.

### 27.3.6.2 Call command

The Call command will execute a function that is written in memory at the address sent in the command. The address needs to be a valid memory location residing in accessible flash (internal or external) or in RAM. The command supports the passing of one 32-bit argument. Although the command supports a stack address, at this time the call will still take place using the current stack pointer. After execution of the function, a 32-bit return value will be returned in the generic response message.

**Table 27-20. Parameters for Call Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Call address |
| 4 - 7 | Argument word |
| 8 - 11 | Stack pointer |

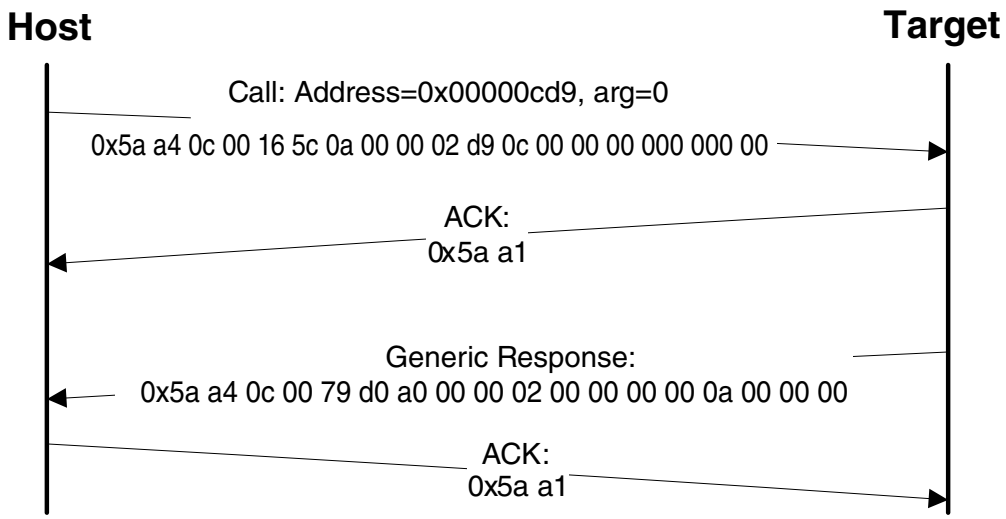



**Figure 27-6. Protocol Sequence for Call Command**

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to the return value of the function called or set to kStatus_InvalidArgument (105).

### 27.3.6.3 ReceiveSBFile command

The ReceiveSBFile command (ReceiveSbFile) will start the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that will be sent in the data phase. The SB file will be processed as it is received by the Bootloader .

**Table 27-21. Parameters for ReceiveSBFile Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Byte count |

**Data Phase:** The Receive SB file command has a data phase; the host will send data packets until the number of bytes of data specified in the byteCount parameter of the Receive SB File command are received by the target.

**Response:** The target (Bootloader) will return a GenericResponse packet with a status code set to the kStatus_Success upon successful execution of the command, or set to an appropriate error code.

### 27.3.6.4 Execute command

The execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command.

**Table 27-22. Parameters for Execute Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Jump address |
| 4 - 7 | Argument word |
| 8 - 11 | Stack pointer address |

The Execute command has no data phase.

**Response:** Before executing the Execute command, the target (Kinetis Bootloader) will validate the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

### 27.3.6.5 Reset command

The Reset command will result in bootloader resetting the chip.

The Reset command requires no parameters.

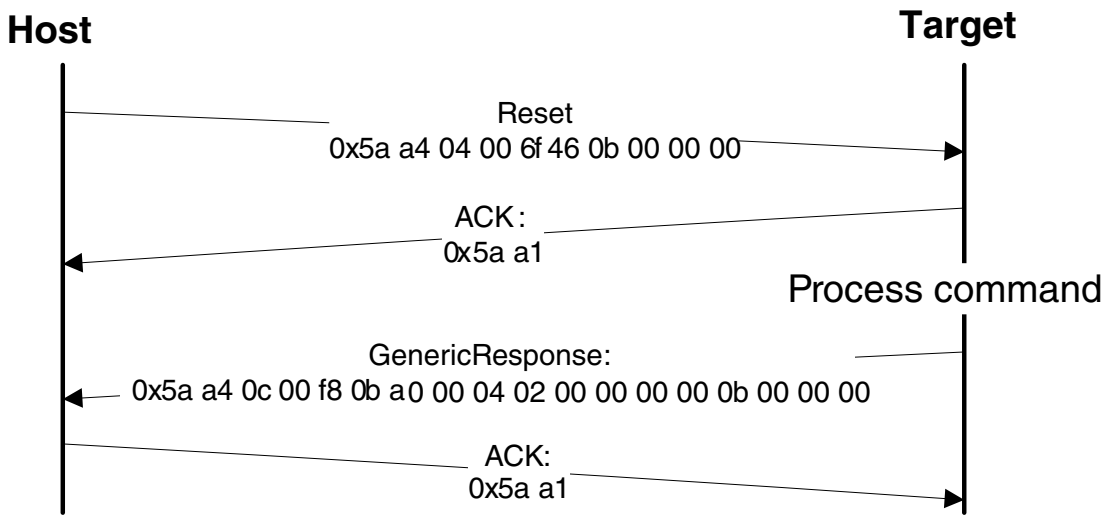



**Figure 27-7. Protocol Sequence for Reset Command**

**Table 27-23. Reset Command Packet Format (Example)**

| Reset | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x04 0x00 |
| | crc16 | 0x6F 0x46 |
| Command packet | commandTag | 0x0B - reset |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x00 |

The Reset command has no data phase.

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

## 27.3.6.6 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

For a list of properties and their associated 32-bit property tags supported by the Kinetis Bootloader in ROM, see Table 27-82.

The 32-bit property tag is the only parameter required for GetProperty command.

**Table 27-24. Parameters for GetProperty Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Property tag |

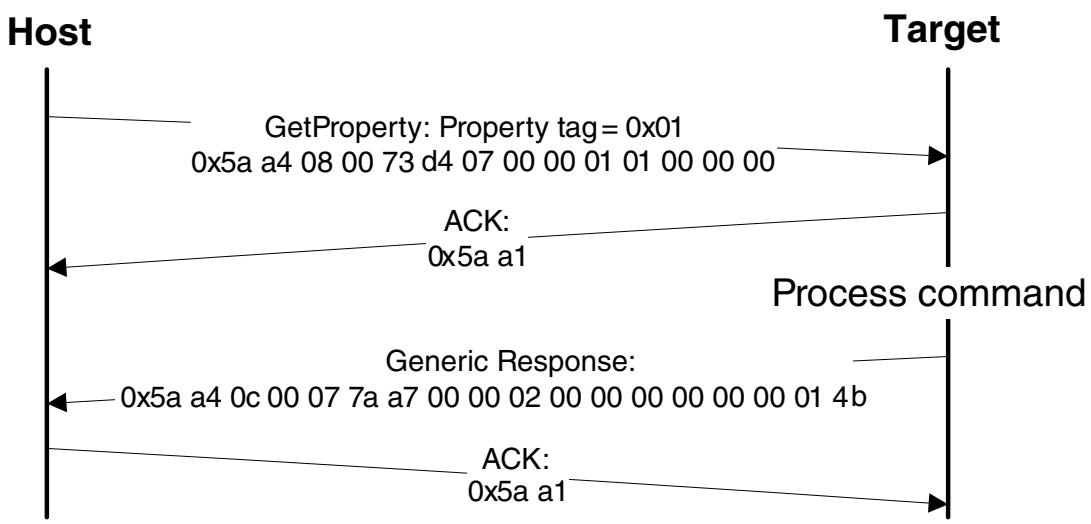



**Figure 27-8. Protocol Sequence for GetProperty Command**

**Table 27-25. GetProperty Command Packet Format (Example)**

| GetProperty | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x08 0x00 |
| | crc16 | 0x73 0xD4 |
| Command packet | commandTag | 0x07 – GetProperty |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x01 |
| | propertyTag | 0x00000001 - CurrentVersion |

The GetProperty command has no data phase.

**Response:** In response to a GetProperty command, the target will send a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). The next table shows an example of a GetPropertyResponse packet.

**Table 27-26. GetProperty Response Packet Format (Example)**

| GetPropertyResponse | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x0c 0x00 (12 bytes) |
| | crc16 | 0x07 0x7a |
| Command packet | responseTag | 0xA7 |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | status | 0x00000000 |
| | propertyValue | 0x0000014b - CurrentVersion |

### 27.3.6.7 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options in the Kinetis Bootloader ROM. However, the SetProperty command can only change the value of properties that are writable—see Table 27-82, Properties used by Get/SetProperty Commands. If you try to set a value for a read-only property, then the Kinetis Bootloader will return an error.

The property tag and the new value to set are the 2 parameters required for the SetProperty command.

**Table 27-27. Parameters for SetProperty Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Property tag |
| 4 - 7 | Property value |

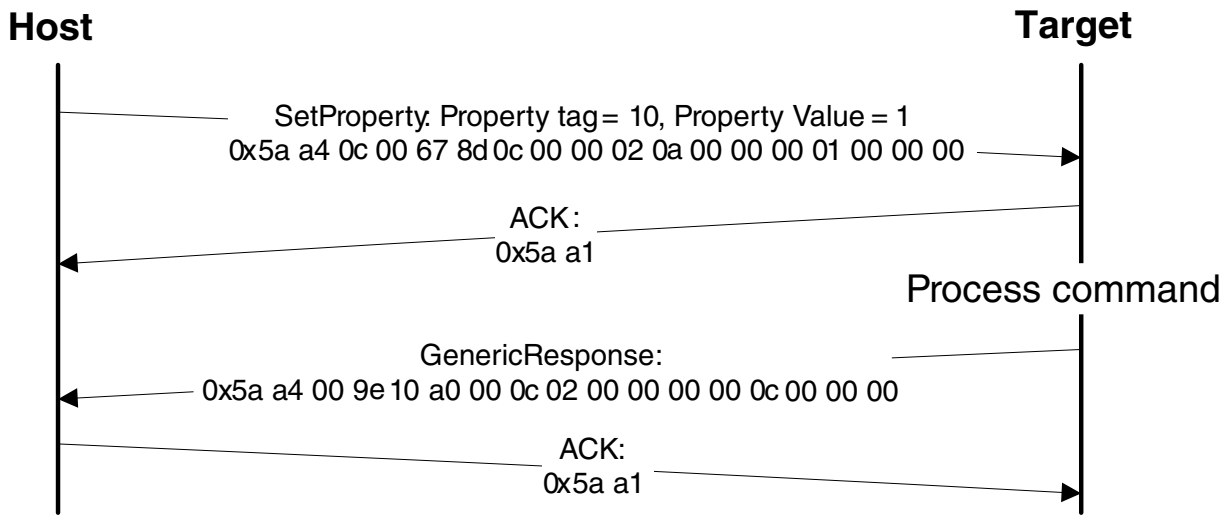



**Figure 27-9. Protocol Sequence for SetProperty Command**

**Table 27-28. SetProperty Command Packet Format (Example)**

| SetProperty | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x0C 0x00 |
| | crc16 | 0x67 0x8D |
| Command packet | commandTag | 0x0C – SetProperty with property tag 10 |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | propertyTag | 0x0000000A - VerifyWrites |
| | propertyValue | 0x00000001 |

The SetProperty command has no data phase.

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with one of following status codes:

**Table 27-29. SetProperty Response Status Codes**

| Status Code |
|---|
| kStatus_Success |
| kStatus_ReadOnly |
| kStatus_UnknownProperty |
| kStatus_InvalidArgument |

### 27.3.6.8 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command will fail and return an error status code. Executing the FlashEraseAll command will release flash security if it (flash security) was enabled, by setting the FTFE_FSEC register. However, the FSEC field of the flash configuration field is erased, so unless it is reprogrammed, the flash security will be re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires no parameters.

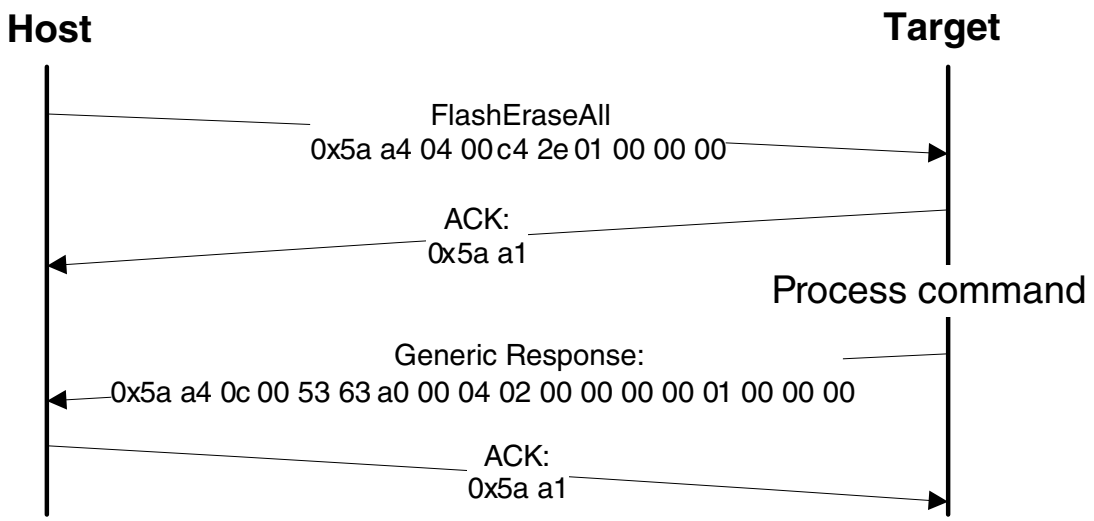



**Figure 27-10. Protocol Sequence for FlashEraseAll Command**

**Table 27-30. FlashEraseAll Command Packet Format (Example)**

| FlashEraseAll | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x04 0x00 |
| | crc16 | 0xC4 0x2E |
| Command packet | commandTag | 0x01 - FlashEraseAll |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x00 |
| | MemoryID | • If MemoryID = 0x00h, then internal flash.<br>• If MemoryID = 0x01h, then QSPI0 memory. |

The FlashEraseAll command has no data phase.

**Response:** The target (Kinetis Bootloader ) will return a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command, or set to an appropriate error status code.

### 27.3.6.9 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory or a specified range of flash within the connected SPI flash devices.

The start address and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be , or the FlashEraseRegion command will fail and return kStatus_FlashAlignmentError (0x101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus_FlashAddressError (0x102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus_MemoryRangeInvalid (0x10200).

**Table 27-31. Parameters for FlashEraseRegion Command**

| Byte # | Parameter |
|---|---|
| 0 - 3 | Start address |
| 4 - 7 | Byte count |

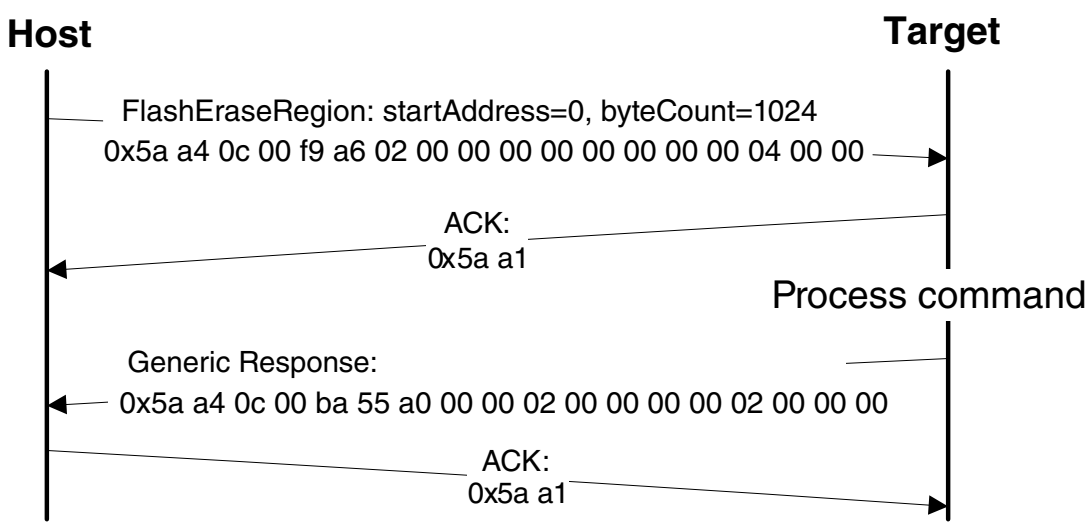



**Figure 27-11. Protocol Sequence for FlashEraseRegion Command**

**Table 27-32. FlashEraseRegion Command Packet Format (Example)**

| FlashEraseRegion | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x0C 0x00 |
| | crc16 | 0xF9 0x A6 |
| Command packet | commandTag | 0x02, kCommandTag_FlashEraseRegion |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | startAddress | 0x00 0x00 0x00 0x00 (0x0000_0000) |
| | byte count | 0x00 0x04 0x00 0x00 (0x400) |

The FlashEraseRegion command has no data phase.

**Response:** The target (Kinetis Bootloader ) will return a GenericResponse packet with one of following error status codes.

**Table 27-33. FlashEraseRegion Response Status Codes**

| Status Code |
|---|
| kStatus_Success (0x0) |
| kStatus_MemoryRangeInvalid (0x10200) |
| kStatus_FlashAlignmentError (0x101) |
| kStatus_FlashAddressError (0x102) |
| kStatus_FlashAccessError (0x103) |
| kStatus_FlashProtectionViolation (0x104) |
| kStatus_FlashCommandFailure (0x105) |

## 27.3.6.10 FlashEraseAllUnsecure command

The FlashEraseAllUnsecure command performs a mass erase of the flash memory, including protected sectors. Flash security is immediately disabled if it (flash security) was enabled, and the FSEC byte in the flash configuration field at address 0x40C is programmed to 0xFE. However, if the mass erase enable option in the FSEC field is disabled, then the FlashEraseAllUnsecure command will fail.

The FlashEraseAllUnsecure command requires no parameters.

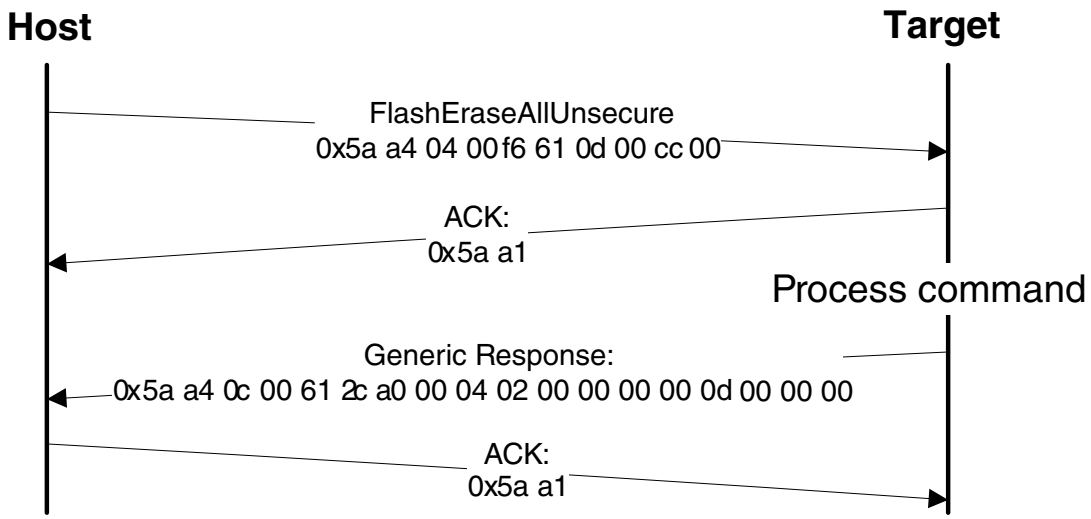



**Figure 27-12. Protocol Sequence for FlashEraseAll Command**

**Table 27-34. FlashEraseAllUnsecure Command Packet Format (Example)**

| FlashEraseAllUnsecure | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x04 0x00 |
| | crc16 | 0xF6 0x61 |
| Command packet | commandTag | 0x0D - FlashEraseAllUnsecure |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x00 |

The FlashEraseAllUnsecure command has no data phase.

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command, or set to an appropriate error status code.

### 27.3.6.11 FlashProgramOnce command

The FlashProgramOnce command writes data (that is provided in a command packet) to a specified range of bytes in the program once field. Special care must be taken when writing to the program once field.

- The program once field only supports programming once, so any attempted to reprogram a program once field will get an error response.
- Writing to the program once field requires the byte count to be 4-byte aligned or 8-byte aligned.

The FlashProgramOnce command uses 3 parameters: index, byteCount, data.

**Table 27-35. Parameters for FlashProgramOnce Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Index of program once field |
| 4 - 7 | Byte count (must be evenly divisible by 4) |
| 8 - 11 | Data |
| 12 - 16 | Data |

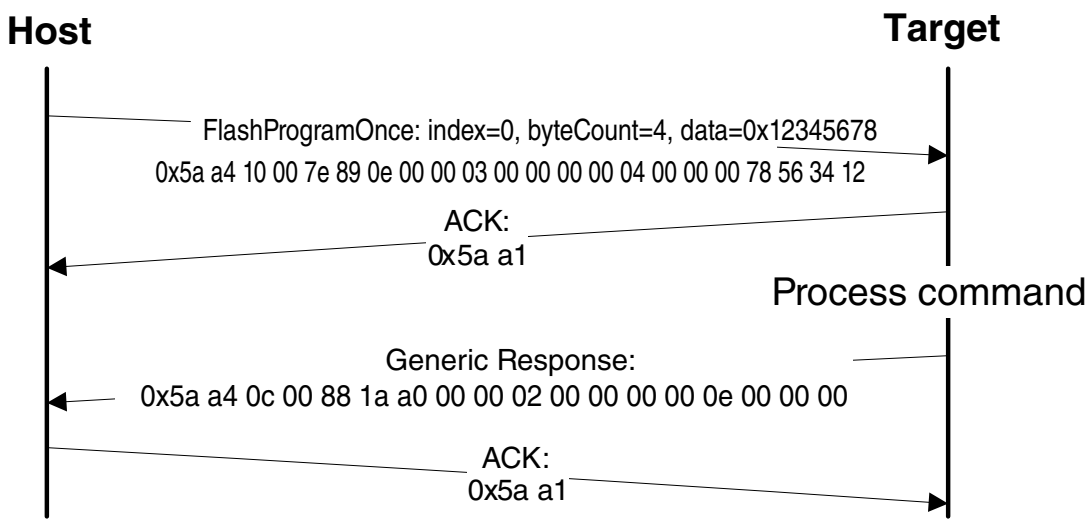



**Figure 27-13. Protocol Sequence for FlashProgramOnce Command**

**Table 27-36. FlashProgramOnce Command Packet Format (Example)**

| FlashProgramOnce | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x10 0x00 |
| | crc16 | 0x7E4 0x89 |
| Command packet | commandTag | 0x0E – FlashProgramOnce |
| | flags | 0 |
| | reserved | 0 |
| | parameterCount | 3 |
| | index | 0x0000_0000 |

*Table continues on the next page...*

**Table 27-36. FlashProgramOnce Command Packet Format (Example) (continued)**

| FlashProgramOnce | Parameter | Value |
|---|---|---|
| | byteCount | 0x0000_0004 |
| | data | 0x1234_5678 |

**Response:** upon successful execution of the command, the target (Kinetis Bootloader) will return a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

## 27.3.6.12 FlashReadOnce command

The FlashReadOnce command returns the contents of the program once field by given index and byte count. The FlashReadOnce command uses 2 parameters: index and byteCount.

**Table 27-37. Parameters for FlashReadOnce Command**

| Byte # | Parameter | Description |
|---|---|---|
| 0 - 3 | index | Index of the program once field (to read from) |
| 4 - 7 | byteCount | Number of bytes to read and return to the caller |

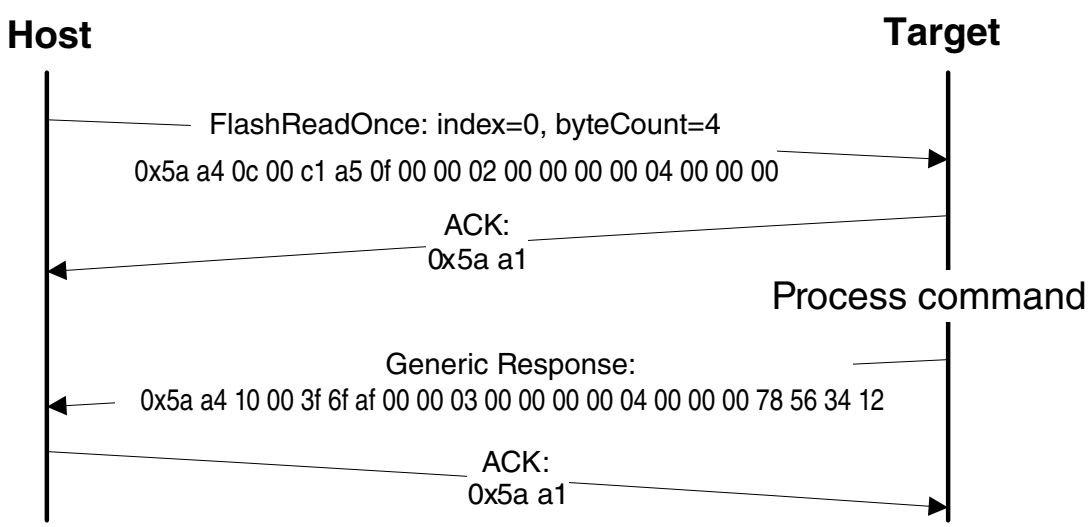



**Figure 27-14. Protocol Sequence for FlashReadOnce Command**

**Table 27-38. FlashReadOnce Command Packet Format (Example)**

| FlashReadOnce | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4 |
| | length | 0x0C 0x00 |
| | crc | 0xC1 0xA5 |
| Command packet | commandTag | 0x0F – FlashReadOnce |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | index | 0x0000_0000 |
| | byteCount | 0x0000_0004 |

**Table 27-39. FlashReadOnce Response Format (Example)**

| FlashReadOnce Response | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4 |
| | length | 0x10 0x00 |
| | crc | 0x3F 0x6F |
| Command packet | commandTag | 0xAF |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x03 |
| | status | 0x0000_0000 |
| | byteCount | 0x0000_0004 |
| | data | 0x1234_5678 |

**Response:** upon successful execution of the command, the target (Kinetis Bootloader) will return a FlashReadOnceResponse packet with a status code set to kStatus_Success, a byte count and corresponding data read from Program Once Field upon successful execution of the command, or will return with a status code set to an appropriate error status code and a byte count set to 0.

## 27.3.6.13 FlashReadResource command

The FlashReadResource command returns the contents of the IFR field or Flash Version ID, by given offset, byte count, and option. The FlashReadResource command uses 3 parameters: start address, byteCount, option.

**Table 27-40. Parameters for FlashReadResource Command**

| Byte # | Parameter | Command |
|---|---|---|
| 0 - 3 | start address | Start address of specific non-volatile memory to be read |
| 4 - 7 | byteCount | Byte count to be read |
| 8 - 11 | option | 0: IFR<br>1: Flash Version ID |

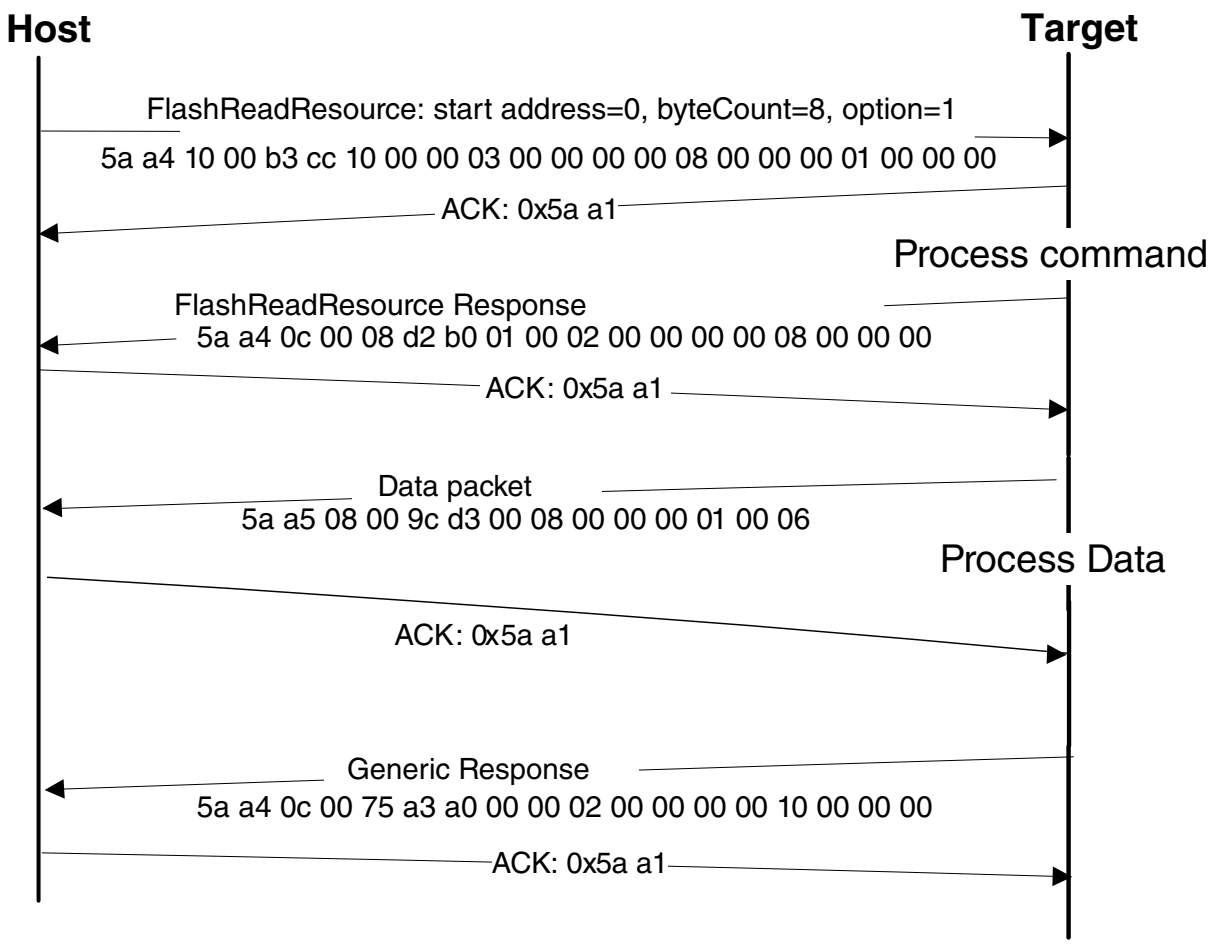



**Figure 27-15. Protocol Sequence for FlashReadResource Command**

**Table 27-41. FlashReadResource Command Packet Format (Example)**

| FlashReadResource | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4 |
| | length | 0x10 0x00 |
| | crc | 0xB3 0xCC |
| Command packet | commandTag | 0x10 – FlashReadResource |
| | flags | 0x00 |

*Table continues on the next page...*

**Table 27-41. FlashReadResource Command Packet Format (Example) (continued)**

| FlashReadResource | Parameter | Value |
|---|---|---|
| | reserved | 0x00 |
| | parameterCount | 0x03 |
| | startAddress | 0x0000_0000 |
| | byteCount | 0x0000_0008 |
| | option | 0x0000_0001 |

**Table 27-42. FlashReadResource Response Format (Example)**

| FlashReadResource Response | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4 |
| | length | 0x0C 0x00 |
| | crc | 0xD2 0xB0 |
| Command packet | commandTag | 0xB0 |
| | flags | 0x01 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | status | 0x0000_0000 |
| | byteCount | 0x0000_0008 |

**Data phase:** The FlashReadResource command has a data phase. Because the target (Kinetis Bootloader ) works in slave mode, the host must pull data packets until the number of bytes of data *specified in the byteCount parameter of FlashReadResource command* are received by the host.

## 27.3.6.14 FlashSecurityDisable command

The FlashSecurityDisable command performs the flash security disable operation, by comparing the 8-byte backdoor key (provided in the command) against the backdoor key stored in the flash configuration field (at address 0x400 in the flash).

The backdoor low and high words are the only parameters required for FlashSecurityDisable command.

**Table 27-43. Parameters for FlashSecurityDisable Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Backdoor key low word |

*Table continues on the next page...*

**Table 27-43. Parameters for FlashSecurityDisable Command (continued)**

| Byte # | Command |
|---|---|
| 4 - 7 | Backdoor key high word |

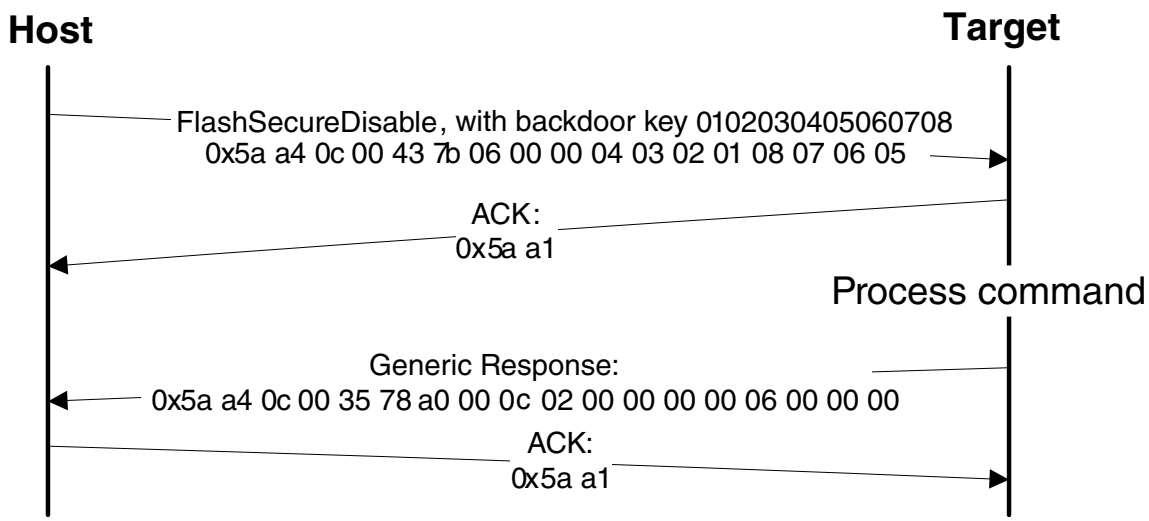



**Figure 27-16. Protocol Sequence for FlashSecurityDisable Command**

**Table 27-44. FlashSecurityDisable Command Packet Format (Example)**

| FlashSecurityDisable | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x0C 0x00 |
| | crc16 | 0x43 0x7B |
| Command packet | commandTag | 0x06 - FlashSecurityDisable |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | Backdoorkey_low | 0x04 0x03 0x02 0x01 |
| | Backdoorkey_high | 0x08 0x07 0x06 0x05 |

The FlashSecurityDisable command has no data phase.

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

### 27.3.6.15 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

**Table 27-45. Parameters for FillMemory Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Start address of memory to fill |
| 4 - 7 | Number of bytes to write with the pattern<br>• The start address should be 32-bit aligned.<br>• The number of bytes must be evenly divisible by 4. (Note: for a part that uses FTFE flash, the start address should be 64-bit aligned, and the number of bytes must be evenly divisible by 8). |
| 8 - 11 | 32-bit pattern |

- To fill with a byte pattern (8-bit), the byte must be replicated 4 times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated 2 times in the 32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern would be 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern would be 0x5AFE5AFE.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be .
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

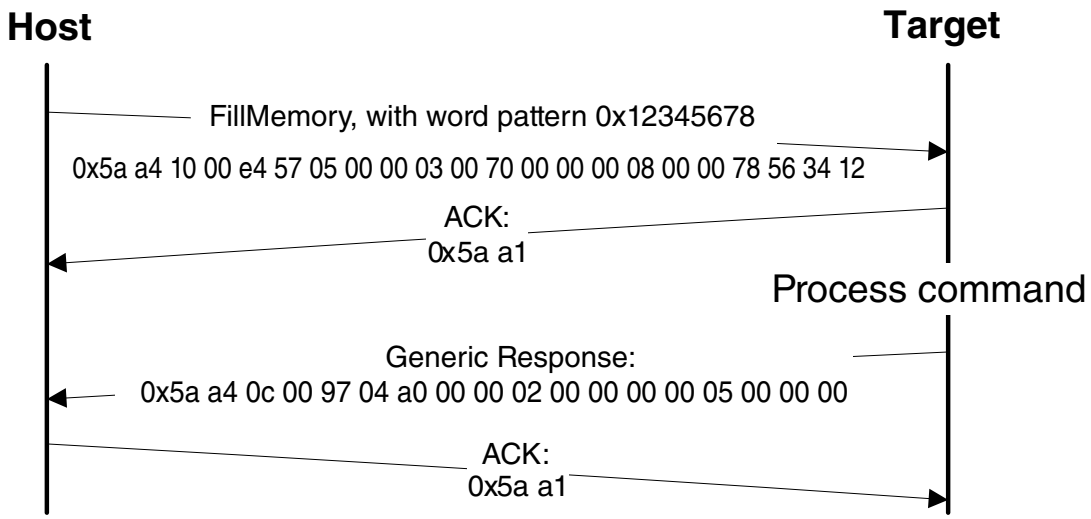



**Figure 27-17. Protocol Sequence for FillMemory Command**

**Table 27-46. FillMemory Command Packet Format (Example)**

| FillMemory | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x10 0x00 |
| | crc16 | 0xE4 0x57 |
| Command packet | commandTag | 0x05 – FillMemory |
| | flags | 0x00 |
| | Reserved | 0x00 |
| | parameterCount | 0x03 |
| | startAddress | 0x00007000 |
| | byteCount | 0x00000800 |
| | patternWord | 0x12345678 |

The FillMemory command has no data phase.

**Response:** upon successful execution of the command, the target (Kinetis Bootloader) will return a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

## 27.3.6.16 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors will fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be .
- The byte count will be rounded up to a multiple of , and the trailing bytes will be filled with the flash erase pattern (0xff).
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

The start address and number of bytes are the 2 parameters required for WriteMemory command.

**Table 27-47. Parameters for WriteMemory Command**

| Byte # | Command |
|---|---|
| 0 - 3 | Start address |
| 4 - 7 | Byte count |

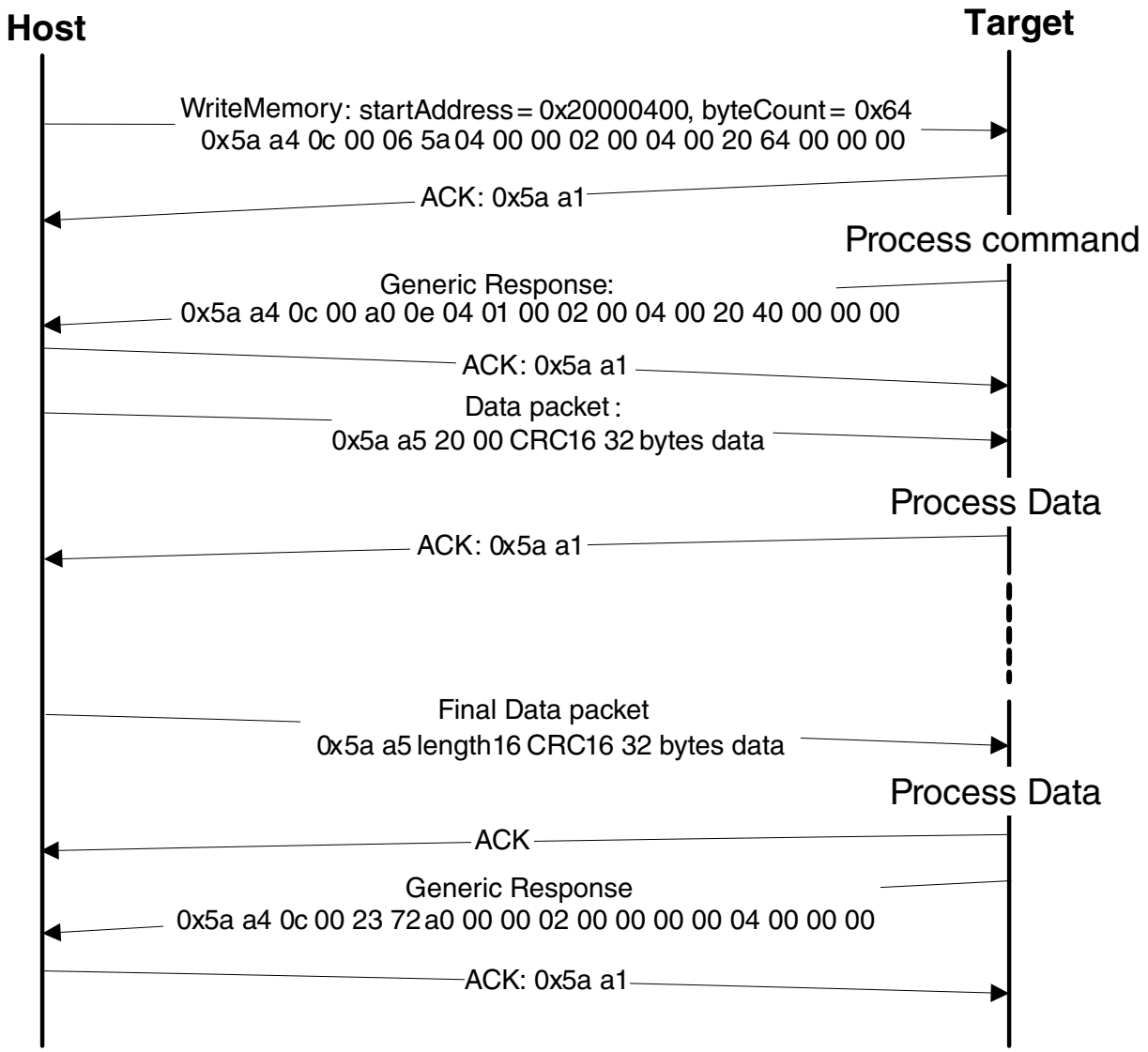



**Figure 27-18. Protocol Sequence for WriteMemory Command**

**Table 27-48. WriteMemory Command Packet Format (Example)**

| WriteMemory | Parameter | Value |
|---|---|---|
| Framing packet | start byte | 0x5A |
| | packetType | 0xA4, kFramingPacketType_Command |
| | length | 0x0C 0x00 |
| | crc16 | 0x06 0x5A |
| Command packet | commandTag | 0x04 - writeMemory |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | startAddress | 0x20000400 |
| | byteCount | 0x00000064 |

**Data Phase:** The WriteMemory command has a data phase; the host will send data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

**Response:** The target (Kinetis Bootloader ) will return a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

### 27.3.6.17 Read memory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address and number of bytes are the 2 parameters required for ReadMemory command.

**Table 27-49. Parameters for read memory command**

| Byte | Parameter | Description |
|---|---|---|
| 0-3 | Start address | Start address of memory to read from |
| 4-7 | Byte count | Number of bytes to read and return to caller |

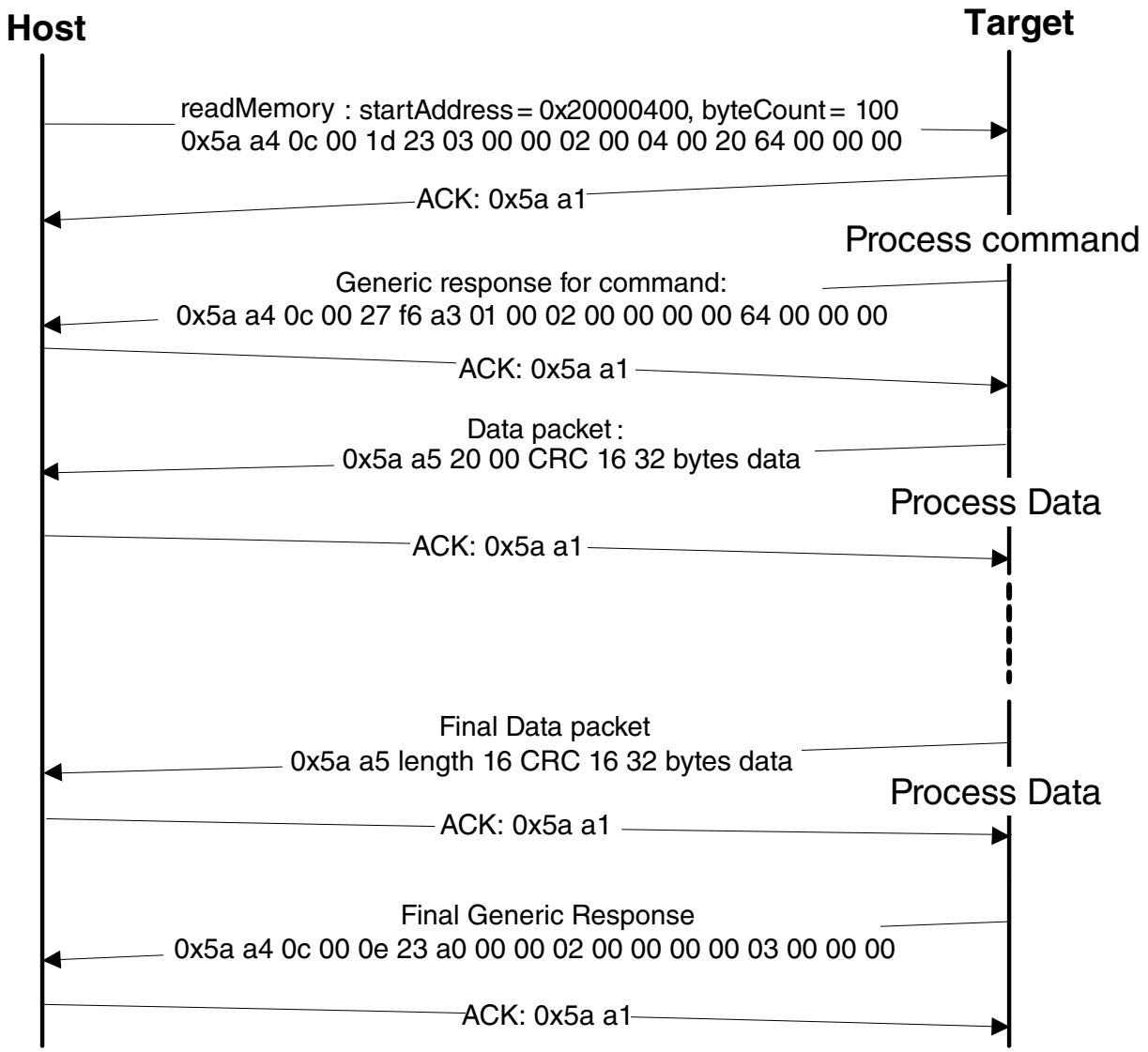



**Figure 27-19. Command sequence for read memory**

| ReadMemory | Parameter | Value |
|---|---|---|
| Framing packet | Start byte | 0x5A0xA4, |
| | packetType | kFramingPacketType_Command |
| | length | 0x0C 0x00 |
| | crc16 | 0x1D 0x23 |
| Command packet | commandTag | 0x03 - readMemory |
| | flags | 0x00 |
| | reserved | 0x00 |
| | parameterCount | 0x02 |
| | startAddress | 0x20000400 |
| | byteCount | 0x00000064 |

**Data Phase:** The ReadMemory command has a data phase. Since the target (Kinetis Bootloader) works in slave mode, the host need pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

**Response:** The target (Kinetis Bootloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

### 27.3.7 Bootloader Exit state

The Kinetis Bootloader tries to reconfigure the system back to the reset state in the following situations:

- After completion of an Execute command, but before jumping to the specified entry point.
- After a peripheral detection timeout, but before jumping to the application entry point.

In general, all peripherals are reset. However, the application must consider the following situations:

- If VLPR mode is active during a system boot, then the bootloader will exit VLPR mode on entry (to bootloader operation), and VLPR mode will not be restored after exiting from the bootloader.
- If high speed clocking is selected in the Bootloader Configuration Area, then high speed clocks will be retained after exiting from the bootloader.
- Upon exit from the bootloader, the bootloader restores the VTOR register to its default value (0x0).

## 27.4 Kinetis Flash Driver API

To simplify flash code development, the Kinetis ROM Bootloader has flash driver APIs that user applications can use, and provides pointers to where these APIs are located. This section describes how to use each flash driver API provided in the Kinetis flash driver API tree.

**NOTE**

For more information on how to use the ROM-resident Flash Driver API from an application space, see the “Kinetis Flash Driver API” chapter of the latest “Kinetis Bootloader Reference Manual,” http://www.nxp.com/KBOOT.

### 27.4.1 Flash Driver Entry Point

The MCU ROM bootloader provides a flash driver API tree entry (flashDriver) that a user application can use to get the entry points for the whole flash API set that is supported by the bootloader.

**NOTE**

The flashloader and flash-resident bootloader do not support this feature (flash driver API tree).

To get the address of the entry point, the user application reads the word containing the pointer to the bootloader API tree at offset 0x1C of the bootloader's vector table. The vector table is placed at the base of the bootloader's address range.

### 27.4.2 Flash driver API Tree

The bootloader API tree is a structure that contains pointers to other structures, which have the function and data address for the bootloader. The Flash driver API tree entry is always the 5th word of the API tree.

The prototype of the entry point is:

```
flash_driver_interface_t flashDriver;
```

There are several slightly different versions of the flash driver API among different targets with ROM bootloader.

**Table 27-50. Different versions of the flash driver**

| Flash driver API version | Supported targets |
|---|---|
| V1.0 | KL03Z4 KL43Z4 KL33Z4 KL27Z4 KL17Z4 |
| V1.1 | KL27Z644 KL17Z644 |
| V1.2 | KL13Z644 KL33Z644 K80F256 K81F256 K82F256 KL81Z7 KL82Z7 KL28Z7 |

There are minor differences in the flash driver interface among the flash driver API versions. See the definitions below.

```
typedef union BootloaderVersion
{
    struct
    {
        uint32_t bugfix : 8; //!lt; bugfix version [7:0]
        uint32_t minor : 8;  //!lt; minor version [15:8]
```

```
        uint32_t major : 8;  //!lt; major version [23:16]
        uint32_t name : 8;   //!lt; name [31:24]
    } B;
    uint32_t version; //!lt; combined version numbers
} standard_version_t;

//! @brief Interface for the flash driver.
typedef struct FlashDriverInterface
{
#if !defined(FLASH_API_TREE_1_0)
    standard_version_t version; //!lt; flash driver API version number.
#endif

    status_t (*flash_init)(flash_config_t *config);

#if defined(FLASH_API_TREE_1_0)
    status_t (*flash_erase_all)(flash_config_t *config);
    status_t (*flash_erase_all_unsecure)(flash_config_t *config);
    status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes);
#else
    status_t (*flash_erase_all)(flash_config_t *config, uint32_t key);
    status_t (*flash_erase_all_unsecure)(flash_config_t *config, uint32_t key);
    status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
uint32_t key);
#endif

    status_t (*flash_program)(flash_config_t *config, uint32_t start, uint32_t *src,
uint32_t lengthInBytes);
    status_t (*flash_get_security_state)(flash_config_t *config, flash_security_state_t
*state);
    status_t (*flash_security_bypass)(flash_config_t *config, const uint8_t *backdoorKey);
    status_t (*flash_verify_erase_all)(flash_config_t *config, flash_margin_value_t margin);
    status_t (*flash_verify_erase)(flash_config_t *config,
                                   uint32_t start,
                                   uint32_t lengthInBytes,
                                   flash_margin_value_t margin);
    status_t (*flash_verify_program)(flash_config_t *config,
                                     uint32_t start,
                                     uint32_t lengthInBytes,
                                     const uint32_t *expectedData,
                                     flash_margin_value_t margin,
                                     uint32_t *failedAddress,
                                     uint32_t *failedData);
    status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t
whichProperty, uint32_t *value);

#if (!defined(FLASH_API_TREE_1_0)) && (!defined(FLASH_API_TREE_1_1))
    status_t (*flash_register_callback)(flash_config_t *config, flash_callback_t callback);
    status_t (*flash_program_once)(flash_config_t *config, uint32_t index, uint32_t *src,
uint32_t lengthInBytes);
    status_t (*flash_read_once)(flash_config_t *config, uint32_t index, uint32_t *dst,
uint32_t lengthInBytes);
    status_t (*flash_read_resource)(flash_config_t *config,
                                    uint32_t start,
                                    uint32_t *dst,
                                    uint32_t lengthInBytes,
                                    flash_read_resource_option_t option);
#endif
} flash_driver_interface_t;
```

The Freescale/NXP standard flash driver (C90TFS flash driver) is the basis for the flash driver API.

## 27.4.3 Quick demo using Kinetis Flash Driver API

The example code below uses the Kinetis Flash Driver API to erase a region of flash memory. For more code examples, get the latest Kinetis bootloader package at http://www.nxp.com/KBOOT/

```
bootloader_tree_t* tree;                  // pointer points to bootloader tree
flash_config_t flash_config;            // variable used to keep runtime state of flash driver

// Get bootloader API tree from ROM
tree = (bootloader_tree_t*)(*(uint32_t*)0x0880001c);

// Init flash driver.
status_t status = tree->flashDriver->flash_init(&flash_config);
if  (status == kStatus_Success)
   {
// Erase flash region from 0x800 to 0xc00.
     status = tree->flashDriver->flash_erase(&flash_config, 0x800, 1024);
   }
```

## 27.4.4 Flash driver data structures

### 27.4.4.1 flash_config_t

The `flash_config_t` data structure is a required argument for all flash driver API functions. `flash_config_t` is initialized by calling FLASH_Init. For other functions, an initialized instance of this data structure should be passed as an argument.

**Table 27-51. `flash_config_t` data structure**

| Offset (hex) | Size | Field | Description |
|---|---|---|---|
| 0 | 4 | PFlashBlockBase | Base address of the first PFlash block |
| 4 | 4 | PFlashTotalSize | Size of all combined PFlash blocks |
| 8 | 1 | PFlashBlockCount | Number of PFlash blocks |
| 9 | 1 | FlashMemoryIndex | Index of flash memory |
| A | 1 | FlashCacheControllerIndex | Index of flash cache controller |
| B | 1 | Reserved | Reserved |
| C | 4 | PFlashSectorSize | Size (in bytes) of sector of PFlash |
| 10 | 4 | PFlashCallback | Pointer to a callback function used to do extra operations during erasure (for example, service watchdog) |
| 14 | 4 | PFlashAccessSegmentSize | Size of FAC access segment |
| 18 | 4 | PFlashAccessSegmentCount | Count of FAC access segment |
| 1C | 4 | flashExecuteInRamFunctionInfo | Info struct of flash execute-in-ram function |

*Table continues on the next page...*

**Table 27-51. `flash_config_t` data structure (continued)**

| Offset (hex) | Size | Field | Description |
|---|---|---|---|
| 20 | 4 | FlexRAMBlockBase | • **FlexNVM device**: FlexRAM base address<br>• **non-FlexNVM device**: acceleration RAM memory base address |
| 24 | 4 | FlexRAMTotalSize | • **FlexNVM device**: FlexRAM size<br>• **non-FlexNVM device**: acceleration RAM memory size |
| 28 | 4 | DFlashBlockBase | • **FlexNVM device**: D-Flash memory (FlexNVM memory) base address<br>• **non-FlexNVM device**: unused |
| 2C | 4 | DFlashTotalSize | • **FlexNVM device**: FlexNVM memory total size<br>• **non-FlexNVM device**: unused |
| 30 | 4 | EEpromTotalSize | • **FlexNVM device**: the size (in bytes) of the EEPROM area that was partitioned from FlexRAM<br>• **non-FlexNVM device**: unused |

### `flash_config_t` prototype:

```
typedef struct _flash_config
{
    uint32_t PFlashBlockBase;                /*!< Base address of the first PFlash block */
    uint32_t PFlashTotalSize;                /*!< Size of all combined PFlash block. */
    uint8_t PflashBlockCount;                /*!< Number of PFlash blocks. */
    uint8_t FlashMemoryIndex;                /*!< Index of flash memory. */
    uint8_t FlashCacheControllerIndex        /*!< Index of flash cache controller. */
    uint8_t Reserved0;                       /*!< Reserved */
    uint32_t PFlashSectorSize;               /*!< Size in bytes of a sector of PFlash. */
    flash_callback_t PFlashCallback;         /*!< Callback function for flash API. */
    uint32_t PFlashAccessSegmentSize;        /*!< Size in bytes of a access segment of
PFlash. */
    uint32_t PFlashAccessSegmentCount;       /*!< Number of PFlash access segments. */
    uint32_t *flashExecuteInRamFunctionInfo; /*!< Info struct of flash execute-in-ram
function. */
    uint32_t FlexRAMBlockBase;               /*!< For FlexNVM device, this is the base
address of FlexRAM
                                                  For non-FlexNVM device, this is the base
address of acceleration RAM memory */
    uint32_t FlexRAMTotalSize;               /*!< For FlexNVM device, this is the size of
FlexRAM
                                                  For non-FlexNVM device, this is the size
of acceleration RAM memory */
    uint32_t DFlashBlockBase; /*!< For FlexNVM device, this is the base address of D-Flash
memory (FlexNVM memory);
                                   For non-FlexNVM device, this field is unused */
    uint32_t DFlashTotalSize; /*!< For FlexNVM device, this is total size of the FlexNVM
memory;
                                   For non-FlexNVM device, this field is unused */
    uint32_t EEpromTotalSize; /*!< For FlexNVM device, this is the size in byte of EEPROM
area which was partitioned
                                 from FlexRAM;
                                   For non-FlexNVM device, this field is unused */
} flash_config_t;
```

## 27.4.5 Flash driver API

This section describes each function supported in the flash driver API.

### 27.4.5.1 FLASH_Init

Checks and initializes the flash module for the other flash API functions.

**NOTE**

FLASH_Init must be always called before calling other API functions.

#### Prototype:

```
status_t FLASH_Init(flash_config_t *config);
```

**Table 27-52. Parameters**

| Parameter | Description |
|---|---|
| config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |

**Table 27-53. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config pointer is NULL. |
| 100 | kStatus_FLASH_SizeError | Returned flash is incorrect. |
| 0 | kStatus_Success | This function has performed successfully. |

#### Example:

```
flash_config_t flashInstance;
status_t status = FLASH_Init(&flashInstance);
```

### 27.4.5.2 FLASH_EraseAll

Erases the entire flash array.

#### Prototype:

```
status_t FLASH_EraseAll(flash_config_t *config, uint32_t key);
```

**Table 27-54. Parameters**

| Parameter | Description |
|---|---|
| config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| key | Key used to validate erase operation. Must be set to 0x6B65666B. |

**Table 27-55. Possible status response**

| Value | Constants | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config pointer is NULL. |
| 103 | kStatus_FLASH_AccessError | Command is not available under current mode/ security. |
| 104 | kStatus_FLASH_ProtectionViolation | Any region of the program flash memory is protected. |
| 107 | kStatus_FLASH_EraseKeyError | Key is incorrect. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

```
status_t status = FLASH_EraseAll(&flashInstance, kFLASH_ApiEraseKey);
```

## 27.4.5.3 FLASH_EraseAllUnsecure

Erases the entire flash (including protected sectors) and restores flash to unsecured mode.

### Prototype:

```
status_t FLASH_EraseAllUnsecure(flash_config_t *config, uint32_t key);
```

**Table 27-56. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Key | Key used to validate erase operation. Must be set to 0x6B65666B. |

**Table 27-57. Possible Status Response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config pointer is NULL. |
| 103 | kStatus_FLASH_AccessError | Command is not available under current mode/ security. |

*Table continues on the next page...*

**Table 27-57. Possible Status Response (continued)**

| Value | Constant | Description |
|---|---|---|
| 107 | kStatus_FLASH_EraseKeyError | Key is incorrect. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

```
status_t status = FLASH_EraseAllUnsecure(&flashInstance, kFLASH_ApiEraseKey);
```

## 27.4.5.4 FLASH_Erase

Erases expected flash sectors specified by parameters. For Kinetis devices, the minimum erase unit is one sector.

### Prototype:

```
status_t FLASH_Erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t
      key);
```

**Table 27-58. Parameters**

| Parameters | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Start | The start address of the desired flash memory to be erased.<br>The start address does not need to be sector aligned, but must be word-aligned. |
| lengthInBytes | The length, given in bytes (not words or long words) to be erased. Must be word-aligned. |
| Key | Key is used to validate erase operation. Must be set to 0x6B65666B. |

**Table 27-59. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config pointer is NULL. |
| 100 | kStatus_FLASH_AlignmentError | Start or lengthInBytes; is not long word-aligned. |
| 102 | kStatus_FLASH_AddressError | The range to be erased is not a valid flash range. |
| 103 | kStatus_FLASH_AccessError | Command is not available under current mode/ security. |
| 104 | kStatus_FLASH_ProtectionViolation | The selected program flash sector is protected. |
| 107 | kStatus_FLASH_EraseKeyError | Key is incorrect. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

```
status_t status = FLASH_Erase (&flashInstance, 0x800, 1024, kFLASH_ApiEraseKey);
```

## 27.4.5.5 FLASH_Program

Programs the flash memory with data at locations that are passed in using parameters.

### Prototype:

```
status_t FLASH_Program(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t
      lengthInBytes);
```

**Table 27-60. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Start | The start address of the desired flash memory to be erased.<br>The start address does not need to be sector-aligned, but the start address must be word-aligned. |
| src | Pointer to the source buffer of data that is to be programmed into flash. |
| lengthInBytes | The length in bytes (not words or long words) to be erased; the length must also be word-aligned. |

**Table 27-61. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or src pointers are NULL. |
| 101 | kStatus_FLASH_AlignmentError | Start or lengthInBytes is not longword aligned. |
| 102 | kStatus_FLASH_AddressError | The range to be programmed is invalid. |
| 103 | kStatus_FLASH_AccessError | Command is not available under current mode/ security. |
| 104 | kStatus_FLASH_ProtectionViolation | The selected program flash address is protected. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

```
uint32_t m_content[] = {0x01234567, 0x89abcdef};
status_t status = FLASH_Program (&flashInstance, 0x800, &m_content[0], sizeof(m_content));
```

**NOTE**

Before calling `flash_program`, make sure that the region to be programmed is empty and is not protected.

### 27.4.5.6 FLASH_GetSecurityState

Retrieves the current flash security status, including the security enabling state and the backdoor key enabling state.

#### Prototype:

```
status_t FLASH_GetSecurityState(flash_config_t *config, flash_security_state_t *state);
```

**Table 27-62. Parameters**

| Parameters | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| State | Pointer to the value returned for the current security status code:<br>**Table 27-63. Returned value**<br>kFLASH_SecurityStateNotSecure — 0 — Flash is under unsecured mode.<br>kFLASH_SecurityStateBackdoorEnabled — 1 — Flash is under secured mode and Backdoor is enabled.<br>kFLASH_SecurityStateBackdoorDisabled — 2 — Flash is under secured mode and Backdoor is disabled. |

**Table 27-64. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or state pointers are NULL. |
| 0 | kStatus_Success | This function has performed successfully. |

#### Example:

```
flash_security_state_t state;
status_t status = FLASH_GetSecurityState (&flashInstance, &state);
```

### 27.4.5.7 FLASH_SecurityBypass

Allows the user to bypass security with a backdoor key. If the MCU is in a secured state, then the FLASH_SecurityBypass function unsecures the MCU, by comparing the provided backdoor key with keys in the Flash Configuration Field.

#### Prototype:

```
status_t FLASH_SecurityBypass(flash_config_t *config, const uint8_t *backdoorKey);
```

**Table 27-65. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| backdoorKey | Pointer to the user buffer containing the backdoor key. |

**Table 27-66. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or backdoorKey pointers are NULL. |
| 103 | kStatus_FLASH_AccessError | The following condition causes this return value:<br>1. An incorrect backdoor key is supplied<br>2. Backdoor key access has not been enabled. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

Assume that the flash range from 0x400 to 0x40c contains the following content after the last reset, which means that the backdoor key is valid and the backdoor key access has been enabled.

0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0xff 0xff 0xff 0xbf

```
uint8_t backdoorKey[] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88};
status_t status = FLASH_SecurityBypass (&flashInstance, & backdoorKey[0]);
```

## 27.4.5.8 FLASH_VerifyEraseAll

Checks if the entire flash has been erased to the specified read margin level.

To verify if the entire flash has been fully erased (after executing an FLASH_EraseAll), call FLASH_VerifyEraseAll.

### Prototype:

```
status_t FLASH_VerifyEraseAll(flash_config_t *config, flash_margin_value_t margin);
```

**Table 27-67. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |

*Table continues on the next page...*

**Table 27-67. Parameters (continued)**

| Parameter | Description |
|---|---|
| Margin1 | Read margin choice:<br>• kFLASH_MarginValueNormal 0<br>• kFLASH_MarginValueUser 1<br>• kFLASH_MarginValueFactory 2 |

**Table 27-68. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or backdoorKey pointers are NULL. |
| 103 | kStatus_FLASH_AccessError | An invalid margin choice is specified. |
| 105 | kStatus_FLASH_CommandFailure | The entire flash is not fully erased. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

Assume that flash_erase_all has been successfully executed.

```
status_t status = flash_verify_erase_all (&flashInstance, kFLASH_MarginValueUser);
```

**NOTE**

For the choice of margin, see the FTFA chapter in the reference manual for detailed information.

## 27.4.5.9 FLASH_VerifyErase

Verifies the erasure of the desired flash area at a specified margin level. This function checks the appropriate number of flash sectors based on the desired start address and length, to see if the flash has been erased at the specified read margin level.

FLASH_VerifyErase is often called after successfully performing the FLASH_Erase API.

### Prototype:

```
status_t FLASH_VerifyErase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
flash_margin_value_t margin);
```

**Table 27-69. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |

*Table continues on the next page...*

**Table 27-69. Parameters (continued)**

| Parameter | Description |
|---|---|
| Start | The start address of the desired flash memory to be verified. |
| lengthInBytes | The length, given in bytes (not words or long words) to be verified.<br>Must be word-aligned. |
| margin | Read margin choice as follows:<br>kFLASH_MarginValueNormal 0 kFLASH_MarginValueUser 1 kFLASH_MarginValueFactory 2 |

**Table 27-70. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or backdoorKey pointers are NULL. |
| 101 | kStatus_FLASH_AlignmentError | Start or lengthInBytes is not longword aligned. |
| 102 | kStatus_FLASH_AddressError | The range to be verified is not a valid flash range. |
| 103 | kStatus_FlashAccessError | The following situation causes this response:<br>1. Command is not available under current mode/ security<br>2. An invalid margin code is provided<br>3. The requested number of bytes is 0<br>4. The requested sector crosses a flash block boundary |
| 105 | kStatus_FLASH_CommandFailure | The flash range to be verified is not fully erased. |
| 0 | kStatus_Success | This function has performed successfully. |

**Example:**

Assume that flash region from 0x800 to 0xc00 has been successfully erased.

```
status_t status = FLASH_VerifyErase(&flashInstance, 0x800, 1024, kFLASH_MarginValueUser);
```

**NOTE**

For the choice of margin, see the FTFA chapter in the reference manual for detailed information.

### 27.4.5.10 FLASH_VerifyProgram

Verifies the data programmed in the flash memory (using the Flash Program Check Command), and compares it with expected data for a given flash area (as determined by the start address and length).

FLASH_VerifyProgram is often called after successfully doing FLASH_Program().

## Prototype:

```
status_t FLASH_VerifyProgram(flash_config_t *config,
                             uint32_t start,
                             uint32_t lengthInBytes,
                             const uint32_t *expectedData,
                             flash_margin_value_t margin,
                             uint32_t *failedAddress,
                             uint32_t *failedData);
```

**Table 27-71. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Start | The start address of the desired flash memory to be verified. |
| LengthInBytes | The length, given in bytes (not words or long-words) to be verified. Must be word-aligned. |
| ExpectedData | Pointer to the expected data that is to be verified against. |
| Margin | Read margin choice as follows:<br>kFLASH_MarginValueUser 1 kFLASH_MarginValueFactory 2 |
| FailedAddress | Pointer to returned failing address. |
| FailedData | Pointer to return failing data. Some derivatives do not include failed data as part of the FCCOBx registers. In this instance, 0x00s are returned upon failure. |

**Table 27-72. Possible status response**

| Value | Contants | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or expectedData pointers are NULL. |
| 101 | kStatus_FlashAlignmentError | Start or lengthInBytes is not longword-aligned. |
| 102 | kStatus_FLASH_AddressError | The range to be verified is invalid. |
| 103 | kStatus_FLASH_AccessError | The following situation causes this response:<br>1. Command is not available under current mode/ security.<br>2. An invalid margin code is supplied. |
| 105 | kStatus_FLASH_CommandFailure | Either of the margin reads does not match the expected data. |
| 0 | kStatus_Success | This function has performed successfully. |

## Example:

Assume that flash region from 0x800 to 0x807 is successfully programmed with:

0x01 0x23 0x45 0x67 0x89 0xab 0xcd 0xef

```
uint8_t expectedData[] = {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef };
status_t status = FLASH_VerifyProgram (&flashInstance, 0x800, 8,
                  &expectedData[0], kFlashMargin_User, NULL, NULL);
```

**NOTE**

For the choice of margin, see the FTFA chapter in the reference manual for detailed information.

### 27.4.5.11 FLASH_GetProperty

Returns the desired flash property, which includes base address, sector size, and other options.

## Prototype:

```
status_t flash_get_property(flash_driver_t * driver, flash_property_t whichProperty, uint32_t
* value);
```

**Table 27-73. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory to store driver runtime state. |
| whichProperty | The desired property from the list of properties.<br>**Table 27-74. Properties**<br>Definition / Value / Description:<br>kFLASH_PropertyPflashSectorSize / 0 / Get Flash Sector size<br>kFLASH_PropertyPflashTotalSize / 1 / Get total flash size<br>kFLASH_PropertyPflashBlockBaseAddr / 4 / Get flash base address<br>kFLASH_PropertyPflashFacSupport / 5 / Get FAC support status<br>kFLASH_PropertyPflashAccessSegmentSize / 6 / Get FAC segment size<br>kFLASH_PropertyPflashAccessSegmentCount / 7 / Get FAC segment count<br>kFLASH_PropertyVersion / 32 / Get version of Flash Driver API |
| Value | Pointer to the value returned for the desired flash property. |

**Table 27-75. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or value pointers are invalid. |
| 106 | kStatus_FLASH_UnknownProperty | Invalid property is supplied. |
| 0 | kStatus_Success | This function has performed successfully. |

## Example:

```
uint32_t propertyValue;
status_t status = FLASH_GetProperty (&flashInstance, kFLASH_PropertyPflashSectorSize,
&propertyValue);
```

## 27.4.5.12 FLASH_ProgramOnce

Programs a certain Program Once Field with the expected data for a given IFR region (as determined by the index and length).

- For each Program Once Field, FLASH_ProgramOnce can only allowed to be called once; otherwise, an error code is returned.
- For targets which do not support FLASH_ProgramOnce, the value of the FLASH_ProgramOnce pointer is 0.

### Prototype

```
status_t flash_program_once (flash_driver_t * driver, uint32_t index, uint32_t *src, uint32_t
      lengthInBytes);
```

**Table 27-76. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Index | Index for a certain Program Once Field. |
| src | Pointer to the source buffer of data that is to be programmed into the Program Once Field. |
| Lengthinbytes | The length, in bytes (not words or long words) to be programmed. Must be word-aligned. |

**Table 27-77. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or src pointers are NULL. |
| 101 | kStatus_FLASH_AlignmentError | index or lengthInBytes is invalid. |
| 103 | kStatus_FLASH_AddressError | The following situation causes this response:<br>1. Command is not available under current mode/ security.<br>2. An invalid index is supplied.<br>3. The requested Program Once field has already been programmed to a non-FFFF value.<br>4. The requested sector crosses a flash block boundary. |
| 115 | kStatus_FLASH_CommandNotSupported | This function is not supported. |
| 0 | kStatus_Success | This function has performed successfully. |

### Example:

Assume the Program Once Field has not been programmed before.

```
uint32_t expectedData = 0x78563412;

status_t status = FLASH_ProgramOnce(&flashInstance, 0, &expectedData, 4);
```

**NOTE**

For the choice of index and length, see the FTFA chapter in RM for detailed information.

### 27.4.5.13 FLASH_ReadOnce

Reads a certain flash Program Once Field according to parameters passed by index and length.

For targets that do not support FLASH_ReadOnce, the value of the FLASH_ReadOnce pointer is 0.

#### Prototype:

```
status_t flash_read_once (flash_driver_t * driver, uint32_t index, uint32_t *dst, uint32_t
lengthInBytes);
```

**Table 27-78. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Index | Index for a certain Program Once Field. |
| dst | Pointer to the destination buffer of data that stores data reads from the Program Once Field. |
| Lengthinbytes | The length, in bytes (not words or long words) to be read. Must be word-aligned. |

**Table 27-79. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or dst pointers are NULL. |
| 101 | kStatus_FlashAlignmentError | Index or lengthInBytes is invalid. |
| 103 | kStatus_FLASH_AddressError | The following situation causes this response:<br>1. Command is not available under current mode/ security.<br>2. An invalid index is supplied. |
| 115 | kStatus_FLASH_CommandNotSupported | This function is not supported. |
| 0 | kStatus_Success | This function has performed successfully. |

#### Example:

```
uint32_t temp;
      status_t status = FLASH_ReadOnce(&flashInstance, 0, &temp, 4);
```

**NOTE**

For the choice of index and length, see the FTFA chapter in RM for detailed information.

### 27.4.5.14 FLASH_SetCallback

Registers (like to write into a list) expected callback functions into the flash driver, for example, like a function that services a watchdog.

**Prototype:**

```
status_t FLASH_SetCallback(flash_config_t *config, flash_callback_t callback);
```

**Table 27-80. Parameters**

| Parameter | Description |
|---|---|
| Config | Pointer to `flash_config_t` data structure in memory, to store driver runtime state. |
| Callback | A pointer points to a function that is called during erasure.<br>A use for this function is to service the watchdog during an erase operation. |

**Table 27-81. Possible status response**

| Value | Constant | Description |
|---|---|---|
| 4 | kStatus_InvalidArgument | Config or dst pointers are NULL. |
| 115 | kStatus_FLASH_CommandNotSupported | This function is not supported. |
| 0 | kStatus_Success | This function has performed successfully. |

**Example:**

Assume that there is a function.

```
void led_toggle(void).
status_t status = FLASH_SetCallback(&flashInstance, led_toggle);
```

## 27.5 Peripherals Supported

This section describes the peripherals supported by the Kinetis ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in Table 27-3. If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default.

### 27.5.1 LPI2C Peripheral

The Kinetis Bootloader in ROM supports loading data into flash via the LPI2C peripheral, where the LPI2C peripheral serves as the LPI2C slave. A 7-bit slave address is used during the transfer.

Customizing an LPI2C slave address is also supported. This feature is enabled if the Bootloader Configuration Area (BCA) (shown in Table 27-3) is enabled (tag field is filled with 'kcfg') and the i2cSlaveAddress field is filled with a value other than 0xFF. 0x10 is used as the default LPI2C slave address.

The maximum supported LPI2C baud rate depends on corresponding clock configuration field in the BCA. Typical supported baud rate is 400 kbps with factory settings. Actual supported baud rate may be lower or higher than 400 kbps, depending on the actual value of the clockFlags and the clockDivider fields.

Because the LPI2C peripheral serves as an LPI2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected LPI2C slave address and the direction bit is set as write.
- An outgoing packet is read by the host with a selected LPI2C slave address and the direction bit is set as read.
- 0x00 will be sent as the response to host if the target is busy with processing or preparing data.

The following flow charts demonstrate the communication flow of how the host reads ping packet, ACK and response from the target.

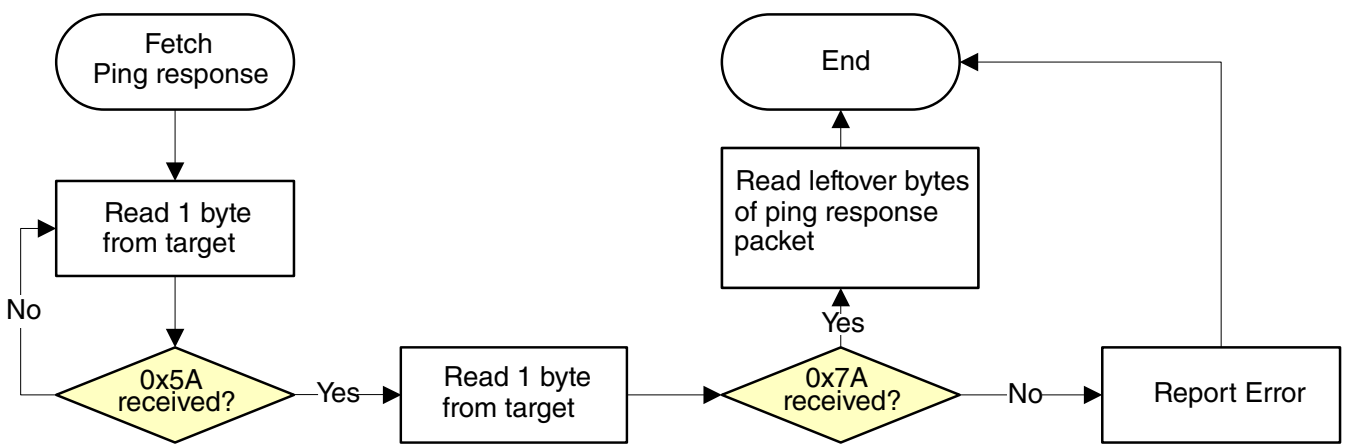



**Figure 27-20. Host reads ping response from target via LPI2C**

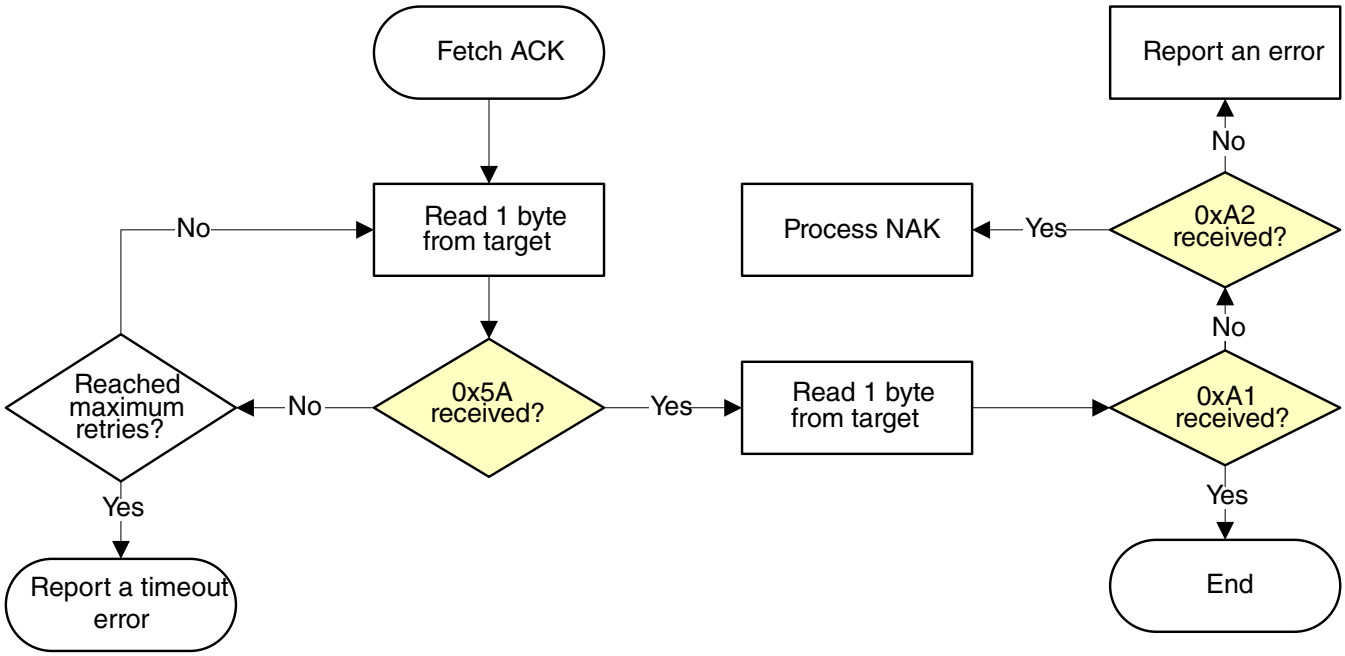



**Figure 27-21. Host reads ACK packet from target via LPI2C**

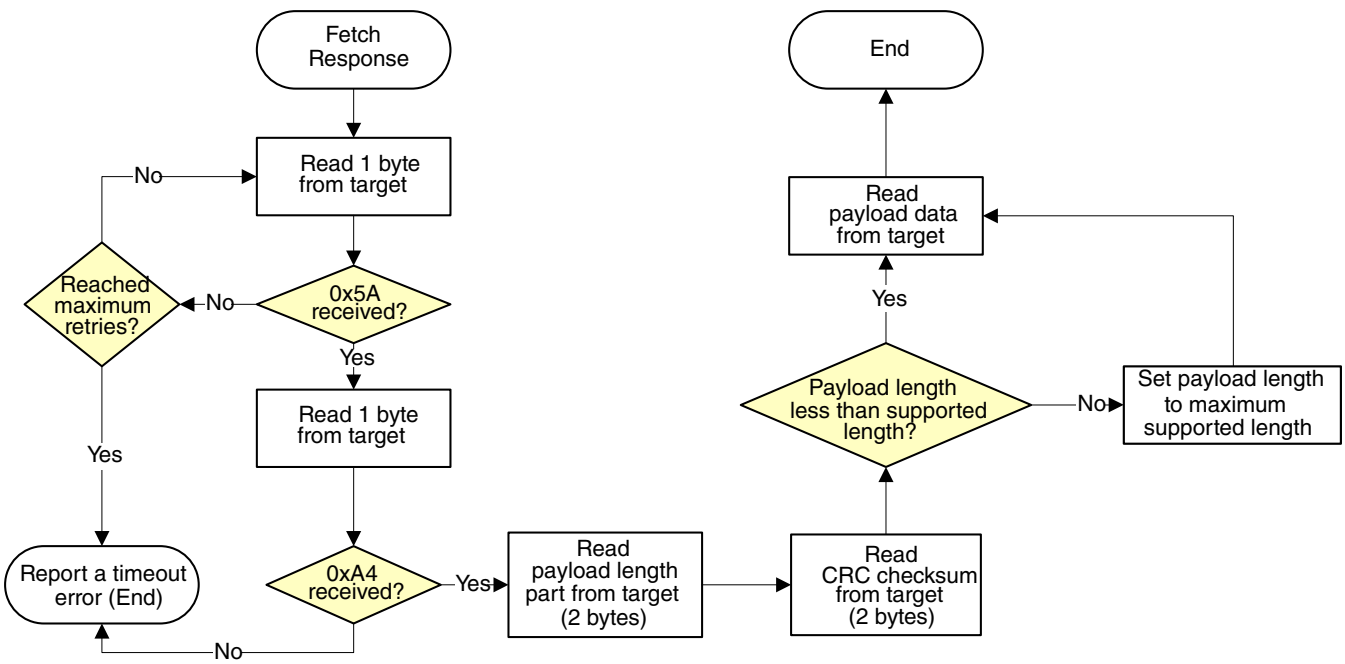



**Figure 27-22. Host reads response from target via LPI2C**

## 27.5.2 LPSPI Peripheral

The Kinetis Bootloader in ROM supports loading data into flash via the LPSPI peripheral, where the LPSPI peripheral serves as a LPSPI slave.

Maximum supported baud rate of LPSPI depends on the clock configuration fields in the Bootloader Configuration Area (BCA) shown in Table 27-3. The typical supported baud rate is 400 kbps with the factory settings. The actual baud rate is lower or higher than 400 kbps, depending on the actual value of the clockFlags and clockDivider fields in the BCA.

The LPSPI peripheral uses the following bus attributes:

- Clock Phase = 1 (Second Edge)
- Clock Polarity = 1 (Active Low)

Because the LPSPI peripheral in ROM serves as a SPI slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

The transfer on LPSPI is slightly different from I2C:

- Host will receive 1 byte after it sends out any byte.
- Received bytes should be ignored when host is sending out bytes to target
- Host starts reading bytes by sending 0x00s to target
- The byte 0x00 will be sent as response to host if target is under the following conditions:
  - Processing incoming packet
  - Preparing outgoing data
  - Received invalid data

The LPSPI bus configuration is:

- Phase = 1; data is sampled on rising edges
- Polarity = 1; idle is high
- MSB is transmitted first

For any transfer where the target does not have actual data to send, the target (slave) is responsible for ensuring that 0x00 bytes will be returned to the host (master). The host uses framing packets to identify real data and not "dummy" 0x00 bytes (which do not have framing packets).

The following flowcharts demonstrate how the host reads a ping response, an ACK and a command response from target via LPSPI.

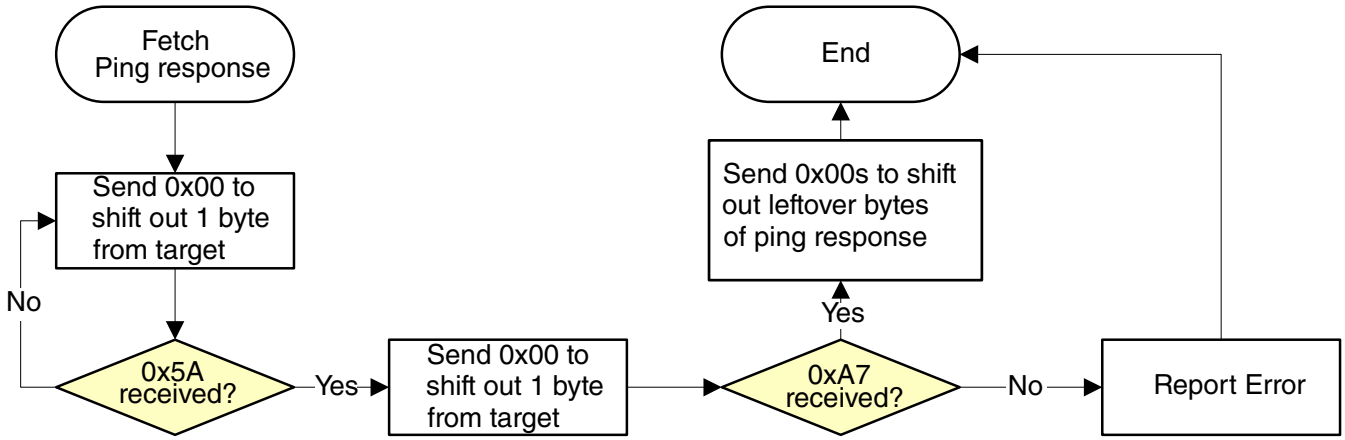



**Figure 27-23. Host reads ping packet from target via LPSPI**

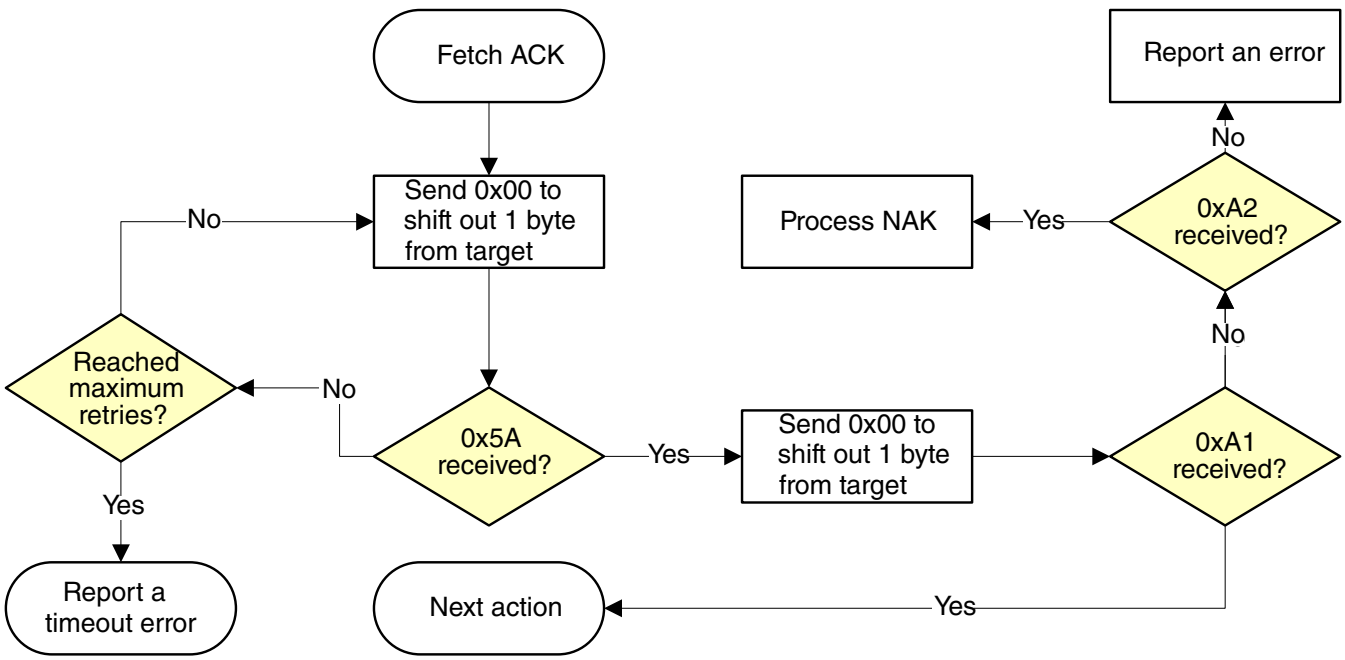



**Figure 27-24. Host reads ACK from target via LPSPI**

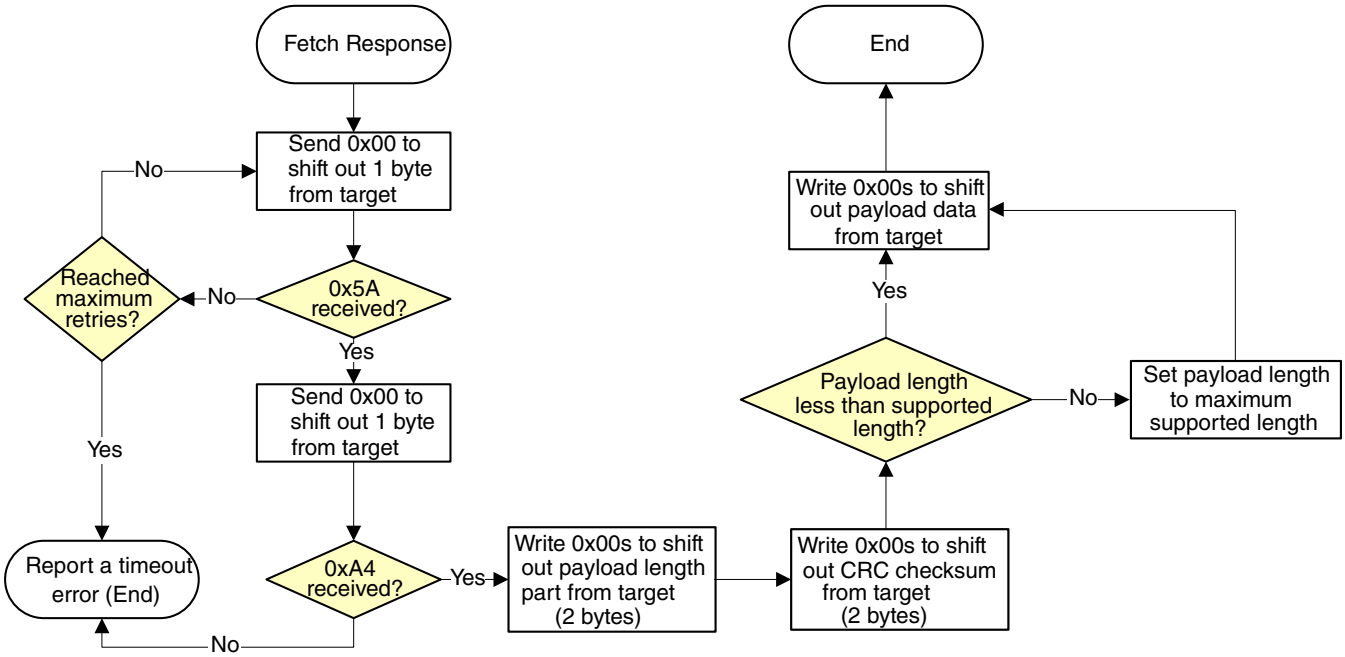



**Figure 27-25. Host reads response from target via LPSPI**

## 27.5.3 LPUART Peripheral

The Kinetis Bootloader integrates an autobaud detection algorithm for the LPUART peripheral, thereby providing flexible baud rate choices.

**Autobaud feature:** If LPUART*n* is used to connect to the bootloader, then the LPUART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the ping packet (0x5A 0xA6) on LPUART*n*_RX, the bootloader firmware executes the autobaud sequence. If the baudrate is successfully detected, then the bootloader will send a ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The Kinetis Bootloader then enters a loop, waiting for bootloader commands via the LPUART peripheral.

**NOTE**

- The autobaud feature requires a ping packet with a higher accuracy (+/-3%), or the ping packet will be ignored as noise.
- The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed LPUART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the

autobaud detection algorithm may calculate an incorrect baud rate. In this case, the autobaud detection state machine should be reset.

**Supported baud rates:** The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, and 115200. Of course, to influence the performance of autobaud detection, the clock configuration in BCA can be changed.

**Packet transfer:** After autobaud detection succeeds, bootloader communications can take place over the LPUART peripheral. The following flow charts show:

- How the host detects an ACK from the target
- How the host detects a ping response from the target
- How the host detects a command response from the target

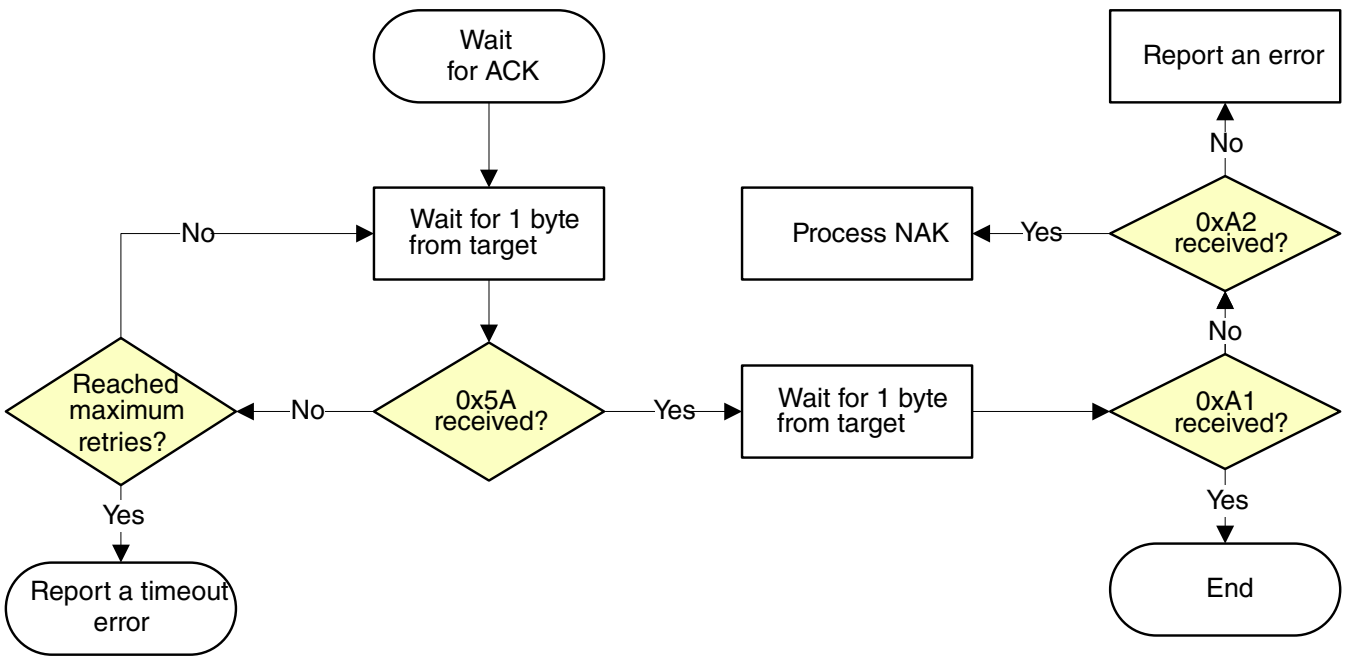



**Figure 27-26. Host reads an ACK from target via LPUART**

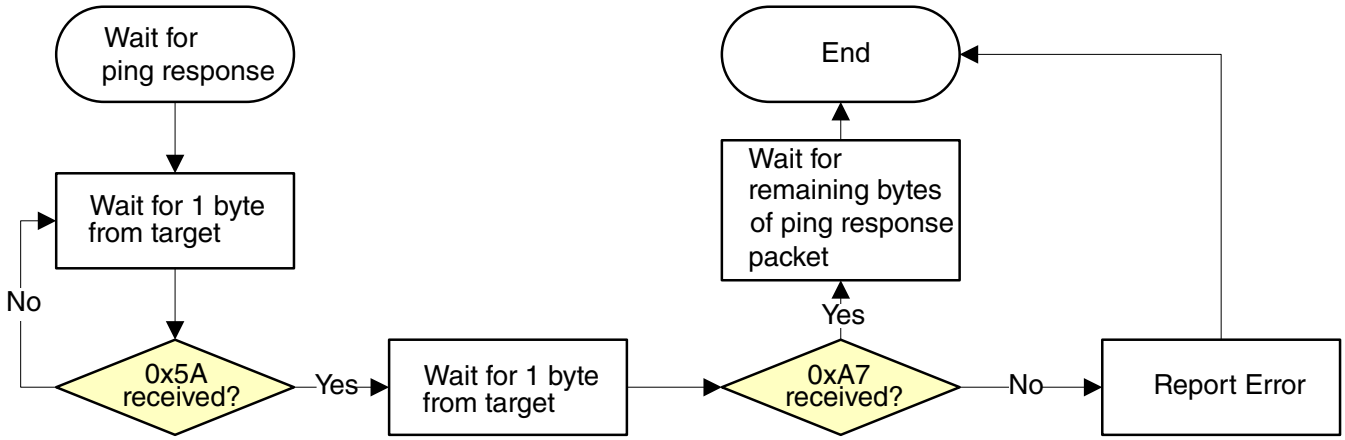



**Figure 27-27. Host reads a ping response from target via LPUART**

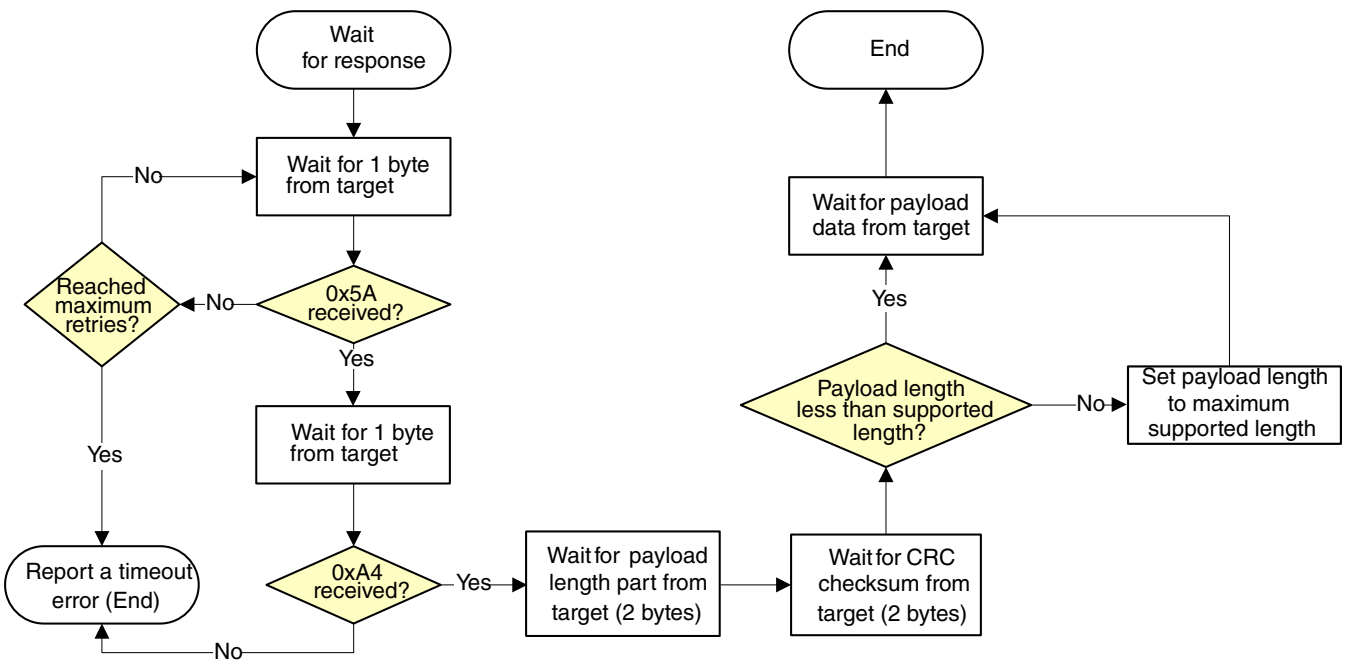



**Figure 27-28. Host reads a command response from target via LPUART**

## 27.5.4 USB peripheral

The Kinetis Bootloader in the ROM supports loading data into flash via the USB peripheral. The target is implemented as a USB HID class.

USB HID does not use framing packets; instead the packetization inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control; the built-in CRC of each USB packet provides the required error detection.

### 27.5.4.1 Clock configuration

The ROM supports the crystal-less USB feature. If the USB peripheral is enabled, then the ROM enables the 48-MHz . The ROM also enables the USB clock recovery feature (by setting USBx_CLK_RECOVER_CTRL[CLOCK_RECOVER_EN] to 1 ).

### 27.5.4.2 Device descriptor

The Kinetis ROM configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x15A2
- PID = 0x0073

Default Strings:

- Manufacturer [1] = "Freescale Semiconductor Inc." (Note that Freescale Semiconductor is now NXP Semiconductors.)
- Product [2] = "Kinetis Bootloader"

You can customize the USB VID/PID/Strings with the Bootloader Configuration Area (BCA) of the flash. See Table 27-3. For example, the USB VID and PID can be customized by writing the new VID to the usbVid(BCA + 0x14) field and the new PID to the usbPid(BCA + 0x16) field of the BCA in flash. To change the USB strings, you need to prepare a structure (like the one shown below) in the flash, and then write the address of the g_languages structure to the usbStringsPointer(BCA + 0x18) field of the BCA.

```
g_languages = { USB_STR_0,
sizeof(USB_STR_0),
(uint_16)0x0409,
(const uint_8 **)g_string_descriptors,
g_string_desc_size};
the USB_STR_0, g_string_descriptors and g_string_desc_size are defined as below.
USB_STR_0[4] =    {0x02,
0x03,
0x09,
0x04
};
g_string_descriptors[4] =
{ USB_STR_0,
USB_STR_1,
USB_STR_2,
USB_STR_3};
g_string_desc_size[4] =
{ sizeof(USB_STR_0),
sizeof(USB_STR_1),
sizeof(USB_STR_2),
sizeof(USB_STR_3)};
```

You can make your own structure of USB_STR_1, USB_STR_2, USB_STR_3:

- USB_STR_1 is used for the manufacturer string.
- USB_STR_2 is used for the product string.
- USB_STR_3 is used for the serial number string.

By default, the 3 strings are defined as below:

```
USB_STR_1[] =
{  sizeof(USB_STR_1),
USB_STRING_DESCRIPTOR,
'F',0,
'r',0,
'e',0,
'e',0,
's',0,
'c',0,
'a',0,
'l',0,
'e',0,
' ',0,
'S',0,
'e',0,
'm',0,
'i',0,
'c',0,
'o',0,
'n',0,
'd',0,
'u',0,
'c',0,
't',0,
'o',0,
'r',0,
' ',0,
'I',0,
'n',0,
'c',0,
'.',0
};

USB_STR_2[] =
{  sizeof(USB_STR_2),
USB_STRING_DESCRIPTOR,
'U',0,
'S',0,
'B',0,
' ',0,
'C',0,
'O',0,
'M',0,
'P',0,
'O',0,
'S',0,
'I',0,
'T',0,
'E',0,
' ',0,
'D',0,
'E',0,
'V',0,
'I',0,
'C',0,
'E',0
};
```

```
USB_STR_3[] =
{  sizeof(USB_STR_3),
USB_STRING_DESCRIPTOR,
'M',0,
'C',0,
'U',0,
' ',0,
'H',0,
'I',0,
'D',0,
' ',0,
'G',0,
'E',0,
'N',0,
'E',0,
'R',0,
'I',0,
'C',0,
' ',0,
'D',0,
'E',0,
'V',0,
'I',0,
'C',0,
'E',0
};
```

### 27.5.4.3 Endpoints

The HID peripheral uses 3 endpoints:

- Control (0)
- Interrupt IN (1)
- Interrupt OUT (2)

The Interrupt OUT endpoint is optional for HID class devices, but the Kinetis Bootloader uses it as a pipe, where the firmware can NAK send requests from the USB host.

### 27.5.4.4 HID reports

There are 4 HID reports defined and used by the bootloader USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

| Report ID | Packet Type | Direction |
|---|---|---|
| 1 | Command | OUT |
| 2 | Data | OUT |
| 3 | Command | IN |
| 4 | Data | IN |

For all reports, these properties apply:

| | |
|---|---|
| Usage Min | 1 |
| Usage Max | 1 |
| Logical Min | 0 |
| Logical Max | 255 |
| Report Size | 8 |
| Report Count | 34 |

Each report has a maximum size of 34 bytes. This is derived from the minimum bootloader packet size of 32 bytes, plus a 2-byte report header that indicates the length (in bytes) of the packet sent in the report.

**NOTE**

In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

| | |
|---|---|
| 0 | Report ID |
| 1 | Packet Length LSB |
| 2 | Packet Length MSB |
| 3 | Packet[0] |
| 4 | Packet[1] |
| 5 | Packet[2] |
| | ... |
| N+3-1 | Packet[N-1] |

This data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length of 35 bytes. The Packet Length header is written in little-endian format, and it is set to the size (in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

## 27.6 Get/SetProperty Command Properties

This section lists the properties of the GetProperty and SetProperty commands.

## Table 27-82. Properties used by Get/SetProperty Commands, sorted by Value

| Property | Writable | Tag Value | Size | Description |
|---|---|---|---|---|
| CurrentVersion | No | 01h | 4 | Current bootloader version. |
| AvailablePeripherals | No | 02h | 4 | The set of peripherals supported on this chip. |
| FlashStartAddress | No | 03h | 4 | Start address of program flash. |
| FlashSizeInBytes | No | 04h | 4 | Size in bytes of program flash. |
| FlashSectorSize | No | 05h | 4 | The size in bytes of one sector of program flash.<br>This is the minimum erase size. |
| FlashBlockCount | No | 06h | 4 | Number of blocks in the flash array. |
| AvailableCommands | No | 07h | 4 | The set of commands supported by the bootloader. |
| VerifyWrites | Yes | 0Ah | 4 | Controls whether the bootloader will verify writes to flash. VerifyWrites feature is enabled by default.<br>0 - No verification is done.<br>1 - Enable verification. |
| MaxPacketSize | No | 0Bh | 4 | Maximum supported packet size for the currently active peripheral interface. |
| ReservedRegions | No | 0Ch | 16 | List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>).<br>• If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs.<br>• If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase. |
| RAMStartAddress | No | 0Eh | 4 | Start address of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains. |
| RAMSizeInBytes | No | 0Fh | 4 | Size in bytes of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains. |
| SystemDeviceId | No | 10h | 4 | Value of the Kinetis System Device Identification register. |
| FlashSecurityState | No | 11h | 4 | Indicates whether Flash security is enabled<br>0 - Flash security is disabled<br>1 - Flash security is enabled |
| UniqueDeviceId | No | 12h | 16 | Unique device identification, value of Kinetis Unique Identification registers (16 for K series devices, 12 for KL series devices) |
| FacSupport | No | 13h | 4 | FAC (Flash Access Control) support flag<br>0 - FAC not supported<br>1 - FAC supported |
| FlashAcessSegmentSize | No | 14h | 4 | The size in bytes of 1 segment of flash |

*Table continues on the next page...*

**Table 27-82. Properties used by Get/SetProperty Commands, sorted by Value (continued)**

| Property | Writable | Tag Value | Size | Description |
|---|---|---|---|---|
| FlashAcessSegmentCount | No | 15h | 4 | FAC segment count (The count of flash access segments within the flash model.) |
| FlashReadMargin | Yes | 16h | 4 | The margin level setting for flash erase and program verify commands.<br>0 = Normal<br>1 = User (default)<br>2 = Factory |
| TargetVersion | No | 18h | 4 | SoC target build version number |

## 27.6.1 Property Definitions

Get/Set property definitions are provided in this section.

### 27.6.1.1 CurrentVersion Property

The value of this property is a 4-byte structure containing the current version of the bootloader.

**Table 27-83. Fields of CurrentVersion property:**

| Bits | [31:24] | [23:16] | [15:8] | [7:0] |
|---|---|---|---|---|
| Field | Name = 'K' (0x4B) | Major version | Minor version | Bugfix version |

### 27.6.1.2 AvailablePeripherals Property

The value of this property is a bitfield that lists the peripherals supported by the bootloader and the hardware on which it is running.

**Table 27-84. Peripheral bits:**

| Bit | [31:7] | [6] | [5] | [4] | [3] | [2] | [1] | [0] |
|---|---|---|---|---|---|---|---|---|
| Peripheral | Reserved | USB DFU | USB CDC | USB HID | Reserved | LPSPI Slave | LPI2C Slave | LPUART |

If the peripheral is available, then the corresponding bit will be set in the property value. All reserved bits must be set to 0.

### 27.6.1.3 AvailableCommands Property

This property value is a bitfield with set bits indicating the commands enabled in the bootloader. Only commands that can be sent from the host to the target are listed in the bitfield. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. 1 is subtracted from the command tag because the lowest command tag value is 0x01. To get the bit mask for a given command, use this expression:

```
mask = 1 << (tag - 1)
```

**Table 27-85. Command bits:**

| Bit | [31:18] | [17] | [16] | [15] | [14] | [13] | [12] | [11] | [10] | [9] | [8] | [7] | [6] | [5] | [4] | [3] | [2] | [1] | [0] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Command | Reserved | ReliableUpdate | Reserved | FlashReadResource | FlashReadOnce | FlashProgramOnce | FlashEraseAllUnsecure | SetProperty | Reset | Call | Execute | ReceiveSBFile | GetProperty | FlashSecurityDisable | FillMemory | WriteMemory | ReadMemory | FlashEraseRegion | FlashEraseAll |

### 27.6.1.4 TargetVersion Property

The value of this property is a 4-byte structure containing the target version of the bootloader. The TargetVersion number returned returned for this device is 1.0.0.

**Table 27-86. Fields of TargetVersion property:**

| Bits | [31:24] | [23:16] | [15:8] | [7:0] |
|---|---|---|---|---|
| Field | Name = 'T' (0x4B) | Major version | Minor version | Bugfix version |

## 27.7 ROM API

The ROM API allows applications to integrate ROM bootloading services with Over the Air (OTA) update, where a firmware update image is downloaded into external memory (SPI flash) and authenticated before being written into internal flash. The ROM API is also used to authenticate regions of internal flash before enabling or jumping to that code, thus forming a chain of trust. This internal flash code may be a bootloader authenticating the main application image, or it may be a boot core authenticating the code for a secondary core before releasing it from reset.

### 27.7.1 ROM API

An application calls into the ROM API to provision and authenticate an SB file image. There are 3 function calls:

1. Call kb_init() to start an API session.
2. Call kb_execute() for image loading/authentication.
3. Call kb_deinit() to clean up API session.

Also see Table 27-96.

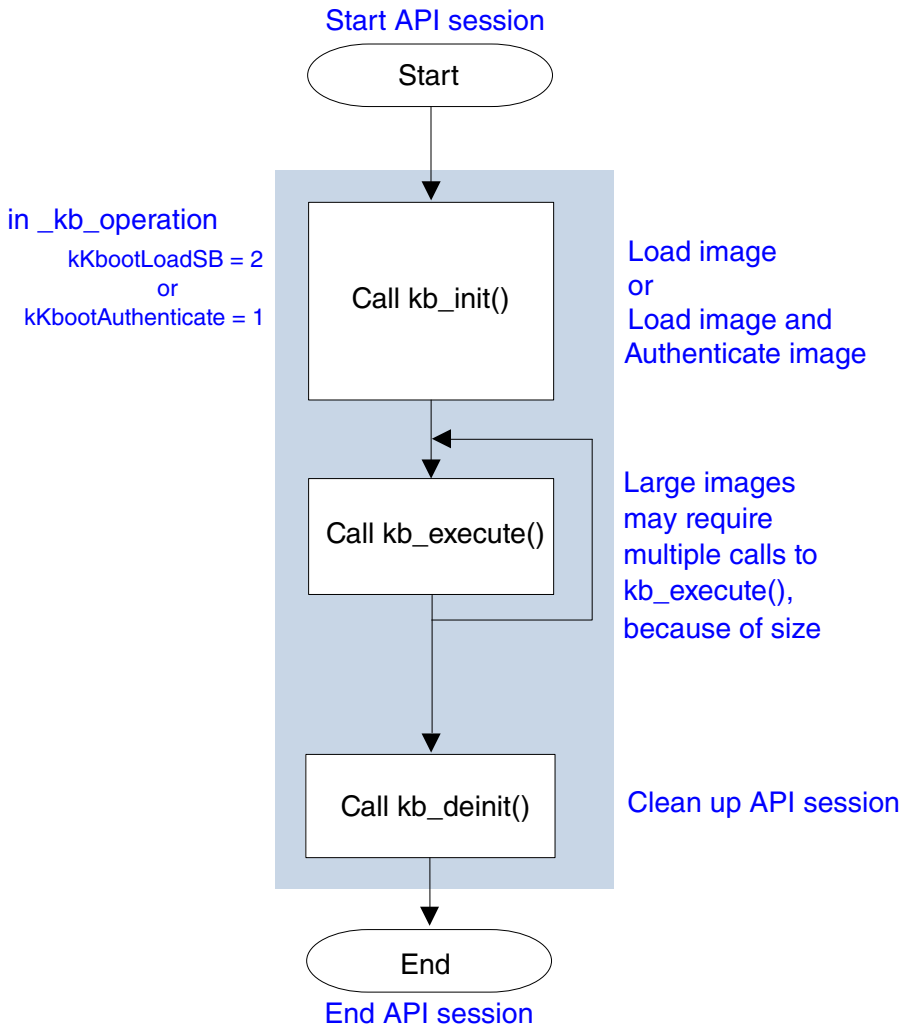



**Figure 27-29. ROM API Sessions**

## 27.7.2 ROM API RAM usage

The kb_session_ref_t structure is allocated by ROM. The initialization function kb_init() returns the kb_session_ref_t structure, and kb_session_ref_t is required to be passed in by other ROM API functions.

The kb_session_ref_t structure points to a fixed block of on-chip RAM for much of its memory use.

- After kb_init() is called, the application is not allowed to touch that fixed block of on-chip RAM (for-ROM-use-only) memory.

- After kb_deinit() is called, the application regains control of this fixed block of on-chip RAM memory, and thereafter the application can use it.
- The ROM will also make use of a buffer that was passed in via the _kb_options structure for temporary memory, thus reducing the overall size of the reserved memory. This is the easiest implementation but imposes restrictions on the application.

The kb_options_t structure provides all of the needed options to carry out the provisioning and authenticating tasks. Some of the structure members could be the functions to read from the external data source that the application is providing, using a pointer to the pre-authenticated function code available in the internal flash (or elsewhere).

## 27.8 Security

### 27.8.1 Introduction

This section defines the Kboot security architecture.

The Kboot security architecture provides 3 basic operations:

- **Signed image authentication:** which verifies RSA signatures over the entire image, using RSA public keys contained in the signed image.
- **Secure boot:** which provides the guarantee that unauthorized code cannot be executed on a given device.
- **Secure firmware update through SB files:** which authenticates the new firmware before committing the firmware to memory.

This security architecture is based on 2 image formats:

- **A format for signed images**. The signed image format is common and may contain executable code resident in internal flash (or other memories), or may even contain a firmware update image.
- **A format for firmware updates**. The firmware update image format is called Secure Binary (SB file). The SB format is signed using the common signed image format, and SB files are always encrypted.

The KBoot provides an API that integrates security architecture operations into customer applications.

## 27.8.2 Notation

| Operator | Definition |
|---|---|
| AES(K, x) | AES-256 encipherment algorithm applied to octet string *x* using key *K*. |
| BE(x) | Integer *x* represented as a big endian octet string with a length of the smallest number of octets required to properly represent *x*. |
| ceil(x) | Returns the smallest integer greater than or equal to a given number. |
| floor(x) | Returns the largest integer less than or equal to a given number. |
| HMAC(K, x) | HMAC-SHA256 algorithm applied to octet string *x* using key *K*. |
| Len(x) | Length in octets of *x*. |
| x MOD y | Integer remainder of *x* divided by *y*. |
| SHA256(x) | SHA-256 hash algorithm applied to octet string *x*. |
| a XOR b | Exclusive-OR between octet strings *a* and *b*. |
| a \|\| b | Concatentation of octet strings *a* and *b*. |
| [ hex-byte$_0$ … hex-byte$_{N-1}$ ] | Octet string of length N. Example: [ 55 10 04 a7 ]. |
| S ^ n | The octet string *S* repeated *n* times, forming a result of (Len(S) * n) octets. |
| x..y | An inclusive range of integers from *x* through *y*. |

| Symbol | Definition |
|---|---|
| DigestInfo | DER encdoded ASN.1 value of type DigestInfo. |
| EM | Encoded data to be signed with an RSA private key in RSASSA-PKCS1_v1_5. |
| H | SHA-256 hash of signed image. |
| HmacRangeCount | Number of HMAC ranges for a given section. |
| HmacRangeLen$_j$ | Number of cipher blocks for section HMAC *j* for a given section. |
| IPK | Public key contained in the image signing certificate. |
| k | Length in octets of the IPK. |
| K DEK | AES-256 data encryption key used to encrypt SB file sections. |
| K MAC | 256-bit HMAC key used to authenticate SB file header. |
| Offset$_x$ | Offset of region *x*. |
| PS | Padding used in *EM*. |
| RKH$_x$ | Hash of root public key number x. |
| RKTH | Root key table hash. |
| RPK | Public key contained in the root certificate. |
| SB$_x$ | Cipher block number *x* of an SB file. |

**NOTE**

Unless explicitly noted, all multi-byte values in the definitions below are to be stored in little endian format.

### 27.8.3 Use cases

All of these use cases are about establishing a trust relationship between pieces of code that run on the target device. The device's ROM is the trust root.

#### 27.8.3.1 Secure boot

Device manufacturers want to ensure that unauthorized images (code) cannot be executed on a given product. Secure boot provides this guarantee, which involves the device's ROM always executing when coming out of reset. When secure boot is enabled, to determine the authenticity of that image, the ROM bootloader will then examine the 1st user executable image resident in internal flash memory. If the image is authentic, then control is transferred to that image; this establishes a chain of trusted code from the ROM bootloader to the user boot image. This chain can be further extended, as described next.

In this architecture, the method used to verify the authenticity of the user boot image is to verify RSA signatures over the entire user boot image. The user boot image is signed with RSA private keys. The corresponding RSA public keys used for signature verification are contained in X.509 certificates that are contained in the signed user boot image.

#### 27.8.3.2 Secure firmware updates

If firmware updates are to be performed in the field when secure boot is enabled, then a secure firmware update mechanism is preferred; otherwise, inauthentic (not authentic) firmware may be written to the device, and may possibly prevent the device from booting. In the most basic sense, a secure firmware update simply performs an authentication of the new firmware before committing it (the new firmware) to memory. In this case, the chain of trust is extended from the old, currently executing code to the new code.

Another use case for a secure firmware update is to hide the application binary code during transit over public media (like the Internet). This is accomplished by encrypting the firmware update image. As the new firmware is written into device memory, it (the new firmware) is decrypted.

In this architecture, both cases of secure firmware update are supported. The SB file format is always encrypted and also may be signed. If the entire SB file is available before loading, then the ROM API may be used to authenticate it (the SB file). As an SB

file is loaded, it is decrypted. The ROM may support loading of SB files via serial interfaces such as USB, UART, and others. If an SB file is loaded in this way (over a serial interface), then authentication is not supported

#### 27.8.3.3 Extending the chain of trust

After secure boot has transferred CPU control to user code, then that user code may need to load additional pieces of code. This establishes another link in the chain of trust. The process can continue for as many nested submodules are required, with each parent code module authenticating the child (submodule). Another use case is to authenticate boot code for one or more secondary CPU cores, before releasing them (the secondary cores) from reset.

The ROM API is used from customer code to verify signatures on the additional code images. Using the ROM API to verify signatures gives complete control (which additional code must be signed and how that code is organized in memory) to the customer code.

### 27.8.4 Security fields

#### 27.8.4.1 Keys

Keys are in 2 locations:

- In IFR fields
- In ROM API

Both sets of keys provide the same "key" function, but keys in the ROM API struct are only used by ROM API loading image/authentication session. Each ROM API session must provide its own keys. After calling kb_deinit():

- Keys provided by kb_init are not available any more.
- Keys in IFR fields will be still available (unless calling flash mass erase).

There are 3 keys:

- The 1st key is an AES-256 key encryption key (KEK) called the SBKEK. The SBKEK key is used only for unwrapping additional keys stored in an SB file.

- The 2nd "key" is actually not a key, but is a SHA-256 hash of a table of hashes of RSA public keys. This hash is called the root key table hash (RKTH). RKTH is used to lock the public key in the root certificate against change.
- The 3rd key (USERKEK) is a separate AES-256 key held in IFR. This key is designated for customer use as a KEK and is not used by the loader. To distinguish the KEK key from the SBKEK key, this KEK key is called the USERKEK. USERKEK is programmable once and erasable via mass erase, but cannot be read by software. There is a hardware bus that feeds the USERKEK key directly into the crypto-accelerator.

How the keys are used:

- For secure boot, RKTH are values in the IFR index 0x94 - 0x97.
- Through the command line, SBKEK are values in the IFR index 0x90 - 0x93.
- Through the ROM API, SBKEK / RKTH can also be provided by the user in kb_options_t. If a key value is zero, then the corresponding key in IFR will be used instead.
- For a user-defined situation, USERKEY in the IFR index can be used.

**Table 27-87. Keys**

| Key | Indices | Size (bytes) | Name | Description |
|---|---|---|---|---|
| 1st | 0x90 - 0x93 | 32 | SBKEK[1] | AES-256 key for SB key unwrapping. |
| 2nd | 0x94 - 0x97 | 32 | RKTH[2] | SHA-256 hash of root public key hash table. |
| 3rd | 0xB1 - 0xB4 | 32 | USERKEK | Customer write-only AES-256 key. |

1. If SBKEK has a value of 0xFFFF FFFF FFFF FFFF, then SBKEK will be processed as if it had a value of 0.
2. To generate the RKTH, a SHA-256 is computed over the root public key hash table (RKH table).

### 27.8.4.2 Flags

Several critical flags used for secure boot (and other purposes) are placed in write-once flash IFR (or OTP).

**Table 27-88. Flag words in IFR**

| Index | Bytes | Name | Description |
|---|---|---|---|
| 0x98 | 8 | ROM_SECURITY_FLAGS | ROM security settings |
| 0x99 | 8 | ROM_SECURITY_PROFILE | Security profile |
| 0x9C | 8 | ROM_REVOKE_RK0 | Root key #0 revocation flag |

*Table continues on the next page...*

**Table 27-88. Flag words in IFR (continued)**

| Index | Bytes | Name | Description |
|---|---|---|---|
| 0x9D | 8 | ROM_REVOKE_RK1 | Root key #1 revocation flag |
| 0x9E | 8 | ROM_REVOKE_RK2 | Root key #2 revocation flag |
| 0x9F | 8 | ROM_REVOKE_RK3 | Root key #3 revocation flag |

**Table 27-89. Flag word bit/field definitions**

| Index | Bit(s) | Name | Description |
|---|---|---|---|
| 0x98 | 1:0 | SECURE_BOOT_EN | 2'b00: Secure boot is disabled.<br>2'b10: Secure boot is enabled.<br>2'b01: Secure boot is enabled.<br>2'b11: Secure boot is disabled. |
| 0x98 | 3:2 | SECURE_BOOT_DEV_EN | 2'b00: Secure boot development mode is disabled.<br>2'b10: Secure boot development mode is enabled.<br>2'b01: Secure boot development mode is enabled.<br>2'b11: Secure boot development mode is disabled. |
| 0x98 | 4 | STOP_ON_FAILURE | If secure boot authentication fails:<br>• And if STOP_ON_FAILURE is set, then the ROM will enter an infinite loop.<br>• And if STOP_ON_FAILURE is not set, then the ROM will enter bootloader mode. |
| 0x99 | 16 | RSA4096_KEY | 4096-bit RSA keys required. |
| 0x9C | 63:0 | ROM_REVOKE_RK0 | Any bit being cleared causes root key table entry #0 to be revoked. |
| 0x9D | 63:0 | ROM_REVOKE_RK1 | Any bit being cleared causes root key table entry #1 to be revoked. |
| 0x9E | 63:0 | ROM_REVOKE_RK2 | Any bit being cleared causes root key table entry #2 to be revoked. |
| 0x9F | 63:0 | ROM_REVOKE_RK3 | Any bit being cleared causes root key table entry #3 to be revoked. |

The ROM_SECURITY_PROFILE at index 0x99 matches the bit masks defined by the _kb_security_profile enumeration in the ROM API.

**NOTE**

If ROM_SECURITY_PROFILE has a value of 0xFFFF FFFF FFFF FFFF, then it will processed as if it had a value of 0.

**Table 27-90. Additional IFR fields that affect secure boot and the ROM**

| Index | Bytes | Name | Description |
|---|---|---|---|
| 0x80 | 2 | SEC0 | Flash security - Primary |

*Table continues on the next page...*

**Table 27-90. Additional IFR fields that affect secure boot and the ROM (continued)**

| Index | Bytes | Name | Description |
|---|---|---|---|
| 0x81 | 2 | FSLACC | Flash security - Factory Security Level Access |
| 0x82 | 2 | MEEN | Flash security - Mass Erase Enable |
| 0x83 | 2 | KEYEN | Flash security - Backdoor Key Enable |
| 0x84 | 4 | FOPT | Used to customize MCU boot time configuration |
| 0xA0 | 2 | SEC1 | Security Qualifier |

For FOPT field definitions, see the Reset and Boot chapters.

## 27.8.5 Signed image structure

There is a single format used for signed data, whether the signed data is an SB file or is code located in internal flash. Using a single format for signed data allows the same API and code to be used for both

- authenticating a signed SB file,
- and authenticating a signed image in flash.

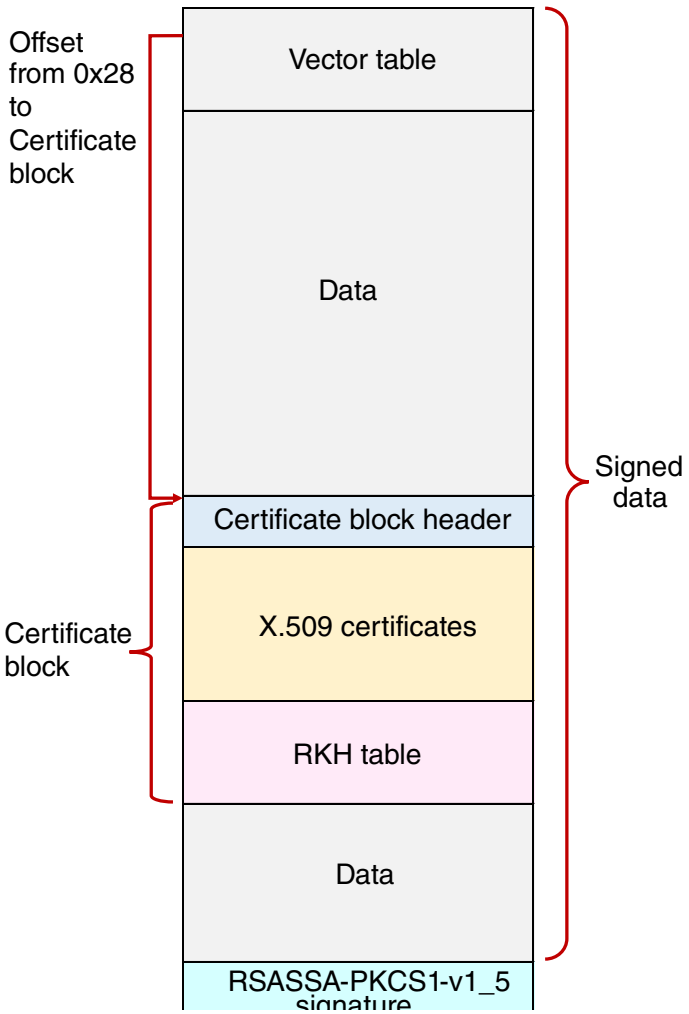



**Figure 27-30. Signed image structure**

In this diagram, data is shown both before and after the certificate block. This is optional; the certificate block may appear anywhere within the signed data.

- Images are signed using the RSASSA-PKCS1-v1_5 algorithm.
- The digest is computed using SHA-256.
- 2048-bit or 4096-bit RSA keys are supported.

### 27.8.5.1 Certificate block offset

A 32-bit offset called offsetToCertificateBlockInBytes, *which goes from the beginning of the signed image to the certificate block*, must reside at offset 0x28 from the start of the signed image. An executable code image in internal flash is expected to start with an NVIC vector table. The word at offset 0x28 is a reserved slot in the vector table. For SB files, offset 0x28 is reserved to hold offsetToCertificateBlockInBytes in the file header. If

signed code in flash does not begin with a vector table (such as for a signed library or similar secondary image), then the word at offset 0x28 must be reserved for the offsetToCertificateBlockInBytes field.

For example, if an image resides in flash at a non-zero address (say 0x8000), and its certificate block header is at address 0x24000, then the word at 0x8028 will contain the value 0x1C000.

Here is a standard Cortex-M4 NVIC vector table with the offset to the certificate block highlighted at offset 0x28.

| Offset | Used for |
|---|---|
| 0x00 | Initial SP |
| 0x04 | Reset |
| 0x08 | NMI |
| 0x0C | HardFault |
| 0x10 | MemManage |
| 0x14 | BusFault |
| 0x18 | UsageFault |
| 0x1C | *Reserved* |
| 0x20 | *Reserved* |
| 0x24 | *Reserved* |
| **0x28** | **offsetToCertificateBlockInBytes** |
| 0x2C | SVC |
| 0x30 | DebugMon |
| 0x34 | *Reserved* |
| 0x38 | SysTick |

### 27.8.5.2 Certificate block

The certificate block consists of the certificate block header, the certificate table, and the RKH table, concatenated together.

- The certificate block is allowed to reside anywhere within the signed image, but the certificate block must be fully contained within the signed data, such that the certificate block itself is signed.
- The certificate block offset is pointed to by the offsetToCertificateBlockInBytes.
- The most common constructions will have the certificate block placed at either the beginning of the signed data (after the vector table) or at the end of the signed data.

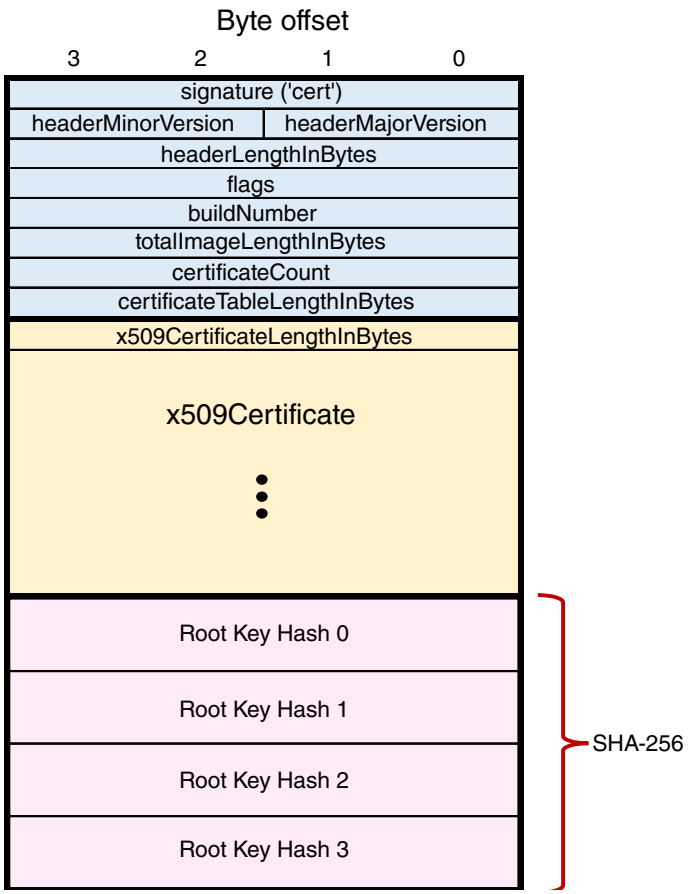



**Figure 27-31. Certificate block structure**

### 27.8.5.2.1 Certificate block header

The certificate block header is a structure containing the information required to properly verify a signed image. As described previously, it is pointed to by the offsetToCertificateBlockInBytes header offset field.

Let $Offset_{header}$ = (offsetToCertificateBlockInBytes)

The first word of the certificate block header must be 4-byte aligned.

Structure definition for the certificate block header:

```
struct certificate_block_header_t {
    uint32_t signature;
    uint16_t headerMajorVersion;
    uint16_t headerMinorVersion;
    uint32_t headerLengthInBytes;
    uint32_t flags;
    uint32_t buildNumber;
    uint32_t totalImageLengthInBytes;
    uint32_t certificateCount;
    uint32_t certificateTableLengthInBytes;
};
```

**Table 27-91. Certificate block header fields**

| Field | Description |
|---|---|
| signature | Always set to 'cert'. |
| headerMajorVersion | Set to 1. |
| headerMinorVersion | Set to 0. |
| headerLengthInBytes | Number of bytes that the header has, starting from the signature. headerLengthInBytes does not include the certificate table. |
| flags | Reserved for future use. |
| buildNumber | User-specified build number for the signed image. Allows the user to prevent reverting to old versions. The API compares the buildNumber against the minBuildNumber specified in kb_options_t. buildNumber must be a multiple of 4. |
| totalImageLengthInBytes | Length in bytes of the signed data |
| certificateCount | Must be greater than 0. |
| certificateTableLengthInBytes | Total length in bytes of the certificate table. certificateTableLengthInBytes must be a multiple of 4. |

The key field in the certificate header is totalImageLengthInBytes, which indicates the number of bytes of signed data, starting at offset 0 of the image. The entire certificate block must be contained within the signed data.

The signature field can be treated as 4-character string, without a terminating null byte, with a value of 'cert'. Represented as a 32-bit little endian constant, the value would be (('c') | ('e' << 8) | ('r' << 16) | ('t' << 24)).

### 27.8.5.2.2 Certificate table

Immediately following the certificate block header is the certificate table. It consists of a complete chain of one or more X.509 certificates, with each certificate prefixed with a length word.

$$\text{Let } Offset_{\text{certificate table}} = (\text{offsetToCertificateBlockInBytes} + \text{headerLengthInBytes})$$

Structure definition for the certificate table:

```
struct cert_entry {
    uint32_t x509CertificateLengthInBytes;
    uint8_t x509Certificate[];
};

struct cert_table {
    cert_entry entries[];
};
```

- The x509CertificateLengthInBytes field for each certificate must be set to the length of that certificate's data in bytes, rounded up to the next word (4-byte) alignment; therefore, x509CertificateLengthInBytes must be divisible by 4.

- Each x509Certificate contains the actual certificate data, and can be of variable length. There may be from 0-3 bytes of padding inserted after the certificate data.
- The cert_entry struct is repeated for certificateCount entries in the table.
- The total number of bytes occupied by the table must equal certificateTableLengthInBytes, and must always be divisible by 4.

There are restrictions on the certificates:

- Only X.509 v3 format certificates are supported.
- Must be DER-encoded.
- Must use RSA-2048 or RSA-4096 and SHA-256.
- All certificates must use RSA keys with a modulus bitlength greater than or equal to the RSA bitlength specified by the security profile. For example, a 4096-bit root key followed by a 2048-bit image key is allowed, if the security profile is set to 2048-bit keys.
- The SHA-256 hash of the public key contained in the first certificate in the table must be present in the RKH table.

The certificate table can contain one or more certificates:

- Certificates must be positioned in the table starting with the root certificate, followed by each subsequent certificate in the chain, in the order of signing. The final certificate in the table is called the image-signing certificate.
- Using a single certificate is allowed; in this case, the sole certificate must be self-signed and must not be a CA.
- If multiple certificates are used, then the root must be self-signed and all but the last certificate must be CAs.

The 2 most common configurations will be:

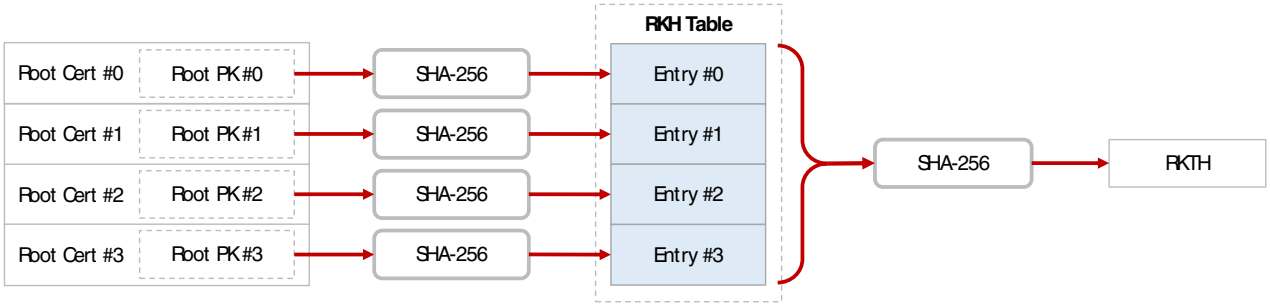
- One self-signed certificate.
- A self-signed root CA certificate, followed by an image signing certificate (which is itself signed by a root certificate).

#### 27.8.5.2.3 RKH table

Following the certificate table is a table containing SHA-256 hashes of up to 4 allowed root public keys. Multiple root public keys are supported to allow for key revocation.

```
Let Offset_RKHT = (offsetToCertificateBlockInBytes
                                          + headerLengthInBytes
                                          + certificateTableLengthInBytes)
```

The structure of this table is shown here:




The C definition of the table:

```
struct rkh_entry {
    uint8_t rkh[32];
};

struct rkh_table {
    rkh_entry entries[4];
};
```

Each entry in the table is a SHA-256 computed over the concatenation of an RSA public key's modulus and exponent (modulus || exponent). Both modulus and exponent must be in big endian byte order, with the minimum number of bytes required to represent the value. For instance, an exponent of 65537 would be represented by a 3-byte value of [01 00 01], while an exponent of 3 would be represented by a single byte of that value. The entire RKH table is itself hashed with SHA-256. This final hash is then stored in the RKTH field in IFR.

```
For i in 0..3:
                Let M_i = BE(Modulus_i)
                Let E_i = BE(Exponent_i)
                Let RKH_i = SHA256( M_i || E_i)
                Let RKTH = SHA256( RKH_0 || RKH_1 || RKH_2 || RKH_3 )
```

The number of hashes of keys in the RKH table must be from at least 1 through a maximum of 4. Unused table entries must be set to all 0 bytes.

The extra root public keys and root certificates must be created in advance and would be held in reserve in case a public key had to be revoked. The customer is responsible for implementing the mechanism to determine whether a key needs to be revoked, and then set the appropriate RK_REVOKE bit(s). This would usually be done through an authenticated connection with a server during a firmware update.

Note that only one of the root certificates whose keys are listed in the RKH table may be included in the certificate table at a time.

### 27.8.5.3 Signature

The signature itself is placed at offset totalImageLengthInBytes from the beginning of the image and must be in RSASSA-PKCS1-v1_5 format.

Let $Offset_{sig}$ = (totalImageLengthInBytes)

The first word of the signature must be 4-byte aligned.

A SHA-256 hash is computed over the entire image data from offset 0 through totalImageLengthInBytes. The resulting hash is then signed using RSASSA-PKCS1-v1_5 format with the RSA private key corresponding to the public key contained in the final certificate in the certificate table. See PKCS#1 for the complete details of RSASSA-PKCS1-v1_5.

For RSA-2048, the signature will be 256 bytes; for RSA-4096, it will be 512 bytes.

**NOTE**

DigestInfo is an encoding of the ASN.1 DigestInfo value with the following ASN.1 syntax. The fixed octet string used for the DigestInfo value in the algorithm above is for an AlgorithmIdentifier of SHA-256.

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm AlgorithmIdentifier,
    digest OCTET STRING
}
```

## 27.8.6 SB file formats

The Secure Binary (SB) image format is a command-based firmware update image. The SB image format has a long history, and has been used on multiple STMP, i.MX, and Kinetis devices.

In addition to version 1.0 of SB image format, this section also defines the new version 2.0 of the SB image format. Version 2.0 updates the encryption scheme to use modern algorithms, and also adds support for signed images. SB format version 2.0 format only supports 256-bit AES keys.

The SB format is composed of 16-byte cipher blocks as its basic unit. All regions of the file are aligned to 16 bytes. The maximum size of an SB file is $2^{31}$ - 1 cipher blocks, or $2^{39}$ - 128 bytes.

### 27.8.6.1 Regions

The layout of an SB 2.0 file is shown next. The header fields with a brighter blue background are the differences from the SB 1.x file and header layout.

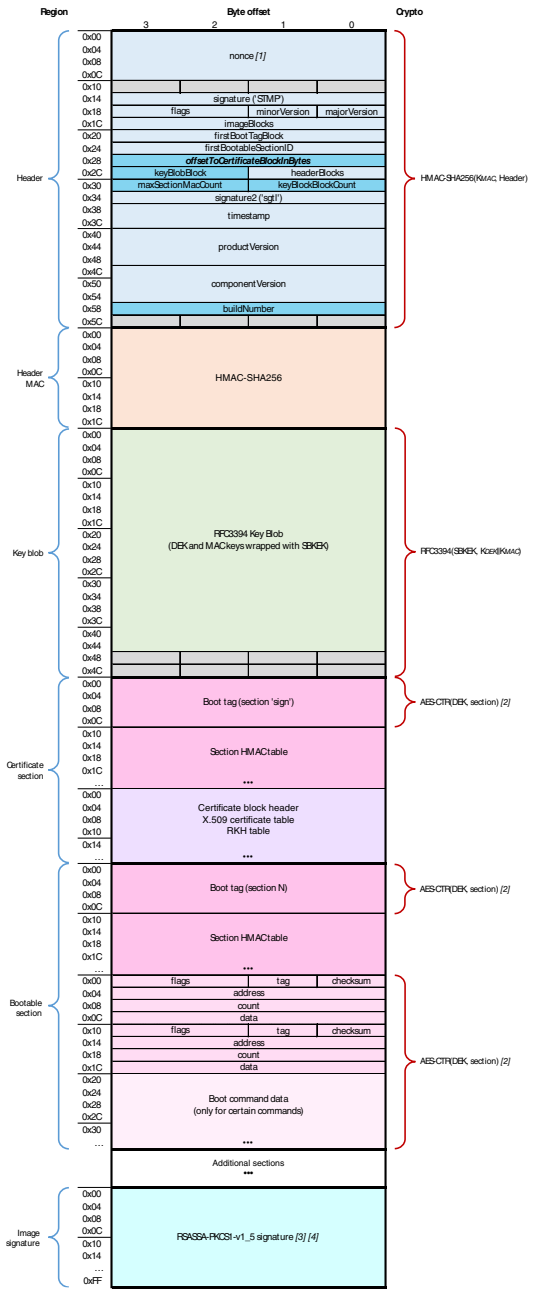



**Figure 27-32. SB format version 2.0 layout**

[1] 128-bit random nonce for AES-CTR with bit 31 cleared.

[2] AES-CTR uses nonce in header + cipher block offset in file for counter.

[3] Signature must be at offset of certificate_block_header_t::totalImageLengthInBytes and sign all data from 0 to signature.

[4] Signature will be either 256 or 512 bytes in length, depending on the RSA key size.

For your online reading convenience, SB format version 2.0 is shown next in 3 parts: top, middle, bottom.

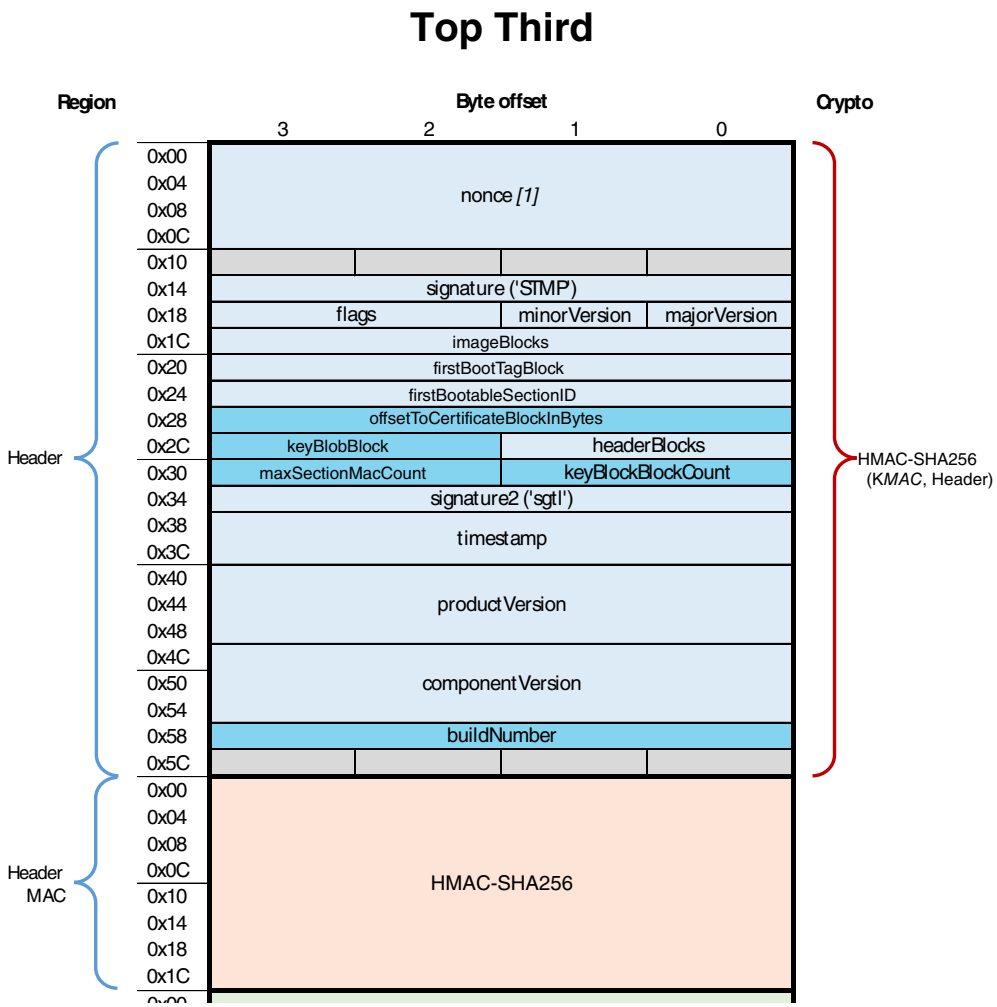



**Figure 27-33. SB format version 2.0 layout: top**

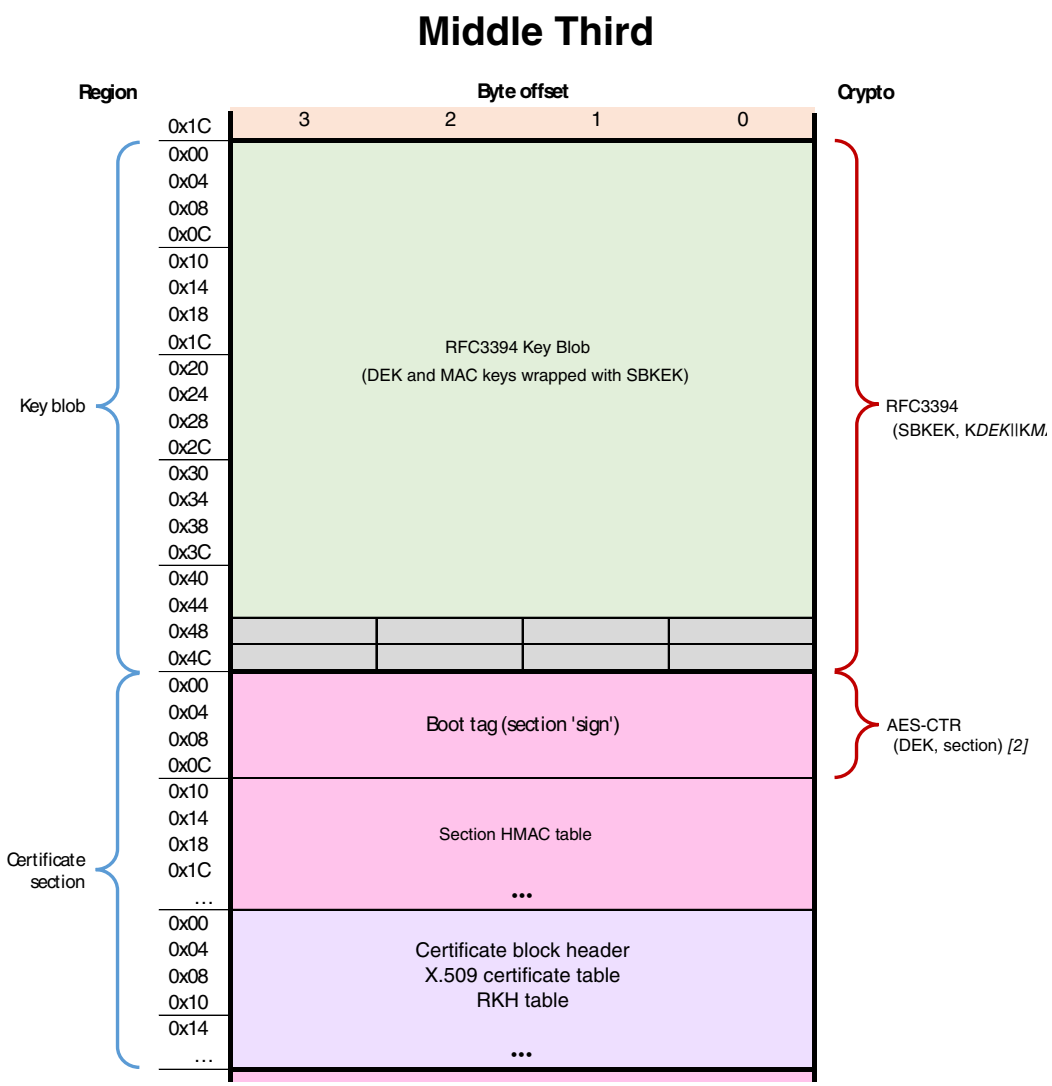



**Figure 27-34. SB format version 2.0 layout: middle**

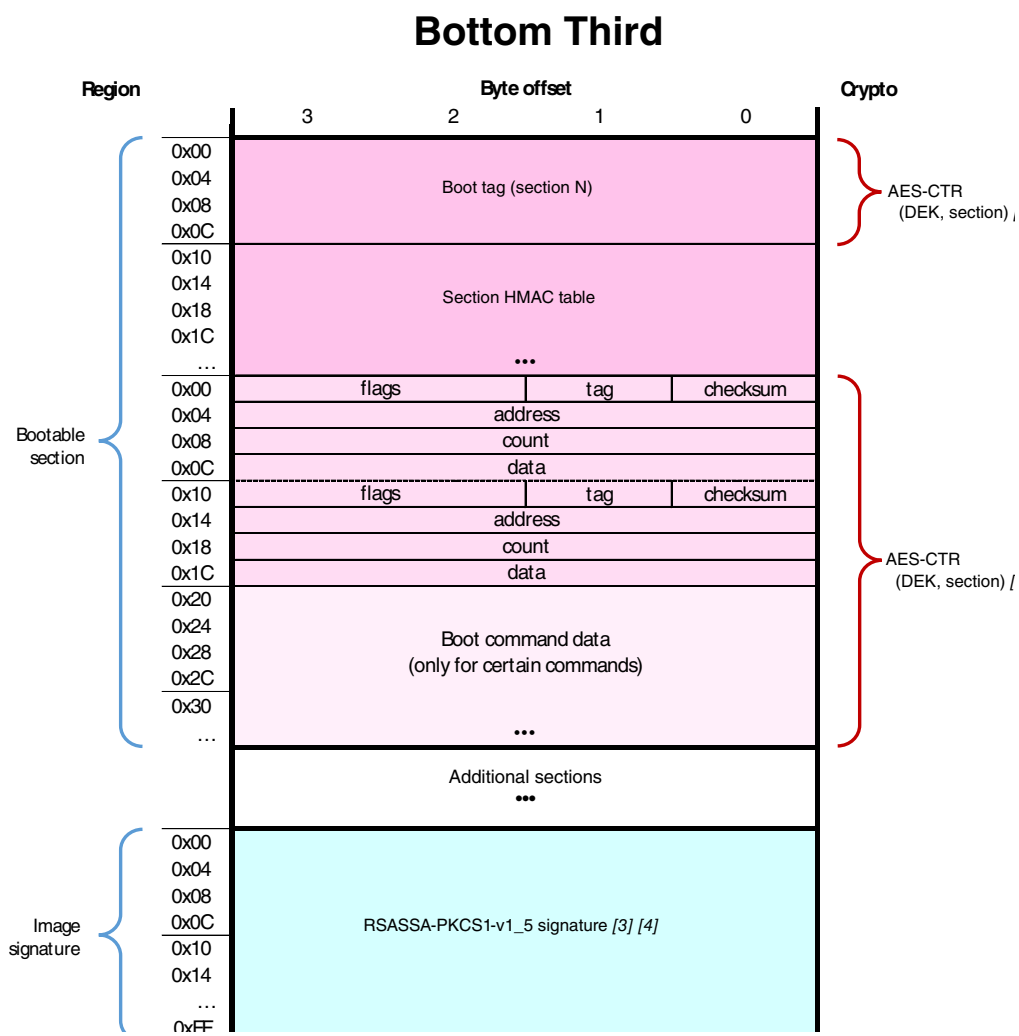



**Figure 27-35. SB format version 2.0 layout: bottom**

## 27.8.6.1.1 Header

Many of the SB 2.0 header fields are in the same position as the SB 1.0 format, including the magic numbers and format version. Other header fields were replaced.

```
struct version_t {
    uint32_t major;
    uint32_t minor;
    uint32_t bugfix;
};

struct sb2_header_t {
    uint32_t nonce[4]; //!< Nonce for AES-CTR.
    uint32_t _padding0;
    uint32_t signature;  //!< 'STMP'
    uint8_t majorVersion;
    uint8_t minorVersion;
    uint16_t flags;
    uint32_t imageBlocks;
    uint32_t firstBootTagBlock;
    uint32_t firstBootableSectionID;
    uint32_t offsetToCertificateBlockInBytes;
```

```
    uint16_t headerBlocks;
    uint16_t keyBlobBlock;
    uint16_t keyBlobBlockCount;
    uint16_t maxSectionMacCount;
    uint32_t signature2;  //!< 'sgtl'
    uint64_t timestamp;
    version_t productVersion;
    version_t componentVersion;
    uint32_t buildNumber;  //!< User build number.
    uint32_t _padding1;
};
```

## Table 27-92. Header fields

| Field | Description |
|---|---|
| nonce | 128-bit random nonce for AES-CTR. Bit 31 is cleared.<br>The nonce is a 128-bit random value with bit 31 cleared. This allows the use of a 32-bit modular add instead of a full 128-bit add, while still supporting messages up to 2 (31 - 1) cipher blocks. See RFC5297 for more details. (RFC5297 specifies clearing bits 31 and 63, so that either 32- or 64-bit adds may be used. Because we only use 32-bit adds, bit 63 does not need to be cleared on the nonce.) |
| signature | Always set to 'STMP', big endian.<br>The two signature fields (signature, signature2) can be treated as 4-character strings without null-terminating bytes. The signature must be set to 'STMP'. That is, the byte 0 is 'S' and byte 3 is 'P'. Represented as a 32-bit little endian constant, the value would be (('S') \| ('T' << 8) \| ('M' << 16) \| ('P' << 24)). signature2 must be 'sgtl', thus in 32-bit little endian form would be (('s') \| ('g' << 8) \| ('t' << 16) \| ('l' << 24)). |
| majorVersion | Set to 2. |
| minorVersion | Set to 0. |
| flags | See the table below. |
| imageBlocks | Size of the entire image in blocks. |
| firstBootTagBlock | Offset in cipher blocks, from start of file to the block containing the first boot tag. |
| firstBootableSectionID | Unique identifier of the section to start booting from. |
| offsetToCertificateBlockInBytes | Offset in bytes to the certificate block header for a signed SB file. If the file is not signed, then offsetToCertificateBlockInBytes field will be 0. |
| headerBlocks | Size of the entire image header in blocks. |
| keyBlobBlock | Block number where the key blob starts. |
| keyBlobBlockCount | Number of cipher blocks occupied by the key blob. |
| maxSectionMacCount | Maximum number of HMAC table entries used in all sections of the SB file. |
| signature2 | Always set to 'sgtl', big endian. Also see signature above. |
| timestamp | Timestamp (in microseconds) size 1-1-2000 00:00, when the image was created. |
| productVersion | User-controlled product version, not used by the loader. |
| componentVersion | User-controlled component version, not used by the loader. |
| buildNumber | User-controlled build number. Compared with minBuildNumber from the kb_options_t struct passed into the API. |

## Table 27-93. Header flags

| Bit(s) | Flag | Description |
|---|---|---|

*Table continues on the next page...*

**Table 27-93. Header flags (continued)**

| | | |
|---|---|---|
| 0 | - | Reserved |
| 1 | - | Reserved |
| 2 | - | Reserved |
| 3 | ROM_SIGNED | The SB file is signed if set. offsetToCertificateBlockInBytes must be set appropriately.<br>**NOTE:** The ROM_SIGNED and ROM_CIPHER_MODE flags do not directly affect security; these 2 flags are only used for convenience and reference. |
| 5:4 | ROM_CIPHER_MODE | Block cipher mode used to encrypt section data.<br>2'b00 - AES-CTR<br>2'b01 - Reserved<br>2'b10 - Reserved<br>2'b11 - Reserved |
| 15:6 | - | Reserved |

### 27.8.6.1.2 Header MAC

The header MAC (HMAC) is a message authentication code that provides authentication of the SB file header using a secret key. The header MAC is computed with the HMAC-SHA256 algorithm [5][6] using a 256-bit secret key called K MAC. The HMAC is generated over the entire header plus the section table, including any required padding. HMAC is 32 bytes in size.

As soon as the header, MAC, and key blob are all available, the loader will recompute the MAC and compare the computed MAC with the MAC stored in the file. If the computed-MAC-to-stored-MAC compare fails, then the load is aborted with an error.

The MAC must follow the header and must start on an even block boundary.

### 27.8.6.1.3 Key blob

The key blob wraps two 256-bit keys using the RFC3394 algorithm, which provides integrity and authenticity over the wrapped keys. The two keys in the key blob are:

- Data Encryption Key ($K_{DEK}$): the KDEK is used to encrypt section data in the SB file.
- MAC Key ($K_{MAC}$): the KMAC is used for header HMAC and section HMACs.

The content of the key blob can be represented by this structure:

```
struct key_blob_t {
    uint8_t k_dek[32];
    uint8_t k_mac[32];
};
```

Both keys are uniquely generated each time that an SB file is built. The keys are wrapped with the SBKEK that is programmed into the target device's IFR or OTP. The key blob must follow the header MAC and must start on an even block boundary. Padding also must be added after the key blob, so that the following region starts on an even block boundary.

The key blob (containing the two 256-bit keys) will be 72 bytes in length; therefore, 8 bytes of padding are required.

#### 27.8.6.1.4 Sections

The content of SB files is divided into an arbitrary number of sections, each with a unique ID. Every section is preceded with a boot tag that acts as a header, plus an HMAC table. A section may be either a bootable section that contains boot commands, or a data section containing data not used by the loader. There must be at least 1 bootable section for an SB file to be valid.

##### 27.8.6.1.4.1 Boot tag

Boot tags prefix a section with the information about that section, and the boot tags form a linked list within the SB file, allowing the loader to sequentially search for a given section.

- Boot tags are always encrypted using AES-CTR.
- A boot tag is actually a ROM_TAG_CMD boot command. See Appendix A for the definition of boot commands.
- The C structure definition for a boot tag command is shown next, and the boot tag structure is the same as the boot_command_t structure.

```
struct boot_tag_t {
    uint8_t checksum;
    uint8_t tag;
    uint16_t flags;
    uint32_t address;
    uint32_t count;
    uint32_t data;
};
```

**Table 27-94. Boot tag fields**

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t |
| tag | 0x01 or ROM_TAG_CMD |
| flags | Section flags |
| address | Unique 32-bit identifier for this section.<br>Each section has a 32-bit unique ID, specified in the address field of the boot tag. |

*Table continues on the next page...*

**Table 27-94. Boot tag fields (continued)**

| Field | Description |
|---|---|
| count | The number of cipher blocks that the data for this section occupy.<br>The number of cipher blocks of section content is specified by the count field. The offset in cipher blocks to the next boot tag is denoted NextSectionOffset. It includes the size of the MAC table. |
| data | Section MAC table entry count |

**Table 27-95. Section flags**

| Bit(s) | Flag | Description |
|---|---|---|
| 0 | ROM_SECTION_BOOTABLE | • If ROM_SECTION_BOOTABLE is set, then the section is bootable and contains a sequence of bootloader commands.<br>• If ROM_SECTION_BOOTABLE is cleared, then the section is a data section and will be skipped by the loader. |
| 1 | ROM_SECTION_CLEARTEXT | The section is *unencrypted* if set. Note that unencrypted bootable sections are not supported. |
| 14:2 | - | Reserved |
| 15 | ROM_LAST_TAG | Indicates that there are no more sections in the image.<br>The flag ROM_LAST_TAG must be set on the boot tag for the last section, and indicates that there are no more sections in the image.<br>If the ROM_LAST_TAG flag is set, then there will not be another boot tag following the current one. In this case, NextSectionOffset would indicate the number of cipher blocks to either the end of the SB file, or to the RSA signature in the case of a signed SB file.<br>Let NextSectionOffset = ((boot_tag_t::data + 1) * 2) + boot_tag_t::count |

### 27.8.6.1.4.2 Section MAC table

Following the boot tag is a table of MACs used to verify the integrity and authenticity of section data. These MACs are computed using the HMAC-SHA256 algorithm with the K MAC key from the SB file's key blob. The number of MACs for a section is configurable by the user to trade between memory utilization and protection granularity.

The boot tag data field contains a count of the section MAC table entries that apply to section contents. The minimum count is 1. Unencrypted data sections are still required to have a MAC table.

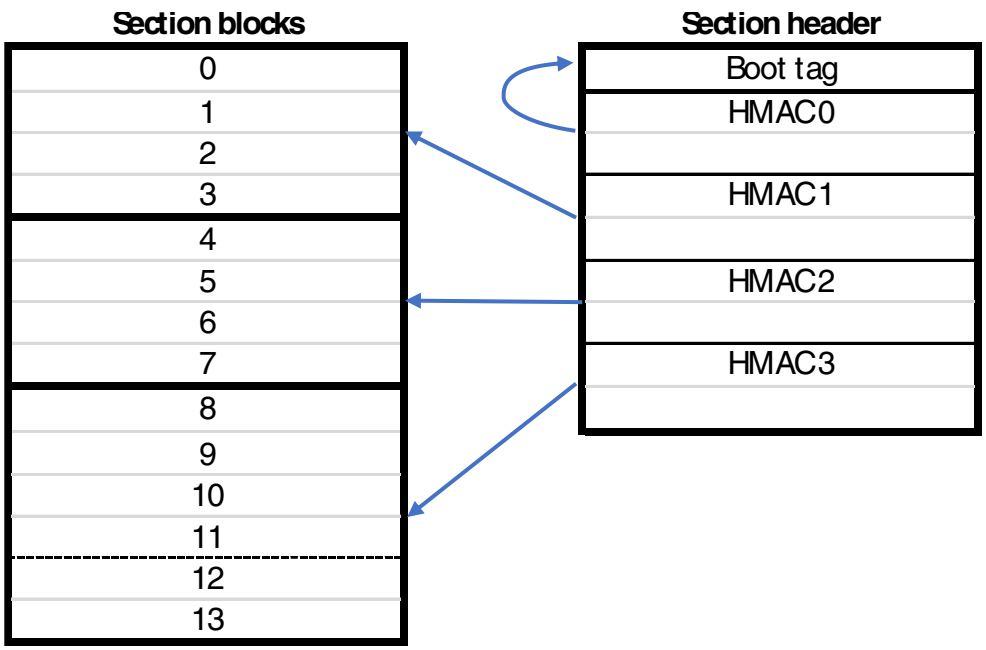
The first HMAC in the table is always the HMAC for the boot tag cipher block. Thus, the actual number of HMACs in the table (boot_tag_t::data + 1). The number of HMACs that apply to section contents is denoted HmacRangeCount.

```
Let HmacRangeCount = boot_tag_t::data
```

The section content cipher blocks are split into HmacRangeCount contiguous sequences of cipher blocks. These sequences are referred to as HMAC ranges. The number of cipher blocks in each HMAC range is denoted HmacRangeLen. Because the section length may not be evenly divisible by HmacRangeCount, the last HMAC range may contain more cipher blocks than the rest.

```
For j in 0..(HmacRangeCount - 1) do:
  HmacRangeLen_j  =  floor(boot_tag_t::count / HmacRangeCount)
  If j ==  (HmacRangeCount - 1) then:
    HmacRangeLen_j =  HmacRangeLen_j + boot_tag_t::count MOD
           HmacRangeCount
```

Here is an example showing how the section MAC table relates to section contents.




The variable values for this example are:

- boot_tag_t::count = 14
- boot_tag_t::data = 3
- HmacRangeCount = 3
- $HmacRangeLen_0 = 4$
- $HmacRangeLen_1 = 4$
- $HmacRangeLen_2 = 6$

The complete algorithm follows.

```
Let i = 0..boot_tag_t::count
Let S_i = AES-CTR  encrypted section cipher blocks
Let S_boot-tag  = boot tag cipher block
SH_0 = HMAC(K_MAC, S_boot-tag )
For j in 0..(HmacRangeCount - 1) do:
  Let b = floor(boot_tag_t::count / HmacRangeCount) * j
  Let q = HmacRangeLenj
  Let R j = S_b + 0  || S_b + 1  || … || S_b + q - 1
```

$$\text{Let } SHj+1 = HMAC(K_{MAC}, R_j)$$
$$\text{Let } SMT = SH_0 \,||\, SH_1 \,||\, \ldots \,||\, SH_{HmacRangeCount}$$

Section MAC table contents are never encrypted. The input to the HMACs is the AES-CTR output for the boot tag and section cipher blocks.

#### 27.8.6.1.4.3 Bootable section

A section that has the ROM_SECTION_BOOTABLE section flag set is called a bootable section. It contains a sequence of boot commands that are processed by the loader to perform a firmware update. Appendix A documents the structure of bootable section contents and boot commands.

Bootable sections may not be unencrypted.

#### 27.8.6.1.4.4 Data section

Any section for which the ROM_SECTION_BOOTABLE section flag is cleared is a data section. The loader does not examine the contents of such sections; it simply skips over them. Data sections may optionally be unencrypted by setting the ROM_SECTION_CLEARTEXT flag. The certificate block header is an example of a data section.

### 27.8.6.1.5 Certificate block header

As with all signed images supported by this design, a signed SB file must have a certificate block header. The certificate header in a signed SB file is stored in a data section with a tag of 'sign'. This section must have its ROM_SECTION_CLEARTEXT flag set to mark it as unencrypted. It also must have the ROM_SECTION_BOOTABLE flag cleared to cause the SB loader to skip the certificate header when executing the SB file.

The certificate chain and RKH table are also included in the 'sign' section, since they must always immediately follow the certificate block header. The 'sign' section itself may be at any position within the sequence of sections, though usually would be the first or last.

### 27.8.6.1.6 Signature

An SB file may optionally be signed. If signed, a signature in RSASSA-PKCS1-v1_5 format is appended to the end of the file. The digest is computed over the entire SB file except for the signature itself.

The imageBlocks field in the header includes the signature, if present, in the block count for the SB file.

### 27.8.6.2 Signed SB files

SB files that are signed will have the same structure as a signed image in internal flash.

Signed SB file must have some fields set to valid values.

- The offsetToCertificateBlockInBytes field in the file's header contains the offset to the certificate header.
- The certificate header and certificate(s) must be contained within an unencrypted, data-only section with an ID of 'sign'. This restriction is so that the SB loader can easily skip over the signature related structures (the SB file is presumed to have been authenticated prior to loading).
- The ROM_SIGNED flag is set in the header flags field.
- The signature in RSASSA-PKCS1-v1_5 format is appended to the tail end of the file.

### 27.8.6.3 Encryption

The contents of SB 2.0 files are always encrypted. The file's header is not encrypted, but is authenticated with an HMAC-SHA256. Several fields must be set with valid values.

- nonce must be set to a cryptographically secure random number, with bit 31 cleared.
- flags must have ROM_CIPHER_MODE set to 2'b00 to select AES-CTR.
- keyBlobBlock and keyBlobBlockCount must both be set to non-zero values.

Section contents are encrypted using AES-256 in CTR mode [9]. The 128-bit counter value for each cipher block of the file is equal to the nonce field from the file's header plus the offset from the beginning of the file of the cipher block being encrypted. Only a 32-bit add is required.

This code demonstrates how to compute the counter value for CTR mode.

```
uint32_t counter[4];
counter[0] = header->nonce[0];
counter[1] = header->nonce[1];
counter[2] = header->nonce[2];
counter[3] = header->nonce[3] + cipher_block_offset;
```

The 4 words of the nonce and counter are treated as little endian. Thus, it is actually the MSB of the last byte of the nonce that is cleared when bit 31 is cleared.

Data sections may optionally have the ROM_SECTION_CLEARTEXT flag set, in which case they are not encrypted. However, the boot tag for such sections is still encrypted, and there must be a MAC table.

## 27.8.7 Using SB files

SB files are always encrypted and may be signed.

- To authenticate an image (which is either signed code or an SB file), call the loader ROM API from the user application code.
- To perform authentication during secure firmware updates, the full SB file must be available.

To perform secure firmware updates:

1. When a user application receives an encrypted SB file containing new firmware, the user application stores the encrypted SB file in external SPI flash or in a similar memory.
2. To authenticate a SB file, use the ROM API.
3. To decrypt and load the SB file, use the ROM API.
4. Secure boot considerations:
   - If you are using secure boot, then before rebooting into it (the flash), use the ROM API to authenticate the new firmware in flash. If this final authentication fails, then the new firmware should be made non-executable (by erasing and writing over critical regions of it, like the vector table).
   - If you are not using secure boot, then the code written to flash can still be signed, to support this final authentication step.

### 27.8.7.1 Authenticate and load using a Secure Binary (SB) file

Two operations are available through the ROM API (load and authenticate a signed image, load an image but do not authenticate that image). The enum _kb_operation defines the two operations. The struct _kb_options defines which operation is selected.

```
typedef  enum _kb_operation {
    kKbootAuthenticate = 1,  //!< Authenticate a signed image.
    kKbootLoadSB = 2,        //!< Load an SB file.
 } kb_operation_t;
```

| | |
|---|---|
| kKbootAuthenticate | Load and authenticate a signed image |
| kKbootLoadSB | Load an SB file (but do not authenticate that file) |

### 27.8.7.2 Functions

**Table 27-96. Functions**

<table>
<tr><th>Function</th><th></th><th>Description</th></tr>
<tr><td>kb_init()</td><td>Initializes Kboot ROM API for a given operation<br><br><pre>status_t kb_init(kb_session_ref_t *session,
                 const kb_options_t *options);</pre></td><td>kb_init() uses the options provided by the application in the second argument.<br>• Every call to kb_init() should be paired with a call to kb_deinit().<br>• On return from kb_init(), the session parameter will be set to a new Kboot session reference. This session reference should then be passed to other ROM API calls.<br>• The options parameter is a pointer to a struct containing the option settings for the operation that is being initialized.<br><br><table><tr><th>Parameter</th><th>Description</th></tr><tr><td>session</td><td>On return from ROM API, the session parameter will be set to the new Kboot session reference; the session reference should then be passed to other ROM API calls.</td></tr><tr><td>options</td><td>Pointer to the struct containing the option settings for the operation that is being initialized</td></tr></table></td></tr>
<tr><td>kb_deinit()</td><td>Cleans up the Kboot ROM API session<br><br><pre>status_t kb_deinit(kb_session_ref_t session);</pre></td><td>After kb_deinit() returns, the session parameter must not be reused for any Kboot API calls.</td></tr>
<tr><td>kb_execute()</td><td>Perform the operation configured during initialization<br><br><pre>status_t kb_execute(kb_session_ref_t session,<br><br>const uint8_t *data,<br><br>uint32_t dataLength);</pre></td><td>The user application may call the kb_execute() function repeatedly, passing in sequential chunks of data from the boot image (SB file) that are to be processed. Kboot will perform the selected operation on this data and then return. The user application may call the kb_execute() function with as much or as little data as it wishes, which can be used to select the granularity of time given to the application inbetween executing the operation.<br><br><table><tr><th>Parameter</th><th>Description</th></tr><tr><td>session</td><td>Kboot session reference</td></tr><tr><td>data</td><td>Buffer of boot image data provided to Kboot by the application</td></tr><tr><td>dataLength</td><td>Length in bytes of the data in the buffer provided to Kboot</td></tr></table><br><table><tr><th>Return code</th><th>Description</th></tr><tr><td>kStatus_Success</td><td>The operation has completed successfully</td></tr><tr><td>kStatus_Fail</td><td>An error occurred while executing the operation</td></tr></table></td></tr>
</table>

**Table 27-96. Functions**

| Function | | Description | |
|---|---|---|---|
| | | **Return code** | **Description** |
| | | kStatus_MoreData | No error occurred, but Kboot needs more data to continue processing the boot image |

**Table 27-97. ROM API parameters**

| Parameter | | Code |
|---|---|---|
| ROM API version | Current Kboot API version | `enum _kb_api_version {`<br>`  kKbootApiVersion = 1,`<br>`};` |
| Operation | Operation selected, either load image and authenticate, or load image (but no not authenticate) | `typedef  enum _kb_operation {`<br>`  kKbootAuthenticate = 1,  //!< Authenticate a signed image.`<br>`  kKbootLoadSB = 2,  //!< Load an SB file.`<br>`} kb_operation_t;` |
| Memory region (in flash) | `address`: the starting address of of where you are loading the secure boot code image to, in RAM or flash | `typedef struct _kb_region {`<br>`  uint32_t address;`<br>`  uint32_t length;`<br>`} kb_region_t;` |
| | `length`: the length of the secure boot code image in RAM or flash, starting at the starting address | |
| | If a memory region is not specified, then all memory (RAM or flash, depending on what you are using for the secure boot image) is available, except for the ROM-reserved region. | |
| Security | RSA 4096-bit key size | `enum _kb_security_profile {`<br>`  kKbootMinRSA4096 = (1 << 20),`<br>`};` |

## 27.8.7.3 Initialization options

To configure the operation, define and pass the struct _kb_options into kb_init().

```
//! @brief Memory region definition.
typedef struct _kb_region {
    uint32_t address;
    uint32_t length;
} kb_region_t;

typedef struct _kb_options {
    uint32_t version;        //!< Should be set to #kKbootApiVersion.
    uint8_t *buffer;        //!< Caller-provided buffer used by Kboot.
    uint32_t bufferLength;
    kb_operation_t op;
    union {
         //! Settings for  #kKbootAuthenticate operation.
```

```
        struct {
            uint32_t profile;
            uint32_t minBuildNumber;
            uint32_t maxImageLength;
            uint32_t *userRHK;
        } authenticate;

         //! Settings for  #kKbootLoadSB operation.
        struct {
            uint32_t profile;
            uint32_t minBuildNumber;
            uint32_t regionCount;
             const kb_region_t *regions;
            uint32_t overrideSBBootSectionID;
            uint32_t *userSBKEK;
        } loadSB;
    };
} kb_options_t;
```

## Table 27-98. Initialization parameters

| Parameter | Description |
|---|---|
| address | The address of the region in flash that has the code image |
| length | Starting at the address parameter, the length defines the size of the region in flash (with the code image) |
| version | The version of the Kboot API in the device |
| buffer | Is a pointer to memory provided by the caller, which Kboot will use during execution of the operation. The minimum buffer size is the size of each certificate in the chain + 432 bytes per certificate. |
| bufferLength | Starting at the buffer address, the bufferLength defines the size of the buffer |
| op | The value of op defines the API operation selected (either load & authenticate the image, or just load the image) |
| profile | Is a mask that specifies which features are required in the SB file or in the image being processed. This includes the minimum AES and RSA key sizes.<br>• For profile mask constants, see the _kb_security_profile enum.<br>• The image being loaded or authenticated must match the profile, or an error will be returned. |
| minBuildNumber | Is an optional field that can be used to prevent version rollback. The ROM API will check the build number of the image, and if the build number is less than minBuildNumber, it will fail with an error. |
| maxImageLength | Used to verify the offsetToCertificateBlockInBytes value at the beginning of a signed image.<br>• Set maxImageLength to the length of the SB file.<br>• If verifying an image in flash, then maxImageLength can be set to the internal flash size or set to a large number (like 0x10000000). |
| userRHK | Can optionally be used by the user to override the RHK in IFR. If userRHK is not NULL, then userRHK points to a 32-byte array containing the SHA-256 of the root certificate's RSA public key. |
| profile | ? |
| regionCount | ? |
| regions | Points to an array of memory regions that the SB file being loaded is allowed to access. If regions is NULL, then all memory is accessible by the SB file. This feature is required to prevent a malicious image from erasing good code or from erasing RAM contents while it (RAM) is being loaded, only for the application to find that the image is inauthentic when it gets to the end of the section. |

*Table continues on the next page...*

**Table 27-98. Initialization parameters (continued)**

| Parameter | Description |
|---|---|
| overrideSBBootSectionID | Lets the caller override the default section of the SB file that is processed during a kKbootLoadSB operation. By default, the section specified in the firstBootableSectionID field of the SB header is loaded. If overrideSbBootSectionID is non-zero, then the section with the given ID will be loaded instead. |
| userSBKEK | Allows a user to provide their own AES-256 key (for unwrapping keys in an SB file during the kKbootLoadSB operation).<br>• userSBKEK should point to a 32-byte AES-256 key.<br>• If userSBKEK is NULL, then the IFR SBKEK will be used.<br>• After kb_init() returns, the caller should zero out the data pointed to by userSBKEK, because the ROM API will have installed the key in the crypto accelerator. |

### 27.8.7.4 Security profile flags

There are security profile flags that can be passed via the profile field of kb_options_t. The same security profile flags are used for the ROM_SECURITY_PROFILE IFR field.

```
enum _kb_security_profile {
    kKbootMinRSA4096 = (1 << 16),
};
```

| Bit(s) | Flag | Description |
|---|---|---|
| 15:0 | - | Reserved |
| 16 | kKbootMinRSA4096 | Specifies a minimum RSA key size of 4096 bits |
| 31:17 | - | Reserved |

The default profile with no bits set is:

- Encryption is not required for Secure Binary (SB) files.
- AES-256
- RSA-4096

Because only AES-256 is supported and 256 bits is the maximum AES key size, the profile does not require a minimum AES key size setting.

## 27.8.8 Security considerations

#### 27.8.8.1 General

Additional root RSA keys must be created in advance, so that their hash can be placed into the RKH table. This means that those keys are effectively sitting on a shelf somewhere. If an attacker is able to gain access to the first root private key, then there is a great likelihood that they may gain access to the other pregenerated root private keys. Physical and other protections can reduce this potential problem.

Pre-authentication and loading are separate steps. If the SB file is stored in external memory such as a SPI flash memory, this leads to a possibility of an attacker swapping SPI flash devices after authenticating the SB file (assuming it is signed) but before loading. Obviously the attacker would need physical access to the device, so this attack cannot be performed remotely. SB files being encrypted addresses this potential attack vector, but relies on a pre-shared secret key. To protect further, the customer can sign the code that is loaded into flash by the SB file. Then, after the load, but before switching to the new code, they can call the ROM API to authenticate what was written into flash. If authentication fails, they would wipe/mark the bad image. This is applicable whether using secure boot or not. If secure boot is enabled, then it provides another layer of defense, in that a modified image can never be executed even if it is successfully programmed.

The SBKEK is readable by software using a flash controller command.

#### 27.8.8.2 Hardware

If using secure boot or secure firmware update:

- Flash security must be enabled.
- Mass erase should be disabled.
- Flash security backdoor key can be enabled or not: it is optional. If the flash security backdoor key is enabled, then the backdoor key should only be used in special circumstances, such as product return debugging.

When waking up from the VLLS low power state via a non-reset wakeup source, this device cannot boot into ROM. This means that secure boot will be bypassed. Fortunately, VLLS mode can only be entered by code that presumably has been authenticated at the last reset.

#### 27.8.8.3 Kboot integration

If secure firmware updates are to be performed on a device that also has the Kboot ROM enabled for serial bootloading, then there are security implications with this configuration that need to be taken into consideration. Flash security must be enabled. This will disable

the majority of Kboot commands. The only method of writing to memory via serial interfaces when flash security is enabled is with encrypted SB files using the ReceiveSB command. For normal firmware updates, flash security must never be disabled using the back door key.

Signed SB files loaded over the Kboot serial interfaces will not be authenticated.

The minimum build number constraint is only supported when using the API. When loading SB files via the ROM's serial interface, the minimum build number is not checked. Thus, a customer who wishes to use the minimum build number to prevent version rollback, must load SB files using the API.

The allowed memory regions list is only supported when using the API. When loading SB files via the ROM's bootloader mode, all available memories are accessible.

## 27.8.9 Bootable section contents

A bootable section in an SB file contains a sequence of boot commands and any data required by those commands. The commands are processed in a linear sequence, starting with the first. Each boot command occupies a single cipher block, plus any cipher blocks required for data associated with that command.

### 27.8.9.1 Command structure

The C structure definition for a boot command:

```
struct boot_command_t {
    uint8_t checksum;
    uint8_t tag;
    uint16_t flags;
    uint32_t address;
    uint32_t count;
    uint32_t data;
};
```

checksum is a simple 8-bit checksum of each byte of boot_command_t, starting with tag. The checksum has a starting value of 0x5A. The tag field specifies the command. The other fields of boot_command_t have different meanings, depending on the actual command.

### 27.8.9.2 Available boot commands

| Tag Value | Mnemonic | Secure | Description |
|---|---|---|---|
| 0x00 | ROM_NOP_CMD | Y | No operation |
| 0x01 | ROM_TAG_CMD | Y | Boot tag |
| 0x02 | ROM_LOAD_CMD | Y | Load data into memory |
| 0x03 | ROM_FILL_CMD | Y | Fill memory with a pattern |
| 0x04 | - | - | Reserved |
| 0x05 | - | - | Reserved |
| 0x06 | - | - | Reserved |
| 0x07 | ROM_ERASE_CMD | Y | Erase flash sectors |
| 0x08 | - | - | Reserved |
| 0x09 | ROM_MEM_ENABLE_CMD | Y | Enable additional memory device |
| 0x0A | ROM_PROG_CMD | Y | Program persistent flash bits |

### 27.8.9.3 Command definition

#### 27.8.9.3.1 ROM_NOP_CMD

This command is a no-operation, and the bootloader simply skips over it. All fields except the m_tag fields are ignored by the bootloader, but should be set to 0. The ROM_NOP_CMD is used as command padding, if a bootable section following another bootable section is aligned to a value that required padding to be inserted.

| Field | Description |
|---|---|
| checksum | Simple checksum, which comes to 0x5a when all other fields are zeroes. |
| tag | 0x00 or ROM_NOP_CMD |
| flags | 0 |
| address | 0 |
| count | 0 |
| data | 0 |

#### 27.8.9.3.2 ROM_TAG_CMD

Used as a header for a section, specifying the section's ID, type, and size. Another way to think of it is as a linked list node.

The ROM_TAG_CMD command in not expected to appear within the command stream in a bootable section, and the bootloader will just ignore it if it is present. The purpose of this command definition is to describe the structure of the boot tag cipher block. Boot tags use the exact same structure as boot commands.

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |
| tag | 0x01 or ROM_TAG_CMD |
| flags | Bit 0: ROM_SECTION_BOOTABLEBit 1: ROM_SECTION_CLEARTEXTBit 15: ROM_LAST_TAG |
| address | Unique 32-bit identifier for this section. |
| count | The number of cipher blocks that the data for this section occupies. |
| data | Section MAC table entry count. |

The flag ROM_LAST_TAG must be set on the boot tag for the last section. ROM_LAST_TAG is used to indicate that there are no more boot tags in the image.

### 27.8.9.3.3 ROM_LOAD_CMD

The ROM_LOAD_CMD command is followed by an arbitrary number of cipher blocks that contain data to be loaded into memory (starting at the location specified by the m_address field of boot_command_t). The count field contains the number of bytes to be loaded to this location in memory.

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |
| tag | 0x02 or ROM_LOAD_CMD |
| flags | Bits 7:4: Memory group ID.Bits 15:8: Memory device ID. |
| address | Memory address to which the data will be stored. |
| count | Number of bytes to load. This is also the number of valid bytes in the data cipher blocks following this command. |
| data | CRC-32 over the data to be loaded. |

The number of cipher blocks following the command is ceil(count / 16). This means that there may be up to 15 bytes of padding in the last data cipher block.

There are no restrictions on alignment for the m_address or count fields. It is up to the loader implementation to decide how to best optimize loading of data. Thus, there is no guarantee on the order in which the data will be written to memory.

The data field contains a CRC-32 value computed over the data following the command header block. Any pad bytes in the last data cipher block are included in the CRC-32 calculation.

The flags field specifies the ID of the memory to which the data will be loaded. The ID is composed of a group ID and device ID. See The ROM_MEM_ENABLE_CMD for more information about memory IDs.

### 27.8.9.3.4 ROM_FILL_CMD

The ROM_FILL_CMD command is used to fill regions of memory with a bit pattern. The fill pattern is always a full 32 bits wide, but a byte-aligned fill length and target address are fully supported.

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |
| tag | 0x03 or ROM_FILL_CMD |
| flags | Always 0. |
| address | The starting memory address to which the fill pattern will be written. |
| count | Number of bytes to fill. |
| data | The fill pattern. Always replicated across the word regardless of the pattern size. |

The fill pattern, regardless of its actual size, must be spread across the entire data field. So a pattern that is a byte wide must be replicated four times across data, and twice for half-word patterns. When filling, the pattern is adjusted so that the most significant byte is aligned with the first byte to be filled.

Note that this command is guaranteed to use word writes between any unaligned ragged edges. This enables using the fill command as a word poke operation to write to registers.

The fill command only supports the standard CPU memory map (internal flash, RAM, and registers).

### 27.8.9.3.5 ROM_ERASE_CMD

The ROM_ERASE_CMD command applies only to devices with an internal flash memory array. It will execute a flash erase command for either the entire flash array or for the range of memory specified in the command fields.

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |
| tag | 0x07 or ROM_ERASE_CMD |
| flags | Bit 0: ROM_ERASE_ALL_MASK<br>• If set, erase all flash instead of only the specified range.<br>• If cleared, the address and count fields are used to determine the range of flash to erase. |

*Table continues on the next page...*

| Field | Description |
|---|---|
| | Bit 1: ROM_ERASE_ALL_UNSECURE_MASK<br>• If set, perform an erase all unsecure operation.<br>Bits 7:4: Memory group ID.<br>Bits 15:8: Memory device ID. |
| address | Start address of flash to erase. |
| count | Number of bytes of flash to erase. The end address is (address + count - 1). |
| data | 0 |

Bits 0 and 1 of the flags field determine whether the entire flash array will be erased, or if only a subset will be erased. If either ROM_ERASE_ALL_MASK or ROM_ERASE_ALL_UNSECURE_MASK are set, then the command will perform a mass erase operation. In addition to a mass erase, ROM_ERASE_ALL_UNSECURE_MASK programs the FSEC field to disable flash security. In this case, the address and count fields are ignored. ROM_ERASE_ALL_UNSECURE_MASK takes precedence over ROM_ERASE_ALL_MASK.

If ROM_ERASE_ALL_MASK and ROM_ERASE_ALL_UNSECURE_MASK are cleared, then the range of flash memory to erase is specified by the address and count command fields. Because flash memory can only be erased on a whole-sector basis, all flash sectors that are intersected by the address range will be erased. This applies even if the address range does not begin or end on an aligned sector boundary.

If an allowed memory region list is specified via the kb_options_t::loadSB.regions field when performing the kKbootLoadSB API operation, then the ROM_ERASE_ALL_MASK and ROM_ERASE_ALL_UNSECURE_MASK flags are ignored.

The memory group and device ID flags form a memory ID that specifies which part of memory that the erase operation will apply to. See ROM_MEM_ENABLE_CMD for more details about memory IDs. Note that ROM_ERASE_CMD ignores the memory ID and only applies to internal flash memory.

### 27.8.9.3.6 ROM_MEM_ENABLE_CMD

Enables additional, device specific memory areas. ROM_MEM_ENABLE_CMD is primarily used to enable QuadSPI.

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |

*Table continues on the next page...*

| Field | Description |
|---|---|
| tag | 0x09 or ROM_MEM_ENABLE_CMD |
| flags | See below. |
| address | Start address of memory. |
| count | Number of bytes in the memory. |
| data | 0 |

The bitfields of the flags field are defined here. These bitfields also apply to the flags for ROM_LOAD_CMD and ROM_ERASE_CMD.

| Bits | Mnemonic | Description |
|---|---|---|
| 15:8 | DEVICE_ID_MASK | Type of memory device. |
| 7:4 | GROUP_ID_MASK | Group of memory device. |

The GROUP_ID_MASK field current has two defined values.

| Value | Definition |
|---|---|
| 0x0 | Internal 4GB memory space. |
| 0x1 | External memory space. |

There are several predefined values for the flags field.

| Value | Memory device |
|---|---|
| 0x0000 | Internal flash |
| 0x0100 | QuadSPI |
| 0x0400 | IFR0 (Index Flash Region) |
| 0x0900 | FlexSPI NOR flash |
| 0x1000 | Execute only region of internal flash |
| 0x0110 | SPI NAND |
| 0x0410 | SPI EEPROM |
| 0x0510 | I2C EEPROM |

Note that support for these memory devices is entirely device-dependant.

### 27.8.9.3.7 ROM_PROG_CMD

Used to write program once regions of internal flash (IFR).

| Field | Description |
|---|---|
| checksum | Simple checksum of the other fields of boot_command_t. |
| tag | 0x0A or ROM_PROG_CMD |
| flags | Bit 0: ROM_PROG_8BYTE_MASK Write an 8 byte index instead of 4 byte.<br>Bits 7:4: Memory group ID.<br>Bits 15:8: Memory device ID. |
| address | Index of word to write. |
| count | Data word 1. |
| data | Data word 2. |

The memory ID flags must be set to 0x0400, selecting memory IFR0.

## 27.9 Verifying the application in flash using CRC-32

Using CRC-32 and a given address range, the ROM bootloader supports performing an application integrity check.

**Before application integrity testing:** the following fields in the bootloader configuration area must be set:

- Set crcStartAddress to the start address that should be used for the CRC check.
- Set crcByteCount to the number of bytes to run the CRC check on, from the start address.
- Set crcExpectedValue to the value that the CRC calculation should result in.

**Application integrity testing:**

- If all of the above fields are unset (all 0xFF bytes for crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader returns kStatus_AppCrcCheckInvalid.
- If any one of the above fields are set (crcStartAddress, crcByteCount, and crcExpectedValue), then the ROM bootloader checks if the given address range of the application is valid, and if the application resides in internal flash, by running a CRC check on the image in flash:
  - If the the given address range of the application is not valid (false), then the bootloader returns kStatus_AppCrcCheckOutOfRange.
  - If the given address range of the application is valid (true), then the CRC check is performed. The CRC32 algorithm in ROM uses a polynomial (0x04C1_1DB7), and the initial seed is 0xFFFF_FFFF.
    - If the CRC check fails, then the bootloader returns kStatus_AppCrcCheckFailed;
    - If the CRC check succeeds, then it returns kStatus_AppCrcCheckPassed.

**After application integrity testing:**

- If the bootloader returns kStatus_AppCrcCheckOutOfRange or kStatus_AppCrcCheckFailed, then an external pin (PTA6) will also be asserted, to indicate CRC check failure.

**NOTE**

PTA6 is only available on the 121 MAP BGA and 100 LQFP packages.

- The bootloader will only jump to the application if kStatus_AppCrcCheckPassed is returned; if kStatus_AppCrcCheckPassed is not returned, then the bootloader will stay active, and wait for further commands.

## 27.10 Kinetis Bootloader Status Error Codes

This section describes the status error codes that the Kinetis Bootloader returns to the host.

**Table 27-99. Kinetis Bootloader Status Error Codes, sorted by Value**

| Error Code | Value | Description |
|---|---|---|
| kStatus_Success | 0 | Operation succeeded without error. |
| kStatus_Fail | 1 | Operation failed with a generic error. |
| kStatus_ReadOnly | 2 | Requested value cannot be changed because it is read-only. |
| kStatus_OutOfRange | 3 | Requested value is out of range. |
| kStatus_InvalidArgument | 4 | The requested command's argument is undefined. |
| kStatus_Timeout | 5 | A timeout occurred. |
| kStatus_FlashSizeError | 100 | Not used. |
| kStatus_FlashAlignmentError | 101 | Address or length does not meet required alignment. |
| kStatus_FlashAddressError | 102 | Address or length is outside addressable memory. |
| kStatus_FlashAccessError | 103 | The FTFE_FSTAT[ACCERR] bit is set. |
| kStatus_FlashProtectionViolation | 104 | The FTFE_FSTAT[FPVIOL] bit is set. |
| kStatus_FlashCommandFailure | 105 | The FTFE_FSTAT[MGSTAT0] bit is set. |
| kStatus_FlashUnknownProperty | 106 | Unknown Flash property. |
| kStatus_FlashEraseKeyError | 107 | The key provided does not match the programmed flash key. |
| kStatus_FlashRegionExecuteOnly | 108 | The area of flash is protected as execute only. |
| kStatus_I2C_SlaveTxUnderrun | 200 | I2C Slave TX Underrun error. |
| kStatus_I2C_SlaveRxOverrun | 201 | I2C Slave RX Overrun error. |
| kStatus_I2C_AribtrationLost | 202 | I2C Arbitration Lost error. |
| kStatus_SPI_SlaveTxUnderrun | 300 | SPI Slave TX Underrun error. |

*Table continues on the next page...*

## Table 27-99. Kinetis Bootloader Status Error Codes, sorted by Value (continued)

| Error Code | Value | Description |
|---|---|---|
| kStatus_SPI_SlaveRxOverrun | 301 | SPI Slave RX Overrun error. |
| kStatus_SPI_Timeout | 302 | SPI tranfser timed out. |
| kStatus_SPI_Busy | 303 | SPI instance is already busy performing a transfer. |
| kStatus_SPI_NoTransferInProgress | 304 | Attempt to abort a transfer when no transfer was in progress. |
| kStatus_UnknownCommand | 10000 | The requested command value is undefined. |
| kStatus_SecurityViolation | 10001 | Command is disallowed because flash security is enabled. |
| kStatus_AbortDataPhase | 10002 | Abort the data phase early. |
| kStatus_Ping | 10003 | Internal: received ping during command phase. |
| kStatusRomLdrSectionOverrun | 10100 | The loader has finished processing the SB file. |
| kStatusRomLdrSignature | 10101 | The signature of the SB file is incorrect. |
| kStatusRomLdrSectionLength | 10102 | The section length in chunks is invalid. |
| kStatusRomLdrUnencryptedOnly | 10103 | An encrypted SB file has been sent and decryption support is not available. |
| kStatusRomLdrEOFReached | 10104 | The end of the SB file has been reached. |
| kStatusRomLdrChecksum | 10105 | The checksum of a command tag block is invalid. |
| kStatusRomLdrCrc32Error | 10106 | The CRC-32 of the data for a load command is incorrect. |
| kStatusRomLdrUnknownCommand | 10107 | An unknown command was found in the SB file. |
| kStatusRomLdrIdNotFound | 10108 | There was no bootable section found in the SB file. |
| kStatusRomLdrDataUnderrun | 10109 | The SB state machine is waiting for more data. |
| kStatusRomLdrJumpReturned | 10110 | The function that was jumped to by the SB file has returned. |
| kStatusRomLdrCallFailed | 10111 | The call command in the SB file failed. |
| kStatusRomLdrKeyNotFound | 10112 | A matching key was not found in the SB file's key dictionary to unencrypt the section. |
| kStatusRomLdrSecureOnly | 10113 | The SB file sent is unencrypted and security on the target is enabled. |
| kStatusRomLdrResetReturned | 10114 | The SB file reset operation has unexpectedly returned. |
| kStatusMemoryRangeInvalid | 10200 | Memory range conflicts with a protected region. |
| kStatus_UnknownProperty | 10300 | The requested property value is undefined. |
| kStatus_ReadOnlyProperty | 10301 | The requested property value cannot be written. |
| kStatus_InvalidPropertyValue | 10302 | The specified property value is invalid. |
| kStatus_AppCrcCheckPassed | 10400 | CRC check is valid and passed. |
| kStatus_AppCrcCheckFailed | 10401 | CRC check is valid but failed. |
| kStatus_AppCrcCheckInactive | 10402 | CRC check is inactive. |
| kStatus_AppCrcCheckInvalid | 10403 | CRC check is invalid, because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes). |
| kStatus_AppCrcCheckOutOfRange | 10404 | CRC check is valid but addresses are out of range. |

*Table continues on the next page...*

## Table 27-99. Kinetis Bootloader Status Error Codes, sorted by Value (continued)

| Error Code | Value | Description |
|---|---|---|
| kStatus_ReliableUpdateSuccess | 10600 | Reliable update process has completed successfully. |
| kStatus_ReliableUpdateFail | 10601 | Reliable update process has failed. |
| kStatus_ReliableUpdateInactive | 10602 | Reliable update feature is inactive. |
| kStatus_ReliableUpdateBackupApplicationInvalid | 10603 | Backup application image is invalid. |
| kStatus_ReliableUpdateStillInMainApplication | 10604 | The next boot will still use the main application image. |
| kStatus_ReliableUpdateSwapSystemNotReady | 10605 | Cannot swap flash by default, because the swap system is not ready. |
| kStatus_ReliableUpdateBackupBootloaderNotReady | 10606 | Cannot swap flash, because there is no valid backup bootloader image. |
| kStatus_ReliableUpdateSwapIndicatorAddressInvalid | 10607 | Cannot swap flash, because the provided swap indicator is invalid. |

# Chapter 28
# Clock Distribution

## 28.1 Introduction

This chapter provides an overview of the system level clock architecture. This includes the clock generation, division, multiplexing, gating, and distribution for this device.

## 28.2 Clock control modules

The core of the clocking architecture is the System Clock Generator (SCG) module and the Peripheral Clock Controller (DTEST_EN_MASK) module. The SCG controls the configuration of the system level clocks while the PCC controls the distribution of the clocks to the peripheral modules. The following diagram shows the high-level conceptual implemention of the SCG and the PCC. Each of these modules will be described in the following sections.

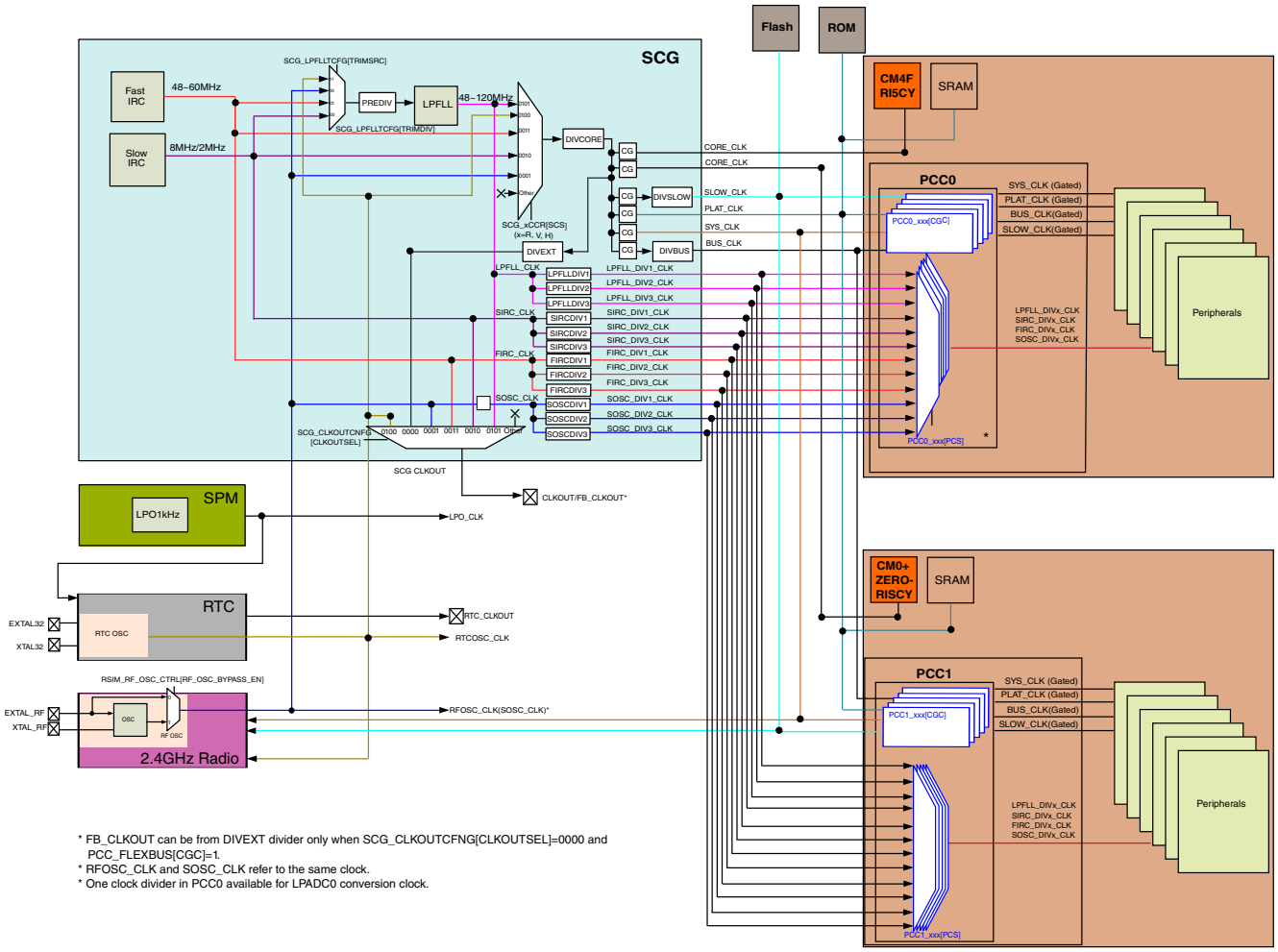



**Figure 28-1. Generic clocking architecture diagram**

## 28.2.1 System Clock Generation (SCG) module

The System Clock Generation (SCG) module provides a central location for accessing all of the system clock controls. The SCG controls the enables for most of the system clock sources as well as their configuration, the clock dividers and clock gates for all of the system clocks, and the dividers for the peripheral functional clocks. System clock sources that are tightly copupled with another module, such as the Real Time Clock Oscillator (RTCOSC) with the battery voltage domain ($V_{BAT}$) or the Low-Power Oscillator (LPO) and the System Power Manager (SPM), are controlled from those modules.

Each system clock resource has configuration registers in the SCG. These configuration registers provide control of the features as well as the enable for that clock source. These clock features can be output frequency ranges, trim controls, or frequency or phase detector settings. These controls enable the clocks to be configured to meet the needs of multiple applications. For details on the configuration of the clock sources, please refer to

the System Clock Generation chapter of the Reference Manual. Although the Real Time Clock (RTC) oscillator output can be used by the SCG, the RTC clock is a part of the battery backup domain and has its controls in that domain.

The SCG registers control the selection of the root clock for all of the system clocks. The root clock can be configured for each of the operational modes of the device to achieve the appropriate balance between performance and low-power. Once the root clock is selected, a divider can be set to reduce the frequency. While each system clock can have a divider, not all of them are implemented in each device. Please refer to the section SCG system output clocks to see which clock dividers are present in this device.

Finally, the SCG provides three groups of clocks to the Peripheral Clock Control (PCC) module that are used as the peripheral functional clocks. The three groups of clocks are the "Platform" clocks, the "Bus" clocks, and the "Slow" clocks. The "Platform" clocks are for peripherals that will most likely operate at the highest speeds produced by the device. This includes peripherals such as the Universial Serial bus (USB). The "Bus" clocks are for a majority of the peripherals that operate at equal frequencies as or lower frequencies than the platform clocks, but still need the performance of a fast clock. These peripherals can be timers or serial communication peripherals. The "Slow" clocks are for peripherals that do not have a high performance requirements. These peripherals can be certain timers or monitors.

## 28.2.2 Peripheral Clock Control (PCC) module

The Peripheral Clock Control (PCC) module controls all of the clocks that are distributed to the peripherals. Most peripherals have two clocks: the peripheral interface clock and the peripheral functional clock. The peripheral interface clock drives the peripheral's registers and system interface. There are three clocks that can be used for the peripheral interface clock: the "Platform" clock, the "Bus" clock, and the "Slow" clock. The "Platform" clock is used for high-speed peripherls that will operate at the highest clock rates that the device can produce. These peripherals will many times be bus masters in the system. The "Bus" clock is used for most peripherals. These are peripherals that still have performance requirements, but do not need to operate at the maximum operational frequency. The "Slow" clocks are for peripherals that do not require performance and are monitoring long term events.

The peripheral functional clock provides the clock for the peripherals primary operation. This could be the BAUD clock for a serial commincation peripheral or the counter clock in a timer. This allows the system bus clock to be reduced to save power while continuing to operate the peripheral without any configuration changes. As with the peripheral

interface clocks, there are groups of clocks for the peripheral functional clocks: the "Platform", "Bus", and "Slow" clock groups. Altough these clocks are used for similar groupings, the clocks are completely independent.

## 28.3 SCG input clock sources

The System Clock Generator (SCG) has multiple clock sources that can be selected as the root clock of the system clock outputs and their respective clock trees. The input clock sources for a given device may consist of Internal Reference Clocks (IRCs), crystal oscillators (OSCs), Phase Locked Loops (PLLs), and Frequency Locked Loops (FLLs). Typically, each device has at least one IRC, one external OSC, and either a PLL or FLL.

The following table lists the input clocks available in this device. For details on these clock sources and their configuration options, please refere to the SCG chapter.

**Table 28-1. Input clock sources**

| Clock | Frequencies | Source |
|---|---|---|
| Fast Frequency Internal Reference Clock (FIRC) | 48MHz<br>52MHz<br>56MHz<br>60MHz | SCG |
| Slow Frequency Internal Reference Clock (SIRC) | 2MHz<br>8MHz | SCG |
| Real Time Clock Oscillator (RTCOSC) | 32.768kHz | $V_{BAT}$ Domain |
| Radio Frequency Oscillator (RFOSC) | 26MHz<br>32MHz | Radio |
| Low-Power Frequency Locked Loop (LPFLL) | 48MHz<br>72MHz | SCG |

## 28.4 SCG system output clocks

The SCG module is the source of all system clocks. The different system clock trees are provided to allow flexible frequency configuration and gating control for the processor core, the platform modules, interfaces, and external memories. The system output clocks available in this devices are described in the following table.

**Table 28-2. System clock frequencies**

| System Clock | Source | Divider | Description |
|---|---|---|---|
| System Core Clock (CORE_CLK) | Any of the SCG input clock sources | ÷ 1 to 16 | The System Core Clock (CORE_CLK) is the main system clock and is the root of the processor core's clock tree. The CORE_CLK is generally the fastest clock in the system, but it can also run slower than peripheral functional clocks in order to save power. In general, all other system clock trees are fed from the CORE_CLK. |
| System Platform Clock (PLAT_CLK) | CORE_CLK | ÷ 1 | The System Platform Clock (PLAT_CLK) is used to clock platform modules that directly interface with the core processor or bus-matrix. These platform modules can include modules such as the Direct Memory Access (DMA) controller, the Crossbar Switch Lite (AXBS-Lite), or the Low-Power Local Memory (LPLMEM) caches. |
| System Bus Clock (BUS_CLK) | CORE_CLK | ÷ 1 to 16 | The System Bus Clock (BUS_CLK) is used to clock the majority of peripherals that require a low latency interface with the processor core or a bus master. Most timers, serial communications, LPADC peripherals use the BUS_CLK. |
| System Slow Clock (SLOW_CLK) | CORE_CLK | ÷ 1 to 16 | The System Slow Clock (SLOW_CLK) is used to clock the flash memory module. In addition, the SLOW_CLK can be used for peripherals that do not require low latency interfacing with the core processor or other bus masters. Peripherals and modules that are used for configuration of the MCU, such as the System Clock Generator (SCG), the System Power Manager (SPM), and the Multi System Mode Controller (MSMC) can use the SLOW_CLK as their clock source. |
| System External Clock (EXT_CLK) | CORE_CLK | ÷ 1 to 16 | The System External Clock (EXT_CLK) is used to clock peripherals that access external memories. These peripherals can include the External Bus Interface (FlexBus) module. |

## 28.5 SCG peripheral output clocks

The SCG module is the source of all peripheral functional clocks. The different peripheral clock trees are provided to allow flexible frequency configuration and gating control for the peripherals. The peripheral output clocks available in this devices are described in the following table.

**Table 28-3. System clock frequencies**

| Clock | Frequencies | Outputs | Divider |
|---|---|---|---|
| Fast Frequency Internal Resistor Capactitor Oscillator (FIRC) | 48MHz | Platform | ÷ 1 to 64 (powers-of-2)[1] |
| | 52MHz | Bus | ÷ 1 to 64 (powers-of-2)[1] |
| | 56MHz | Slow | ÷ 1 to 64 (powers-of-2)[1] |
| | 60MHz | | |
| Slow Frequency Internal Resistor Capactitor Oscillator (SIRC) | 2MHz | Platform | ÷ 1 to 64 (powers-of-2) |
| | 8MHz | Bus | ÷ 1 to 64 (powers-of-2) |

*Table continues on the next page...*

**Table 28-3. System clock frequencies (continued)**

| Clock | Frequencies | Outputs | Divider |
|---|---|---|---|
| | | Slow | ÷ 1 to 64 (powers-of-2) |
| Real Time Clock Oscillator (RTCOSC) | 32.768kHz | Platform | ÷ 1 |
| | | Bus | ÷ 1 |
| | | Slow | ÷ 1 |
| Radio Frequency Oscillator (RFOSC) | 26MHz | Platform | ÷ 1 to 64 (powers-of-2) |
| | 32MHz | Bus | ÷ 1 to 64 (powers-of-2) |
| | | Slow | ÷ 1 to 64 (powers-of-2) |
| Low-Power Frequency Locked Loop (LPFLL) | 48MHz | Platform | ÷ 1 to 64 (powers-of-2)[1] |
| | 72MHz | Bus | ÷ 1 to 64 (powers-of-2)[1] |
| | 96MHz | Slow | ÷ 1 to 64 (powers-of-2)[1] |
| | 120MHz | | |

1. Even though the clock source is capable of generating higher frequencies, the appropriate divider value should be used to keep the clocks below their maximum allowable values for each of the operational modes. Please refer to the the Clock frequencies section to determine the maximum frequencies for this device.

## 28.6 Clock definitions

The following table lists the clock used in this device.

**Table 28-4. Clock definitions**

| Clock | Description |
|---|---|
| SCG Input Clocks | |
| FIRC | Fast Frequency Internal Resistor Capacitor oscillator - Clock input from the FIRC oscillator with a frequency of 48 Mhz, 52 MHz, 56 MHz, or 60 MHz. |
| SIRC | Slow Frequency Internal Resistor Capacitor oscillator - Clock input from the SIRC oscillator with a frequency of 2 MHz or 8 MHz. |
| LPOSC | Low-Power Oscillator - Clock input from the Low-Power Oscillator module located in the System Power Management (SPM) module with a frequency of 1 kHz. |
| RTCOSC | Real Time Clock Oscillator - Clock input from the Real Time Clock (RTC) oscillator located in the Battery voltage domain ($V_{BAT}$) with a frequency of 32.768 kHz. |
| RFOSC | Radio Frequency Oscillator - Clock input from the Radio module with a frequency of 26 MHz of 32 MHz depending on the external crystal connected to the oscillator. |
| LPFLL | Low-Power Frequency Locked Loop - Clock input from the Low-Power Frequency Locked Loop with a frequency of 48MHz, 72MHz, 96MHz, or 120MHz. |
| System Output Clocks | |
| CORE_CLK | System Core Clock - The clock provided to the processor cores and tighly coupled memories, modules, and peripherals. The CORE_CLK is derived from one of the SCG's input clock sources. |
| SYS_CLK | System Clock - The clock provided to cross bar masters. The SYS_CLK is derived from the CORE_CLK. |

*Table continues on the next page...*

**Table 28-4. Clock definitions (continued)**

| Clock | Description |
|---|---|
| PLAT_CLK | System Platform Clock - The clock provided to the platform modules and peripherals. The PLAT_CLK is derived from the CORE_CLK. |
| BUS_CLK | System Bus Clock - The clock provided to the system buses to interface with low latency peripherals such as timer and serial communication peripherals. The BUS_CLK is derived from the CORE_CLK. |
| SLOW_CLK | System Slow Clock - The clock provided to the flash memory and configuration peripherals. The SLOW_CLK is derived from CORE_CLK. |
| EXT_CLK | System External Clock - The clock provided to peripherals that interface with external memories. The EXT_CLK is derived from the CORE_CLK. |
| PCC Input Clocks | |
| SIRC_PLAT | SIRC Platform Clock - Clock from the SIRC intended for use with modules using the Platform Interface clock. |
| SIRC_BUS | SIRC Bus Clock - Clock from the SIRC intended for use with modules using the Bus Interface clock. |
| SIRC_SLOW | SIRC Slow Clock - Clock from the SIRC intended for use with modules using the Slow Interface clock. |
| FIRC_PLAT | FIRC Platform Clock - Clock from the FIRC intended for use with modules using the Platform Interface clock. |
| FIRC_BUS | FIRC Bus Clock - Clock from the FIRC intended for use with modules using the Bus Interface clock. |
| FIRC_SLOW | FIRC Slow Clock - Clock from the FIRC intended for use with modules using the Slow Interface clock. |
| RTCOSC_PLAT | RTCOSC Platform Clock - Clock from the RTCOSC intended for use with modules using the Platform Interface clock. |
| RTCOSC_BUS | RTCOSC Bus Clock - Clock from the RTCOSC intended for use with modules using the Bus Interface clock. |
| RTCOSC_SLOW | RTCOSC Slow Clock - Clock from the RTCOSC intended for use with modules using the Slow Interface clock. |
| RFOSC_PLAT | RFOSC Platform Clock - Clock from the RFOSC intended for use with modules using the Platform Interface clock. |
| RFOSC_BUS | RFOSC Bus Clock - Clock from the RFOSC intended for use with modules using the Bus Interface clock. |
| RFOSC_SLOW | RFOSC Slow Clock - Clock from the RFOSC intended for use with modules using the Slow Interface clock. |
| LPFLL_PLAT | LPFLL Platform Clock - Clock from the LPFLL intended for use with modules using the Platform Interface clock. |
| LPFLL_BUS | LPFLL Bus Clock - Clock from the LPFLL intended for use with modules using the Bus Interface clock. |
| LPFLL_SLOW | LPFLL Slow Clock - Clock from the LPFLL intended for use with modules using the Slow Interface clock. |

## 28.7 Clock gating

Gating the clock of modules that are not operating helps reduce the power consumed during operation. This in turn increases the battery life for mobile devices. Clocks should only be enabled to peripherals when they are needed.

Each peripheral has independent clock gates for both the peripheral interface clock and the peripheral functional clock. This clock gating is controlled through the peripheral's PCC register. Clearing the CGC filed of the PCCn register will disable the peripheral interface clock, while clearing the PCS field of the PCCn register will disable the peripheral functional clock.

The SCG also has the ability to clock gate the peripheral functional clocks of groups of peripherals at the same time. Setting the xxxDIVn field of the SCG_xxxDIV register to 0x00 will disable the clock divider and clock gate the output of the clock. All peripherals that use this clock as their functional clock will be clock gated at the same time.

The CORE_CLK and SLOW_CLK are always enabled in any of the "Run" or "Wait" modes. If CORE_CLK and SLOW_CLK are not used in the low power stop modes, they will be disabled by the Multi System Mode Controller (MSMC) and System Power Manager (SPM) modules.

## 28.8 Module clock assignments

The table below provides the mapping between the device modules and the associated system clock trees.

The table below provides the mapping between the device modules and the associated system clock trees.

**Table 28-5. Clock assignments**

| Module Name | Bus Interface Clock | PCC Clock Gate | Peripherial Functional Clock | Clock Domain[1] | I/O Interface Clocks |
|---|---|---|---|---|---|
| Core Modules | | | | | |
| ARM Cortex-M0+ Core | CORE_CLK | — | — | SCG | — |
| ARM Cortex-M4 Core | CORE_CLK | — | — | SCG | — |
| CM0+ TRACE | SYS_CLK | Yes | SYS_CLK | SCG | — |
| CM4F TRACE | SYS_CLK | Yes | SYS_CLK, SOSC_DIV1_CLK, SIRC_DIV1_CLK, FIRC_DIV1_CLK, LPFLL_DIV1_CLK | PCC0 | TRACE_CLK_OUT |
| JTAG | NA | — | TCK | — | JTAG_TCLK |
| SWD | PLAT_CLK | — | SWCK | SCG[2] | SWD_CLK |
| Platform Modules | | | | | |
| CM0+ Crossbar | PLAT_CLK | — | NA | SCG | — |
| CM4F Crossbar | PLAT_CLK | — | NA | SCG | — |

*Table continues on the next page...*

## Table 28-5. Clock assignments (continued)

| Module Name | Bus Interface Clock | PCC Clock Gate | Peripherial Functional Clock | Clock Domain[1] | I/O Interface Clocks |
|---|---|---|---|---|---|
| xRDC | PLAT_CLK | Yes | — | PCC0,PCC1 | — |
| SEMA42 | PLAT_CLK | Yes | — | PCC0,PCC1 | — |
| System Modules | | | | | |
| MSMC_SMC0 | SLOW_CLK | Yes | — | PCC0 | — |
| MSMC_SMC1 | SLOW_CLK | Yes | — | PCC1 | — |
| INTMUX | BUS_CLK | | — | SCG | — |
| DMA0 | SYS_CLK | Yes | — | PCC0 | — |
| DMA1 | SYS_CLK | Yes | — | PCC1 | — |
| DMA MUX0 | BUS CLK | Yes | — | PCC0 | — |
| DMA MUX1 | BUS CLK | Yes | — | PCC1 | — |
| EWM | SLOW_CLK | Yes | LPO clock. | PCC0[2] | — |
| LLWU0 | SLOW_CLK | Yes | LPO | SCG[2] | — |
| LLWU1 | SLOW_CLK | Yes | LPO | SCG[2] | — |
| MU | SLOW_CLK | Yes | — | PCC0,PCC1 | — |
| SPM | SLOW_CLK | Yes | — | | — |
| TRGMUX0 | SLOW_CLK | Yes | — | SCG | — |
| TRGMUX1 | SLOW_CLK | Yes | — | SCG | — |
| WDOG0 | SLOW_CLK | Yes | RFOSC, LIRC, or LPO clock. | PCC0[2] | — |
| WDOG1 | SLOW_CLK | Yes | RFOSC, LIRC, or LPO clock. | PCC1[2] | — |
| Memories | | | | | |
| Flash Controller (FMC) | PLAT_CLK | — | — | SCG | — |
| Flash Memory (FTFE) | SLOW_CLK | — | — | SCG | — |
| FlexBus | PLAT_CLK | Yes | — | PCC0[2] | FB_CLKOUT/ CLKOUT |
| CM4F SRAM | PLAT_CLK | — | PLAT_CLK | SCG | — |
| CM0+ SRAM | PLAT_CLK | — | PLAT_CLK | SCG | — |
| Security and Integrity | | | | | |
| CAU3 | SYS_CLK | Yes | — | PCC1 | — |
| CRC | BUS_CLK | Yes | — | PCC0 | — |
| TRNG | BUS_CLK | Yes | — | PCC1 | — |
| Timers | | | | | |
| LPIT0 | SLOW_CLK | Yes | SOSC_DIV3_CLK, SIRC_DIV3_CLK, FIRC_DIV3_CLK, LPFLL_DIV3_CLK | PCC0 | — |

*Table continues on the next page...*

## Table 28-5. Clock assignments (continued)

| Module Name | Bus Interface Clock | PCC Clock Gate | Peripherial Functional Clock | Clock Domain[1] | I/O Interface Clocks |
|---|---|---|---|---|---|
| LPIT1 | SLOW_CLK | Yes | SOSC_DIV3_CLK, SIRC_DIV3_CLK, FIRC_DIV3_CLK, LPFLL_DIV3_CLK | PCC1 | — |
| RTC | SLOW_CLK | Yes | RTCOSC, LPO | SCG[2] | RTC_CLKOUT |
| LPTMR0/1 | SLOW_CLK | — | SIRC_DIV3_CLK,LPO, RTCOSC, RFOSC | SCG[2] | — |
| LPTMR2 | SLOW_CLK | — | SIRC_DIV3_CLK,LPO, RTCOSC, RFOSC | SCG[2] | — |
| LPTPM0~2 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC0[2] | TPM0~32_CLKIN |
| LPTPM3 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC1[2] | TPM3_CLKIN |
| TSTMR | SLOW_CLK | Auto | SIRC_1MHZ_CLK | SCG | — |
| Communication Interfaces | | | | | |
| EMVSIM | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK, RTCOSC | PCC0 | EMVSIM_CLK |
| FlexIO | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC0 | — |
| LPI2C0~2 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC0 | LPI2C0~2_SCL |
| LPI2C3 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC1 | LPI2C3_SCL |
| RF Analog | — | — | RFOSC | SCG[2] | RF_BSM_CLK |
| RF RSIM | SLOW_CLK | — | RTC OSC | SCG[2] | RF_BSM_CLK |
| RF XCVR | SYS_CLK | — | RFOSC | SCG[2] | RF_BSM_CLK |
| RF Bluetooth Link Layer | SYS_CLK | — | RTC OSC/RFOSC | SCG[2] | RF_BSM_CLK |
| RF Generic Link Layer | SYS_CLK | — | RFOSC | SCG[2] | RF_BSM_CLK |
| RF IEEE 802.15.4 Link Layer | SYS_CLK | — | RFOSC | SCG[2] | RF_BSM_CLK |

*Table continues on the next page...*

## Table 28-5. Clock assignments (continued)

| Module Name | Bus Interface Clock | PCC Clock Gate | Peripherial Functional Clock | Clock Domain[1] | I/O Interface Clocks |
|---|---|---|---|---|---|
| I2S | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK, SAI_MCLK | PCC0[3] | SAI_TX_BCLK, SAI_RX_BCLK, SAI_MCLK |
| uSDHC | SYS_CLK | Yes | SOSC_DIV1_CLK, SIRC_DIV1_CLK, FIRC_DIV1_CLK, LPFLL_DIV1_CLK, SDHC_DCLK | PCC0 | SDHC_DCLK |
| LPSPI0~2 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC0 | LPSPI0~2_SCK |
| LPSPI3 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC1 | LPSPI3_SCK |
| LPUART0~2 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC0 | — |
| LPUART3 | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK | PCC1 | — |
| USB | SYS_CLK | Yes | SOSC_DIV1_CLK, SIRC_DIV1_CLK, FIRC_DIV1_CLK, LPFLL_DIV1_CLK add note on IRC48 | PCC0[2] | — |
| USB SRAM | SOSC_DIV1_CLK, SIRC_DIV1_CLK, FIRC_DIV1_CLK, LPFLL_DIV1_CLK, SYS_CLK | Yes | — | PCC0[2] | — |
| Human-Machine Interfaces | | | | | |
| PORTA~D | SLOW_CLK | Yes | SLOW_CLK, LPO | PCC0 | — |
| PORTE | SLOW_CLK | Yes | SLOW_CLK, LPO | PCC1 | — |
| RGPIOA~D | PLAT_CLK | Yes | — | PCC0 | — |
| RGPIOE | PLAT_CLK | Yes | — | PCC1 | — |
| Analog | | | | | |
| LPADC | BUS_CLK | Yes | SOSC_DIV2_CLK, SIRC_DIV2_CLK, FIRC_DIV2_CLK, LPFLL_DIV2_CLK, LPADC Internal Clock. | PCC0 | — |

*Table continues on the next page...*

## Table 28-5. Clock assignments (continued)

| Module Name | Bus Interface Clock | PCC Clock Gate | Peripherial Functional Clock | Clock Domain[1] | I/O Interface Clocks |
|---|---|---|---|---|---|
| LPCMP0 | SLOW_CLK | — | — | SCG | — |
| LPCMP1 | SLOW_CLK | — | — | SCG | — |
| LPDAC | BUS_CLK | Yes | — | PCC0 | — |
| VREF | SLOW_CLK | Yes | — | PCC0 | — |

1. PPC0 controls CM4F domain,and PPC1 controls CM0+ domain.
2. It is bus interface clock domain.
3. SAI_MCLK doesn't blong to PCC clock domain.

# Chapter 29
# System Clock Generator (SCG)

## 29.1 Chip-specific SCG information

### 29.1.1 LPFLL configuration

The LPFLLCFG[FSEL] is used to select frequency. For this device, LPFLL is factory default trimmed to 72 MHz, nominal conditions trim value for 72 MHz is automatically loaded at reset. To use the LPFLL 48 MHz as clock source, the user must first trim the LPFLL to 48 MHz.

## 29.2 Introduction

The system clock generator (SCG) module provides the system clocks of the MCU. The SCG contains a frequency-locked loop (LPFLL), a slow internal reference clock (SIRC), a fast internal reference clock (FIRC), and the system oscillator clock (SOSC). The LPFLL trimming is controlled by the either the SOSC reference clock, ROSC reference clock, SIRC reference clock or FIRC reference clock. The SCG can select either the output clock of the LPFLL, or a SCG reference clock (SIRC, FIRC, and SOSC) as the source for the MCU system clocks. The SCG also supports operation with crystal oscillators, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock (which are also available as clock sources for the MCU systems clocks).

### 29.2.1 Features

Key features of the SCG module are:

- Low Power Frequency Locked-Loop (LPFLL):

    - Programmable multiplier for up to 4 different frequency ranges
    - Internal reference clocks or oscillators reference clocks can be used as the LPFLL source for trimming purposes
    - Can be selected as the clock source for the MCU system clocks
    - 3 programmable post-divider clock outputs, which can be used as a clock source for other on-chip peripherals
- 2 Internal reference clock (IRC) generators:
    - Slow IRC clock with programmable High and Low frequency range, with each range having a set of 8 trim bits for accuracy
    - Fast IRC clock with programmable High and Low frequency range, with 3 sets of trim bits for accuracy
    - Either the slow or the fast clock can be selected as the clock source for the MCU system clocks
    - 3 programmable post-divider clock outputs for each IRC, which can be used as clock sources for other on-chip peripherals
- System Crystal Oscillator:
    - Can be used as a source for the LPFLL
    - Can be selected as the clock source for the MCU system clocks
- External clock from the Real Time Counter (RTC):
    - Can be used as a source for the LPFLL
    - Can be selected as the clock source for the MCU
- Clock monitor with reset and interrupt request capability for SOSC, ROSC clocks

The next figure shows the SCG module block diagram.

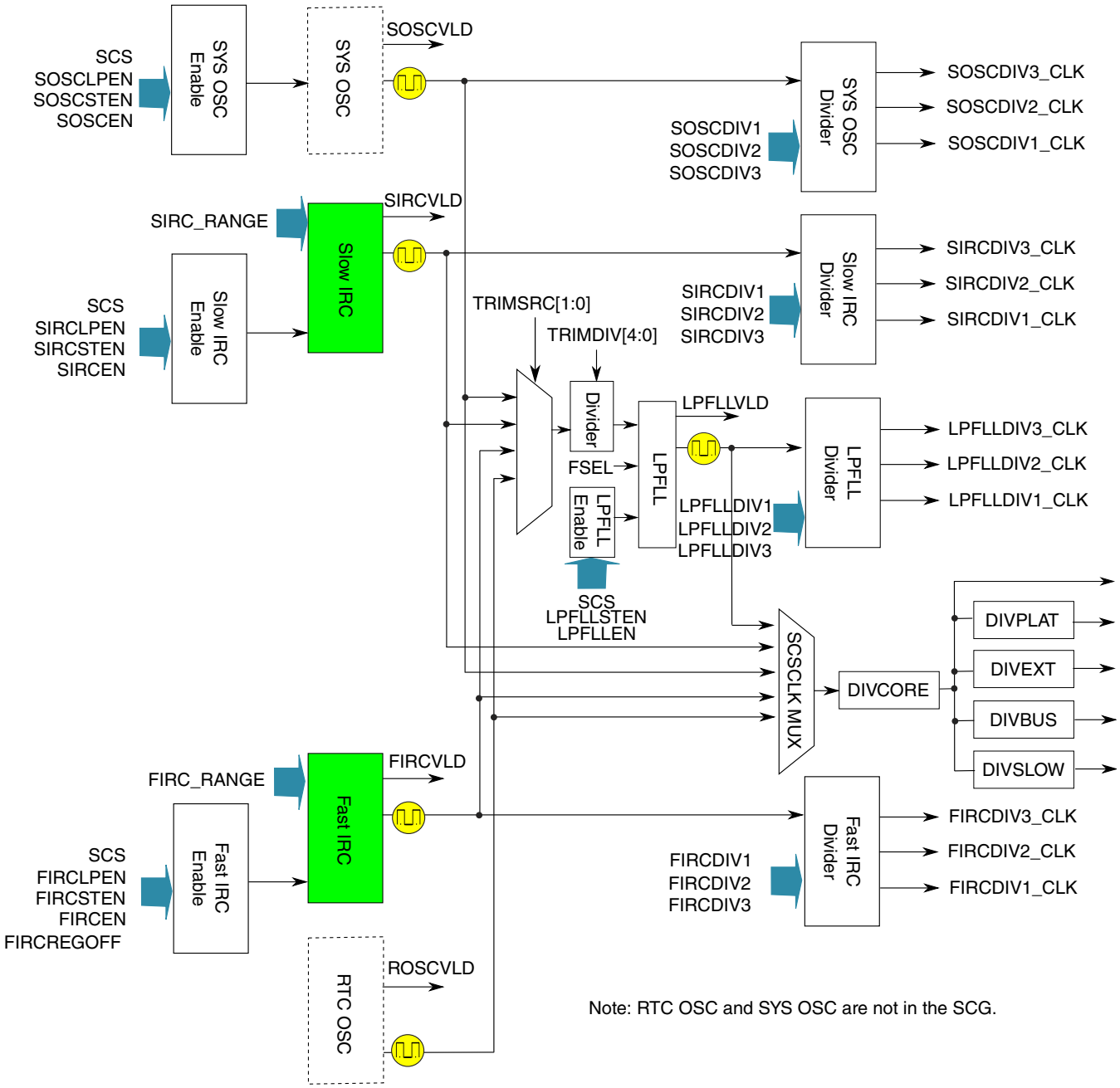



**Figure 29-1. System Clock Generator (SCG) block diagram**

**NOTE**

To identify the oscillator used in your specific MCU device, see chip specific configuration or clocking chapters.

## 29.3 Memory Map/Register Definition

This section includes the memory map and register definition.

The SCG registers can only be written when in supervisor mode. Write accesses when in user mode will result in a bus transfer error. Read accesses may be performed in both supervisor and user mode.

## 29.3.1 SCG register descriptions

**NOTE**

For any writeable SCG registers, only 32-bit writes are allowed. 8-bit or 16-bit writes will result in transfer errors.

### 29.3.1.1 SCG Memory map

SCG base address: 4002_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | B800_003Eh |
| 10h | Clock Status Register (CSR) | 32 | RO | See description. |
| 14h | Run Clock Control Register (RCCR) | 32 | RW | See description. |
| 18h | VLPR Clock Control Register (VCCR) | 32 | RW | See description. |
| 1Ch | HSRUN Clock Control Register (HCCR) | 32 | RW | 0300_0001h |
| 20h | SCG CLKOUT Configuration Register (CLKOUTCNFG) | 32 | RW | 0300_0000h |
| 100h | System OSC Control Status Register (SOSCCSR) | 32 | RW | See description. |
| 104h | System OSC Divide Register (SOSCDIV) | 32 | RW | 0000_0000h |
| 200h | Slow IRC Control Status Register (SIRCCSR) | 32 | RW | 0000_0005h |
| 204h | Slow IRC Divide Register (SIRCDIV) | 32 | RW | 0000_0000h |
| 208h | Slow IRC Configuration Register (SIRCCFG) | 32 | RW | 0000_0001h |
| 300h | Fast IRC Control Status Register (FIRCCSR) | 32 | RW | 0300_0001h |
| 304h | Fast IRC Divide Register (FIRCDIV) | 32 | RW | 0000_0000h |
| 308h | Fast IRC Configuration Register (FIRCCFG) | 32 | RW | 0000_0000h |
| 30Ch | Fast IRC Trim Configuration Register (FIRCTCFG) | 32 | RW | 0000_0000h |
| 318h | Fast IRC Status Register (FIRCSTAT) | 32 | RW | See description. |
| 400h | RTC OSC Control Status Register (ROSCCSR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 500h | Low Power FLL Control Status Register (LPFLLCSR) | 32 | RW | 0000_0000h |
| 504h | Low Power FLL Divide Register (LPFLLDIV) | 32 | RW | 0000_0000h |
| 508h | Low Power FLL Configuration Register (LPFLLCFG) | 32 | RW | 0000_0000h |
| 50Ch | Low Power FLL Trim Configuration Register (LPFLLTCFG) | 32 | RW | 0000_0000h |
| 514h | Low Power FLL Status Register (LPFLLSTAT) | 32 | RW | See description. |

## 29.3.1.2 Version ID Register (VERID)

### 29.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 29.3.1.2.2 Function

Note: Writing to this register will result in a transfer error.

### 29.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VERSION | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VERSION | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 29.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>VERSION | SCG Version Number |

## 29.3.1.3 Parameter Register (PARAM)

### 29.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 29.3.1.3.2 Function

Note: Writing to this register will result in a transfer error.

### 29.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DIVPRES | | | | | 0 | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1[1] | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CLKPRES | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0[2] | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

1. The reset value is controlled by which SCG System Dividers are used by the SoC.
2. The reset value is controlled by which SCG Clock Sources are used by the SoC. Please reference the Reference manual clocking chapter.

### 29.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-27<br>DIVPRES | Divider Present<br>Indicates which system clock dividers are present in this instance of SCG.<br>1xxxxb - System DIVCORE is present.<br>xx1xxb - System DIVEXT is present.<br>xxx1xb - System DIVBUS is present.<br>xxxx1b - System DIVSLOW is present. |
| 26-16<br>— | Reserved |
| 15-8<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-0<br>CLKPRES | Clock Present<br>Indicates which clock sources are present in this instance of SCG. Any bits not defined in this bit field are Reserved and always has the value 0 when read.<br>00000000-00000001b - Reserved.<br>xx1xxxxxb - Low Power FLL (LPFLL) is present.<br>xxx1xxxxb - RTC OSC (ROSC) is present.<br>xxxx1xxxb - Fast IRC (FIRC) is present.<br>xxxxx1xxb - Slow IRC (SIRC) is present.<br>xxxxxx1xb - System OSC (SOSC) is present. |

## 29.3.1.4 Clock Status Register (CSR)

### 29.3.1.4.1 Offset

| Register | Offset |
|---|---|
| CSR | 10h |

### 29.3.1.4.2 Function

This register returns the currently configured system clock source and the system clock dividers for the core (DIVCORE) and peripheral interface clock (DIVSLOW). The SCG_CSR reflects the configuration set by one of three clock control registers SCG_RCCR, SCG_VCCR, SCG_HCCR.

Note: Writing to this register will result in a transfer error.

### 29.3.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SCS | | | | 0 | | | | DIVCORE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | u[1] | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DIVEXT | | | | DIVBUS | | | | DIVSLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

1. The reset value is controlled by user FOPT bits that get uploaded during reset. The two option reset values are div-by-1 or div-by-2 when resetting into RUN mode or div-by-4 or div-by-8 when resetting into VLPR mode

### 29.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>SCS | System Clock Source<br>Returns the currently configured clock source generating the system clock.<br>0000b - Reserved<br>0001b - System OSC (SOSC_CLK)<br>0010b - Slow IRC (SIRC_CLK)<br>0011b - Fast IRC (FIRC_CLK)<br>0100b - RTC OSC (ROSC_CLK)<br>0101b - Low Power FLL (LPFLL_CLK)<br>0110b - Reserved<br>0111b - Reserved |
| 23-20<br>— | Reserved |
| 19-16<br>DIVCORE | Core Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 15-12<br>— | Reserved |
| 11-8<br>DIVEXT | External Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1111b - Divide-by-16 |
| 7-4<br>DIVBUS | Bus Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 3-0<br>DIVSLOW | Slow Clock Divide Ratio<br>0000b - Reserved<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |

## 29.3.1.5 Run Clock Control Register (RCCR)

### 29.3.1.5.1 Offset

| Register | Offset |
|---|---|
| RCCR | 14h |

### 29.3.1.5.2 Function

This register controls the system clock source and the system clock dividers for the core, platform, external and bus clock domains when in Run mode only. This register can only

be written using a 32-bit write. Selecting a different clock source when in RUN requires that clock source to be enabled first and be valid before system clocks switch to that clock source. If system clock divide ratios also change when selecting a different clock mode when in RUN, new system clock divide ratios will not take affect until new clock source is valid.

**NOTE**

Switching to new system clock source and system clock dividers must be done once previous RCCR changes have been completed.

### 29.3.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | SCS | | | 0 | | | | DIVCORE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | u[1] | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DIVEXT | | | | DIVBUS | | | | DIVSLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

1. The reset value is controlled by user FOPT bits that get uploaded during reset. The two option reset values are div-by-1 and div-by-2

### 29.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>SCS | System Clock Source<br>Selects the clock source generating the system clock in Run mode. Attempting to select a clock that is not valid will be ignored. Selecting a different clock source when in Run mode requires that clock source to be enabled first and be valid before system clocks are allowed to switch to that clock source.<br>**NOTE:** Programming SCS to a different value should only be done was previous SCS clock switch has been completed.<br>000b - Reserved<br>001b - System OSC (SOSC_CLK)<br>010b - Slow IRC (SIRC_CLK)<br>011b - Fast IRC (FIRC_CLK)<br>100b - RTC OSC (ROSC_CLK)<br>101b - Low Power FLL (LPFLL_CLK)<br>110b - Reserved<br>111b - Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-20<br>— | Reserved |
| 19-16<br>DIVCORE | Core Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 15-12<br>— | Reserved |
| 11-8<br>DIVEXT | External Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 7-4<br>DIVBUS | Bus Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1111b - Divide-by-16 |
| 3-0<br>DIVSLOW | Slow Clock Divide Ratio<br>0000b - Reserved<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |

## 29.3.1.6 VLPR Clock Control Register (VCCR)

### 29.3.1.6.1 Offset

| Register | Offset |
|---|---|
| VCCR | 18h |

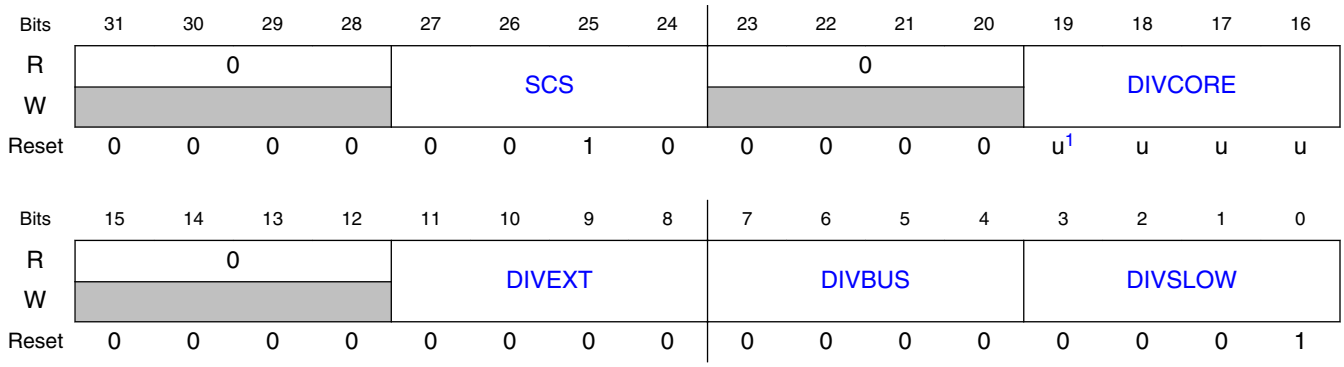
### 29.3.1.6.2 Function

This register controls the system clock source and the system clock dividers for the core, platform, external and bus clock domains when in VLPR mode only. This register can only be written using a 32-bit write. Selecting a different clock source when in VLPR requires that clock source to be enabled first and be valid before system clocks switch to that clock source. If system clock divide ratios also change when selecting a different clock mode when in VLPR, new system clock divide ratios will not take affect until new clock source is valid.

**NOTE**

Switching to new system clock source and system clock dividers must be done once previous RCCR changes have been completed.

### 29.3.1.6.3 Diagram




1. The reset value is controlled by user FOPT bits that get uploaded during reset. The two option reset values are div-by-4 and div-by-8.

### 29.3.1.6.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>SCS | System Clock Source<br>Selects the clock source generating the system clock in VLPR mode. Attempting to select a clock that is not valid will be ignored. Selects the clock source generating the system clock. Selecting a different clock source when in VLPR mode requires that clock source to be enabled first and be valid before system clocks switch to that clock source.<br>0000b - Reserved<br>0001b - System OSC (SOSC_CLK)<br>0010b - Slow IRC (SIRC_CLK)<br>0011b - Reserved<br>0100b - RTC OSC (ROSC_CLK)<br>0101b - Reserved<br>0110b - Reserved<br>0111b - Reserved |
| 23-20<br>— | Reserved |
| 19-16<br>DIVCORE | Core Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 15-12<br>— | Reserved |
| 11-8<br>DIVEXT | External Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 7-4<br>DIVBUS | Bus Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 3-0<br>DIVSLOW | Slow Clock Divide Ratio<br>0000b - Reserved<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |

## 29.3.1.7 HSRUN Clock Control Register (HCCR)

### 29.3.1.7.1 Offset

| Register | Offset |
|---|---|
| HCCR | 1Ch |

### 29.3.1.7.2 Function

This register controls the system clock source and the system clock dividers for the core, platform, external and bus clock domains when in HSRUN mode only. This register can only be written using a 32-bit write. Selecting a different clock source when in HSRUN requires that clock source to be enabled first and be valid before system clocks switch to that clock source. If system clock divide ratios also change when selecting a different clock mode when in HSRUN, new system clock divide ratios will not take affect until new clock source is valid.

**NOTE**

Switching to new system clock source and system clock dividers must be done once previous RCCR changes have been completed.

### 29.3.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SCS | | | | 0 | | | | DIVCORE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DIVEXT | | | | DIVBUS | | | | DIVSLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 29.3.1.7.4 Fields

| Field | Function |
|---|---|
| 31-28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27-24<br>SCS | System Clock Source<br>Selects the clock source generating the system clock in HSRUN mode. Attempting to select a clock that is not valid will be ignored. Selecting a different clock source when in HSRUN mode will enable that clock source and switch to that clock mode when it is valid.<br>0000b - Reserved<br>0001b - System OSC (SOSC_CLK)<br>0010b - Slow IRC (SIRC_CLK)<br>0011b - Fast IRC (FIRC_CLK)<br>0100b - RTC OSC (ROSC_CLK)<br>0101b - Low Power FLL (LPFLL_CLK)<br>0110b - Reserved<br>0111b - Reserved |
| 23-20<br>— | Reserved |
| 19-16<br>DIVCORE | Core Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 15-12<br>— | Reserved |
| 11-8<br>DIVEXT | External Clock Divide Ratio<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 7-4<br>DIVBUS | Bus Clock Divide Ratio<br>0000b - Divide-by-1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 3-0<br>DIVSLOW | Slow Clock Divide Ratio<br>0000b - Reserved<br>0001b - Divide-by-2<br>0010b - Divide-by-3<br>0011b - Divide-by-4<br>0100b - Divide-by-5<br>0101b - Divide-by-6<br>0110b - Divide-by-7<br>0111b - Divide-by-8<br>1000b - Divide-by-9<br>1001b - Divide-by-10<br>1010b - Divide-by-11<br>1011b - Divide-by-12<br>1100b - Divide-by-13<br>1101b - Divide-by-14<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |

## 29.3.1.8 SCG CLKOUT Configuration Register (CLKOUTCNFG)

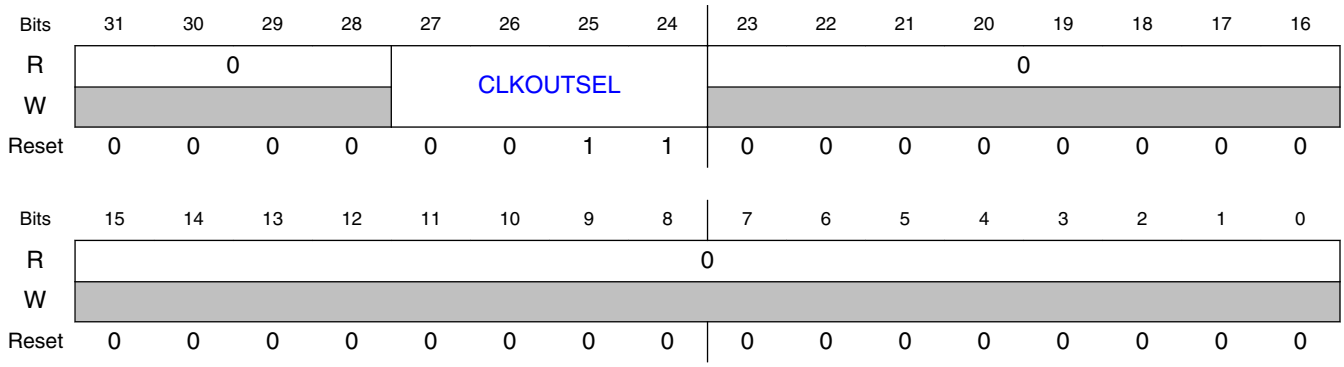
### 29.3.1.8.1 Offset

| Register | Offset |
|---|---|
| CLKOUTCNFG | 20h |

### 29.3.1.8.2 Function

This register controls which SCG clock source is selected to be ported out to the CLKOUT pin.

### 29.3.1.8.3 Diagram




### 29.3.1.8.4 Fields

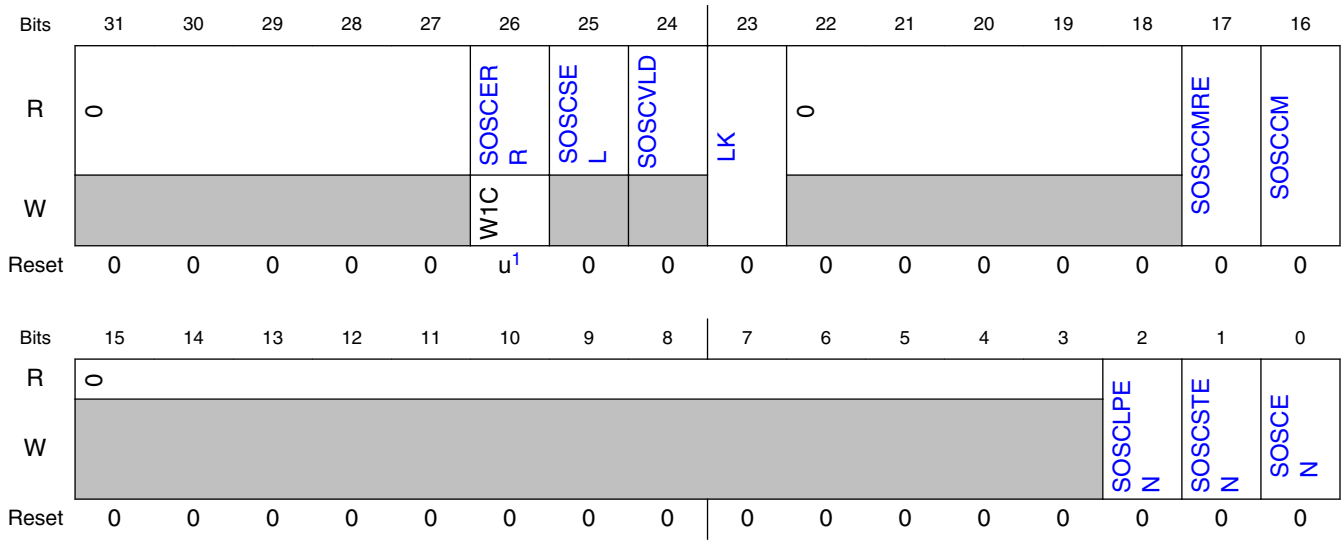
| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>CLKOUTSEL | SCG Clkout Select<br>Selects the SCG system clock.<br>0000b - SCG EXTERNAL Clock<br>0001b - System OSC (SOSC_CLK)<br>0010b - Slow IRC (SIRC_CLK)<br>0011b - Fast IRC (FIRC_CLK)<br>0100b - RTC OSC (ROSC_CLK)<br>0101b - Low Power FLL (LPFLL_CLK)<br>0110b - Reserved<br>0111b - Reserved<br>1111b - Reserved |
| 23-0<br>— | Reserved |

## 29.3.1.9 System OSC Control Status Register (SOSCCSR)

### 29.3.1.9.1 Offset

| Register | Offset |
|---|---|
| SOSCCSR | 100h |

### 29.3.1.9.2 Diagram




1. This flag is reset on Chip POR only

### 29.3.1.9.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>SOSCERR | System OSC Clock Error<br>This flag is reset on Chip POR only, software can also clear this flag by writing a logic one.<br>0b - System OSC Clock Monitor is disabled or has not detected an error<br>1b - System OSC Clock Monitor is enabled and detected an error |
| 25<br>SOSCSEL | System OSC Selected<br>0b - System OSC is not the system clock source<br>1b - System OSC is the system clock source |
| 24<br>SOSCVLD | System OSC Valid<br>0b - System OSC is not enabled or clock is not valid<br>1b - System OSC is enabled and output clock is valid |
| 23<br>LK | Lock Register<br>This bit field can be cleared/set at any time.<br>0b - This Control Status Register can be written.<br>1b - This Control Status Register cannot be written. |
| 22-18<br>— | Reserved |
| 17<br>SOSCCMRE | System OSC Clock Monitor Reset Enable<br>0b - Clock Monitor generates interrupt when error detected<br>1b - Clock Monitor generates reset when error detected |
| 16<br>SOSCCM | System OSC Clock Monitor<br>Enables the clock monitor when SOSCVLD is set. If the clock source is disabled in a low power mode then the clock monitor is also disabled in the low power mode. The clock monitor is always disabled in |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | LLS/VLLS modes. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set following exit from the low power mode.<br>0b - System OSC Clock Monitor is disabled<br>1b - System OSC Clock Monitor is enabled |
| 15-3<br>— | Reserved |
| 2<br>SOSCLPEN | System OSC Low Power Enable<br>SOSCLPEN is required for low power modes. In VLPS mode (low power stop mode), if you want the clock to remain ON, then both SOSCLPEN and SOSCSTEN bits must be enabled.<br>0b - System OSC is disabled in VLP modes<br>1b - System OSC is enabled in VLP modes |
| 1<br>SOSCSTEN | System OSC Stop Enable<br>0b - System OSC is disabled in Stop modes<br>1b - System OSC is enabled in Stop modes if SOSCEN=1. In VLLS0, system oscillator is disabled even if SOSCSTEN=1 and SOSCEN=1. |
| 0<br>SOSCEN | System OSC Enable<br>0b - System OSC is disabled<br>1b - System OSC is enabled |

## 29.3.1.10 System OSC Divide Register (SOSCDIV)

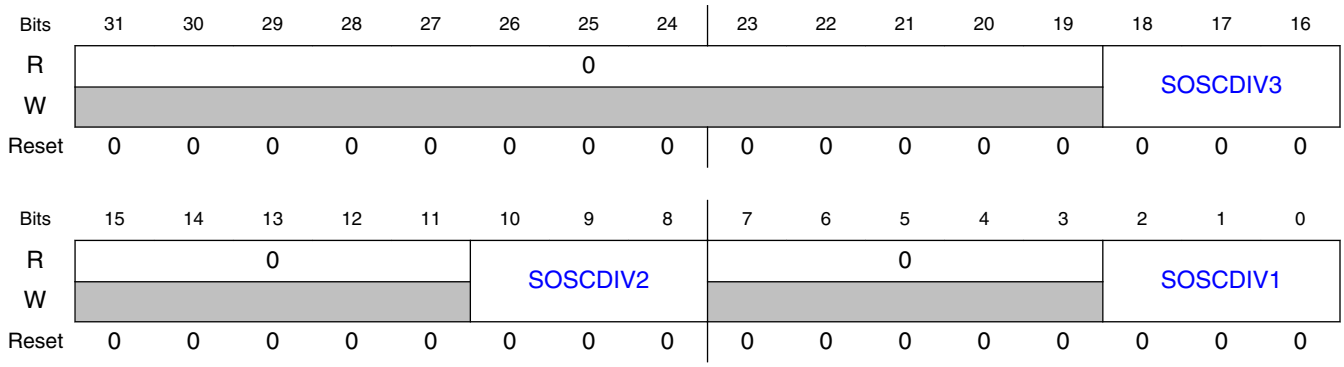
### 29.3.1.10.1 Offset

| Register | Offset |
|---|---|
| SOSCDIV | 104h |

### 29.3.1.10.2 Function

The SCG_SOSCDIV register provides the control of 3 clock trees which can be used to provide optional peripheral functional clocks, or alternative module clocks. Each clock tree has optional dividers of the input SOSC clock. Changes to SOSCDIV should be done when System OSC is disabled to prevent glitches to output divided clock.

### 29.3.1.10.3 Diagram




### 29.3.1.10.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>SOSCDIV3 | System OSC Clock Divide 3<br>Clock divider 3 for System OSC. Used by slow clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 15-11<br>— | Reserved |
| 10-8<br>SOSCDIV2 | System OSC Clock Divide 2<br>Clock divider 2 for System OSC. Used by bus clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 7-3<br>— | Reserved |
| 2-0<br>SOSCDIV1 | System OSC Clock Divide 1<br>Clock divider 1 for System OSC. Used to generate the system clock source. Used by platform clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1 |

| Field | Function |
|---|---|
| | 010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |

## 29.3.1.11 Slow IRC Control Status Register (SIRCCSR)

### 29.3.1.11.1 Offset

| Register | Offset |
|---|---|
| SIRCCSR | 200h |

### 29.3.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | SIRCSEL | SIRCVLD | LK | Reserved | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | 0 | SIRCLPEN | SIRCSTEN | SIRCEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

### 29.3.1.11.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25 | Slow IRC Selected<br>0b - Slow IRC is not the system clock source |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SIRCSEL | 1b - Slow IRC is the system clock source |
| 24<br>SIRCVLD | Slow IRC Valid<br>0b - Slow IRC is not enabled or clock is not valid<br>1b - Slow IRC is enabled and output clock is valid |
| 23<br>LK | Lock Register<br>This bit field can be cleared/set at any time. The purpose for this bitfield is to prevent run away code from changing sirc clock configurations.<br>0b - Control Status Register can be written.<br>1b - Control Status Register cannot be written. |
| 22-4<br>— | This field is reserved and is always has the value 0 |
| 3<br>— | This field is reserved and is always has the value 0 |
| 2<br>SIRCLPEN | Slow IRC Low Power Enable<br>0b - Slow IRC is disabled in VLP modes<br>1b - Slow IRC is enabled in VLP modes |
| 1<br>SIRCSTEN | Slow IRC Stop Enable<br>0b - Slow IRC is disabled in Stop modes<br>1b - Slow IRC is enabled in Stop modes |
| 0<br>SIRCEN | Slow IRC Enable<br>0b - Slow IRC is disabled<br>1b - Slow IRC is enabled |

## 29.3.1.12 Slow IRC Divide Register (SIRCDIV)

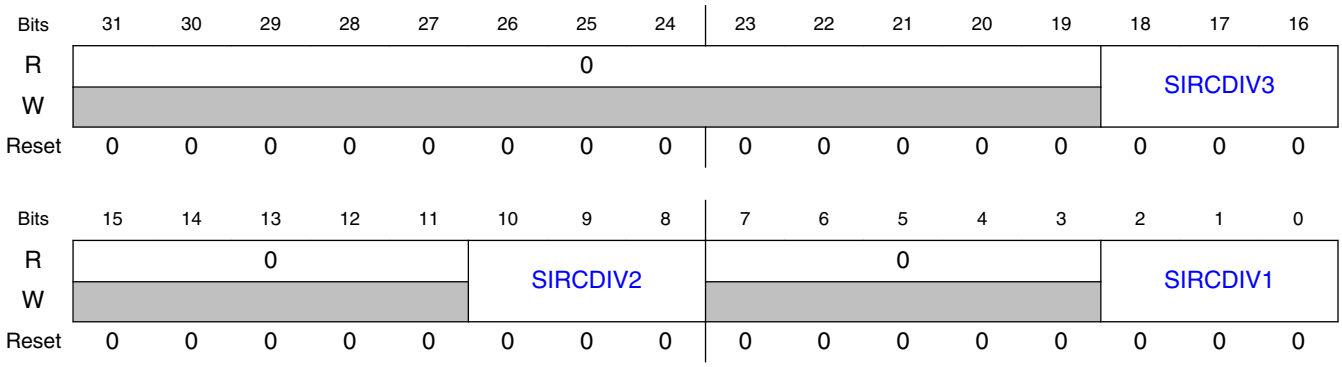
### 29.3.1.12.1 Offset

| Register | Offset |
|---|---|
| SIRCDIV | 204h |

### 29.3.1.12.2 Function

To prevent glitches to the output divided clock, change SIRDIV when the Slow IRC is disabled.

## 29.3.1.12.3 Diagram




## 29.3.1.12.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>SIRCDIV3 | Slow IRC Clock Divider 3<br>Clock divider 3 for Slow IRC. Used by slow clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 15-11<br>— | Reserved |
| 10-8<br>SIRCDIV2 | Slow IRC Clock Divide 2<br>Clock divider 2 for Slow IRC. Used by bus clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 7-3<br>— | Reserved |
| 2-0<br>SIRCDIV1 | Slow IRC Clock Divide 1<br>Clock divider 1 for Slow IRC. Used to generate the system clock source. Used by platform clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1 |

| Field | Function |
|---|---|
| | 010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |

## 29.3.1.13 Slow IRC Configuration Register (SIRCCFG)

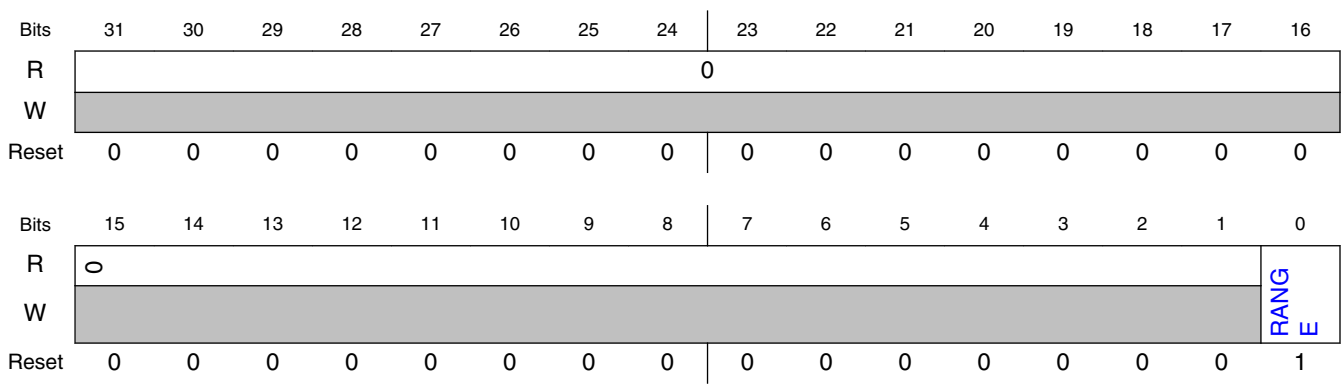
### 29.3.1.13.1 Offset

| Register | Offset |
|---|---|
| SIRCCFG | 208h |

### 29.3.1.13.2 Function

The SIRCCFG register cannot be changed when the slow IRC clock is enabled. When the slow IRC clock is enabled, writes to this register are ignored, and there is no transfer error.

### 29.3.1.13.3 Diagram




### 29.3.1.13.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |

*Table continues on the next page...*

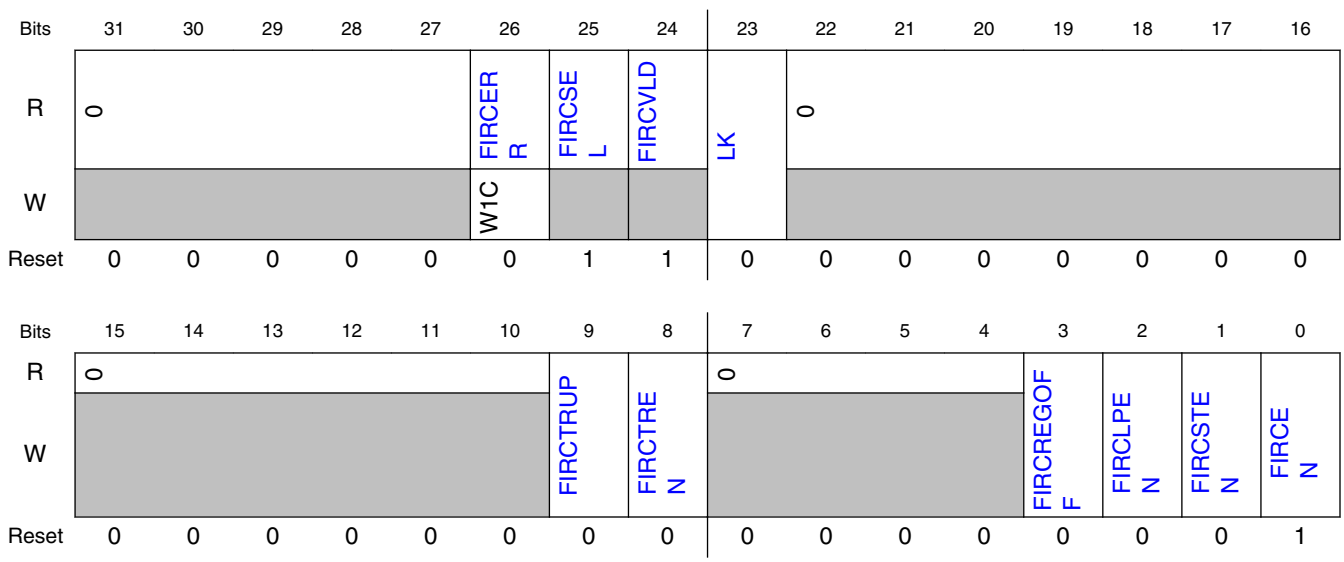
| Field | Function |
|---|---|
| 0<br>RANGE | Frequency Range<br>0b - Slow IRC low range clock (2MHz)<br>1b - Slow IRC high range clock (8 MHz) |

## 29.3.1.14 Fast IRC Control Status Register (FIRCCSR)

### 29.3.1.14.1 Offset

| Register | Offset |
|---|---|
| FIRCCSR | 300h |

### 29.3.1.14.2 Diagram




### 29.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>FIRCERR | Fast IRC Clock Error<br>This flag is reset on Chip POR only, software can also clear this flag by writing a logic one<br>0b - Error not detected with the Fast IRC trimming.<br>1b - Error detected with the Fast IRC trimming. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 25<br>FIRCSEL | Fast IRC Selected status<br>0b - Fast IRC is not the system clock source<br>1b - Fast IRC is the system clock source |
| 24<br>FIRCVLD | Fast IRC Valid status<br>0b - Fast IRC is not enabled or clock is not valid.<br>1b - Fast IRC is enabled and output clock is valid. The clock is valid once there is an output clock from the FIRC analog. |
| 23<br>LK | Lock Register<br>This bit field can be cleared/set at any time.<br>0b - Control Status Register can be written.<br>1b - Control Status Register cannot be written. |
| 22-10<br>— | Reserved |
| 9<br>FIRCTRUP | Fast IRC Trim Update<br>0b - Disable Fast IRC trimming updates<br>1b - Enable Fast IRC trimming updates |
| 8<br>FIRCTREN | Fast IRC Trim Enable<br>0b - Disable trimming Fast IRC to an external clock source<br>1b - Enable trimming Fast IRC to an external clock source |
| 7-4<br>— | Reserved |
| 3<br>FIRCREGOFF | Fast IRC Regulator Enable<br>0b - Fast IRC Regulator is enabled.<br>1b - Fast IRC Regulator is disabled. |
| 2<br>FIRCLPEN | Fast IRC Low Power Enable<br>0b - Fast IRC is disabled in VLP modes<br>1b - Fast IRC is enabled in VLP modes |
| 1<br>FIRCSTEN | Fast IRC Stop Enable<br>0b - Fast IRC is disabled in Stop modes.<br>1b - Fast IRC is enabled in Stop modes |
| 0<br>FIRCEN | Fast IRC Enable<br>0b - Fast IRC is disabled<br>1b - Fast IRC is enabled |

## 29.3.1.15 Fast IRC Divide Register (FIRCDIV)

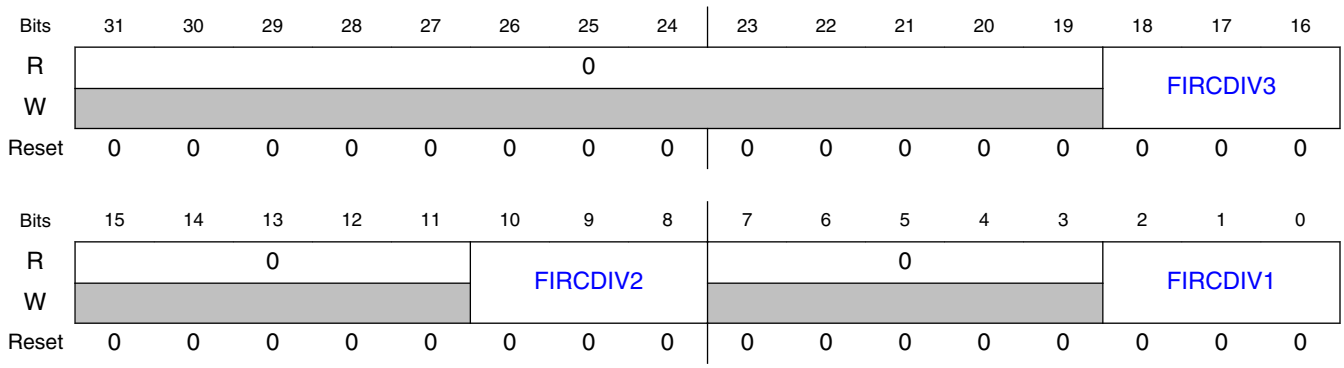
### 29.3.1.15.1 Offset

| Register | Offset |
|---|---|
| FIRCDIV | 304h |

### 29.3.1.15.2 Function

Changes to FIRCDIV should be done when FAST IRC is disabled to prevent glitches to output divided clock.

### 29.3.1.15.3 Diagram




### 29.3.1.15.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>FIRCDIV3 | Fast IRC Clock Divider 3<br>Clock divider 3 for the Fast IRC. Used by slow clock modules that need an asynchronous clock source.<br>000b - Clock disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 15-11<br>— | Reserved |
| 10-8<br>FIRCDIV2 | Fast IRC Clock Divide 2<br>Clock divider 2 for the Fast IRC. Used by bus clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 7-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>FIRCDIV1 | Fast IRC Clock Divide 1<br>Clock divider 1 for Fast IRC. Used to generate the system clock source. Used by platform clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |

## 29.3.1.16 Fast IRC Configuration Register (FIRCCFG)

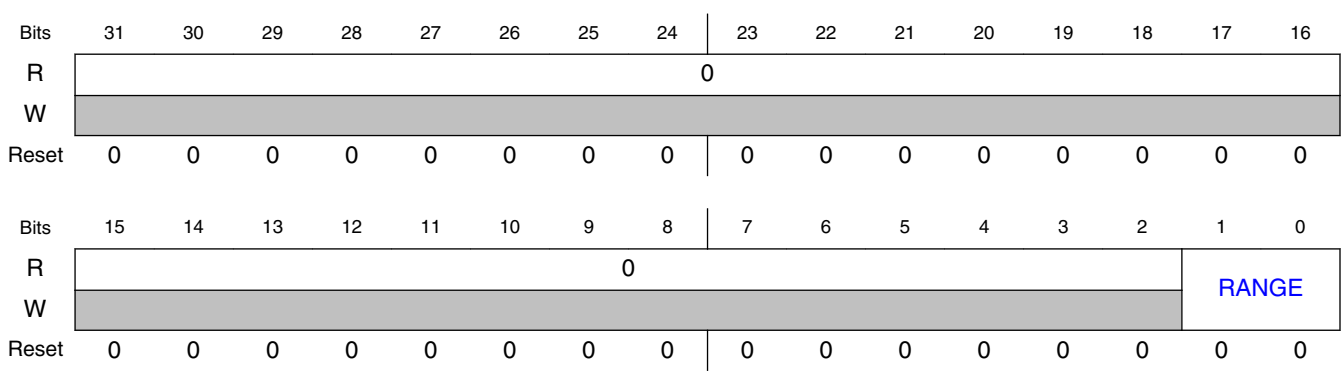
### 29.3.1.16.1 Offset

| Register | Offset |
|---|---|
| FIRCCFG | 308h |

### 29.3.1.16.2 Function

The FIRCCFG register cannot be changed when the Fast IRC is enabled. When the Fast IRC is enabled, writes to this register are ignored, and there is no transfer error.

### 29.3.1.16.3 Diagram



#### 29.3.1.16.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>RANGE | Frequency Range<br>00b - Fast IRC is trimmed to 48 MHz<br>01b - Fast IRC is trimmed to 52 MHz<br>10b - Fast IRC is trimmed to 56 MHz<br>11b - Fast IRC is trimmed to 60 MHz |

### 29.3.1.17 Fast IRC Trim Configuration Register (FIRCTCFG)

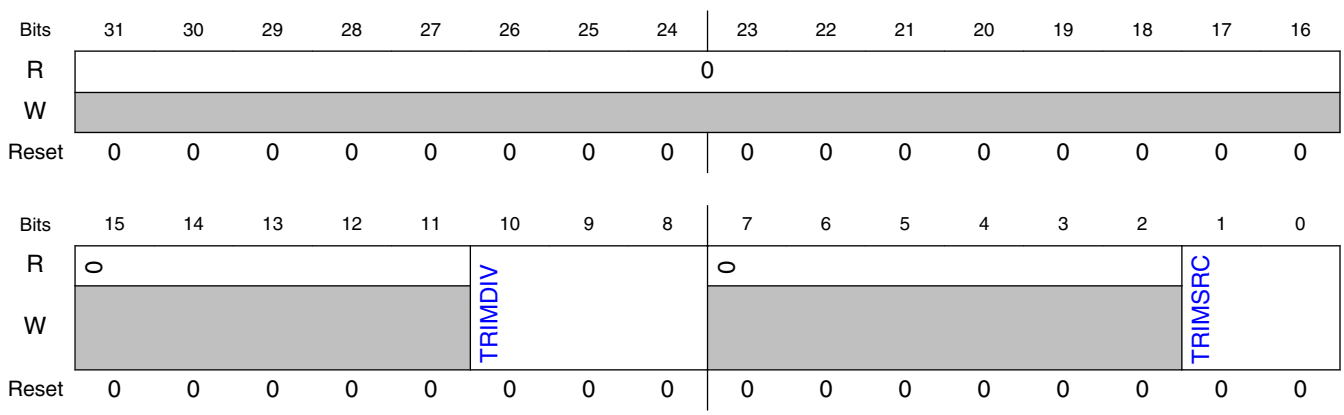
#### 29.3.1.17.1 Offset

| Register | Offset |
|---|---|
| FIRCTCFG | 30Ch |

#### 29.3.1.17.2 Function

The FIRCTCFG register cannot be changed when Fast IRC tuning is enabled. When the Fast IRC tuning is enabled, writes to this register are ignored, and there is no transfer error.

#### 29.3.1.17.3 Diagram



### 29.3.1.17.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-11<br>— | Reserved |
| 10-8<br>TRIMDIV | Fast IRC Trim Predivide<br>Divide the System OSC down for Fast IRC trimming.<br>000b - Divide by 1<br>001b - Divide by 128<br>010b - Divide by 256<br>011b - Divide by 512<br>100b - Divide by 1024<br>101b - Divide by 2048<br>110b - Reserved. Writing this value will result in Divide by 1.<br>111b - Reserved. Writing this value will result in a Divide by 1. |
| 7-2<br>— | Reserved |
| 1-0<br>TRIMSRC | Trim Source<br>Configures the external clock source to tune the Fast IRC. TRIMSRC must be configured before programming FIRCSTAT register for trim update<br>00b - Reserved<br>01b - Reserved<br>10b - System OSC. This option requires that SOSC be divided using the TRIMDIV field to get a frequency slower than 32kHz.<br>11b - RTC OSC (32.768 kHz) |

## 29.3.1.18 Fast IRC Status Register (FIRCSTAT)

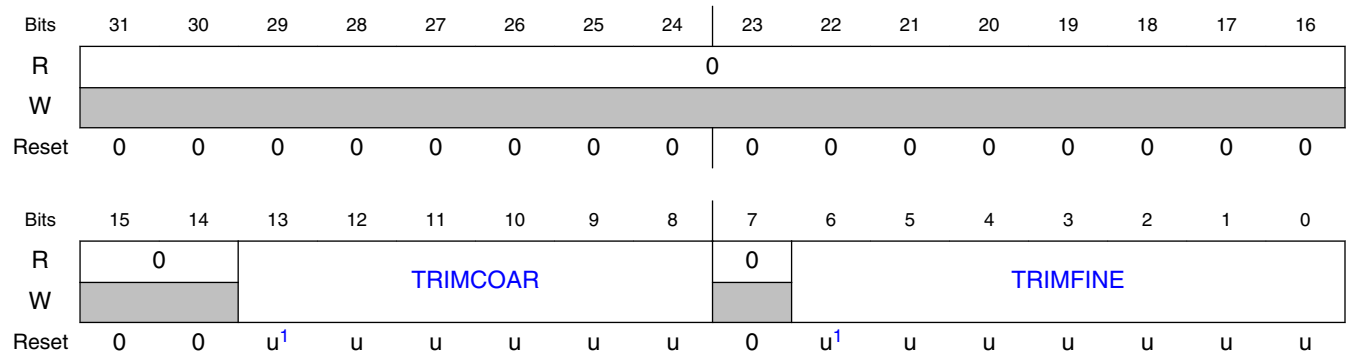
### 29.3.1.18.1 Offset

| Register | Offset |
|---|---|
| FIRCSTAT | 318h |

### 29.3.1.18.2 Function

This register is loaded from IFR during reset. This register is uploaded with the trim values generated by FIRC auto trimming which is enabled when FIRC is enabled and FIRCTREN=1 and FIRCTRUP=1. When FIRC auto trimming is enabled and FIRCTRUP is off (Note: TRIMSRC needs to be programmed to TRIMSRC=10 or TRIMSRC=11), writes to this register are allowed and values written to this register are used to trim FIRC clock.

### 29.3.1.18.3 Diagram




1. Reset values are loaded out of IFR.

### 29.3.1.18.4 Fields

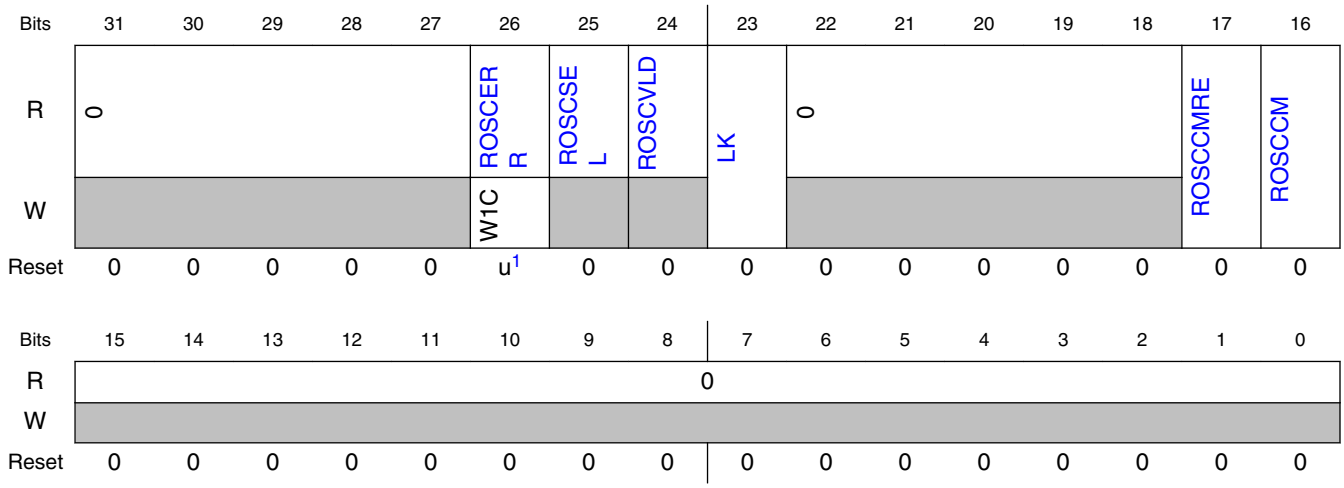
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13-8<br>TRIMCOAR | Trim Coarse<br>TRIMCOAR bits are used to coarsely trim the Fast IRC Clock to within approximately ±0.7% of the target frequency. When FIRC is enabled and auto trimming is enabled (FIRCTREN=1 and FIRCTRUP=1), then TRIMCOAR register gets uploaded with the trimmed coarse value. When FIRCTRUP=0, TRIMCOAR register is writable, to allow user programming of coarse trim values. |
| 7<br>— | Reserved |
| 6-0<br>TRIMFINE | Trim Fine Status<br>Once the Fast IRC Clock is trimmed to ±0.7% of the target frequency using the TRIMCOAR bits, the TRIMFINE bits can be used to trim the Fast IRC Clock to within ±0.04% of the target frequency. When FIRC is enabled and auto trimming is enabled (FIRCTREN=1 and FIRCTRUP=1), TRIMFINE register gets uploaded with the trimmed fine value. When FIRCTRUP=0, TRIMFINE register is writeable, to allow user programming of fine trim values. |

## 29.3.1.19 RTC OSC Control Status Register (ROSCCSR)

### 29.3.1.19.1 Offset

| Register | Offset |
|---|---|
| ROSCCSR | 400h |

## 29.3.1.19.2 Diagram




1. Reset values are loaded out of IFR.

## 29.3.1.19.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>ROSCERR | RTC OSC Clock Error<br>This flag is reset on Chip POR only, software can also clear this flag by writing a logic one<br>0b - RTC OSC Clock Monitor is disabled or has not detected an error<br>1b - RTC OSC Clock Monitor is enabled and detected an RTC loss of clock error |
| 25<br>ROSCSEL | RTC OSC Selected<br>0b - RTC OSC is not the system clock source<br>1b - RTC OSC is the system clock source |
| 24<br>ROSCVLD | RTC OSC Valid<br>0b - RTC OSC is not enabled or clock is not valid<br>1b - RTC OSC is enabled and output clock is valid |
| 23<br>LK | Lock Register<br>This bit field can be cleared/set at any time.<br>0b - Control Status Register can be written.<br>1b - Control Status Register cannot be written. |
| 22-18<br>— | Reserved |
| 17<br>ROSCCMRE | RTC OSC Clock Monitor Reset Enable<br>0b - Clock Monitor generates interrupt when error detected<br>1b - Clock Monitor generates reset when error detected |
| 16<br>ROSCCM | RTC OSC Clock Monitor |

*Table continues on the next page...*

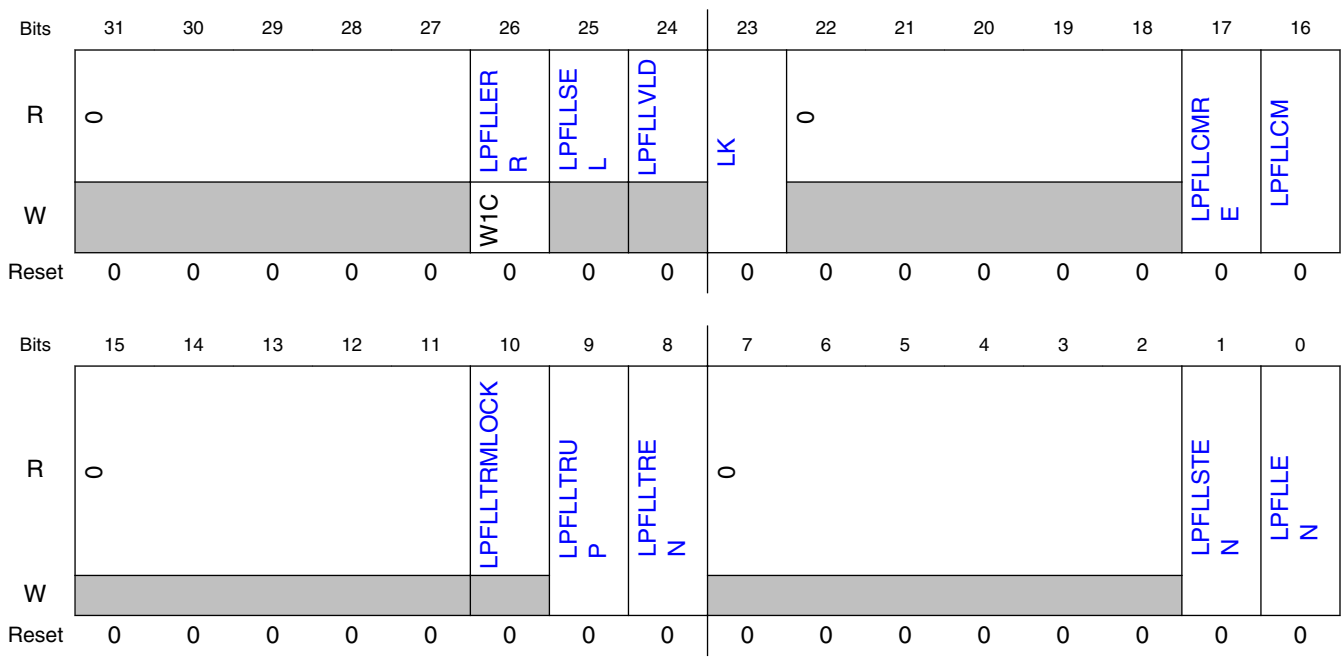
| Field | Function |
|---|---|
| | Enables the clock monitor when ROSCVLD is set. ROSC clock monitor remains enabled in VLPR and VLPS if ROSCCM is enabled. The clock monitor is always disabled in LLS/VLLS modes. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set following exit from the low power mode.<br>0b - RTC OSC Clock Monitor is disabled<br>1b - RTC OSC Clock Monitor is enabled |
| 15-0<br>— | Reserved |

## 29.3.1.20 Low Power FLL Control Status Register (LPFLLCSR)

### 29.3.1.20.1 Offset

| Register | Offset |
|---|---|
| LPFLLCSR | 500h |

### 29.3.1.20.2 Diagram



### 29.3.1.20.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>LPFLLERR | LPFLL Clock Error<br>This flag is reset on Chip POR only, software can also clear this flag by writing a logic one<br>When LPFLLTREN=1 and LPFLLTRUP=1, LPFLLERR=1 if the LPFLL can't lock the reference clock. This occurs when the reference clock is too fast/slow or LPFLL clock is stopped. LPFLLERR indicates a loss of lock or loss of clock.<br>To change the reference clock frequency to re-lock, the LPFLLTREN or LPFLLTRUP bits must also be re-enabled (LPFLLTREN=1 or LPFLLTRUP=1).<br>0b - Error not detected with the LPFLL trimming.<br>1b - Error detected with the LPFLL trimming. |
| 25<br>LPFLLSEL | LPFLL Selected<br>0b - LPFLL is not the system clock source<br>1b - LPFLL is the system clock source |
| 24<br>LPFLLVLD | LPFLL Valid<br>0b - LPFLL is not enabled or clock is not valid.<br>1b - LPFLL is enabled and output clock is valid. |
| 23<br>LK | Lock Register<br>This bit field can be cleared/set at any time.<br>0b - Control Status Register can be written.<br>1b - Control Status Register cannot be written. |
| 22-18<br>— | Reserved |
| 17<br>LPFLLCMRE | LPFLL Clock Monitor Reset Enable<br>0b - Clock Monitor generates interrupt when error detected<br>1b - Clock Monitor generates reset when error detected |
| 16<br>LPFLLCM | LPFLL Clock Monitor<br>Enables the clock monitor when LPFLLTREN is set and LPFLL is enabled. The clock monitor is always disabled in low power modes. When the clock monitor is disabled in a low power mode, it remains disabled until the clock valid flag is set following exit from the low power mode.<br>0b - LPFLL Clock Monitor is disabled<br>1b - LPFLL Clock Monitor is enabled |
| 15-11<br>— | Reserved |
| 10<br>LPFLLTRMLOCK | LPFLL Trim LOCK<br>Asserts only when LPFLLTREN=1 and LPFLLTRUP=1 and LPFLL has locked to target frequency.<br>In open-loop mode (LPFLLTRUP=0), lock conditions cannot be checked.<br>0b - LPFLL not Locked<br>1b - LPFLL trimmed and Locked |
| 9<br>LPFLLTRUP | LPFLL Trim Update<br>0b - Disable LPFLL trimming updates. LPFLL frequency determined by AUTOTRIM written value.<br>1b - Enable LPFLL trimming updates. LPFLL frequency determined by reference clock multiplication |
| 8 | LPFLL Trim Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LPFLLTREN | 0b - Disable trimming LPFLL to an reference clock source<br>1b - Enable trimming LPFLL to an reference clock source |
| 7-2<br>— | Reserved |
| 1<br>LPFLLSTEN | LPFLL Stop Enable<br>0b - LPFLL is disabled in Stop modes.<br>1b - LPFLL is enabled in Stop modes |
| 0<br>LPFLLEN | LPFLL Enable<br>0b - LPFLL is disabled<br>1b - LPFLL is enabled |

## 29.3.1.21 Low Power FLL Divide Register (LPFLLDIV)

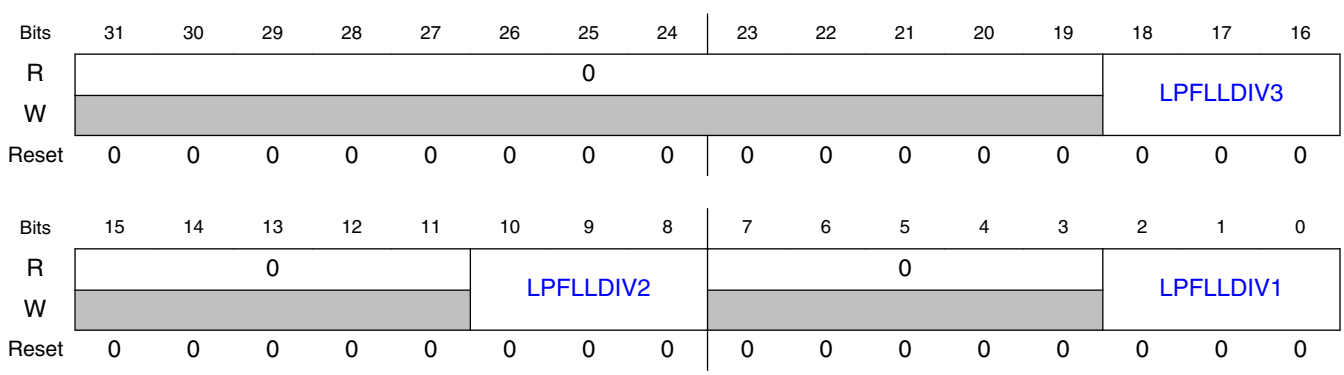
### 29.3.1.21.1 Offset

| Register | Offset |
|---|---|
| LPFLLDIV | 504h |

### 29.3.1.21.2 Function

Changes to LPFLLDIV should be done when LPFLL is disabled to prevent glitches to output divided clock.

### 29.3.1.21.3 Diagram



### 29.3.1.21.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>LPFLLDIV3 | LPFLL Clock Divide 3<br>Clock divider 3 for the LPFLL. Used by slow clock modules that need an asynchronous clock source.<br>000b - Clock disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 15-11<br>— | Reserved |
| 10-8<br>LPFLLDIV2 | LPFLL Clock Divide 2<br>Clock divider 2 for the LPFLL. Used by bus clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |
| 7-3<br>— | Reserved |
| 2-0<br>LPFLLDIV1 | LPFLL Clock Divide 1<br>Clock divider 1 for LPFLL. Used to generate the system clock source. Used by platform clock modules that need an asynchronous clock source.<br>000b - Output disabled<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 4<br>100b - Divide by 8<br>101b - Divide by 16<br>110b - Divide by 32<br>111b - Divide by 64 |

## 29.3.1.22 Low Power FLL Configuration Register (LPFLLCFG)

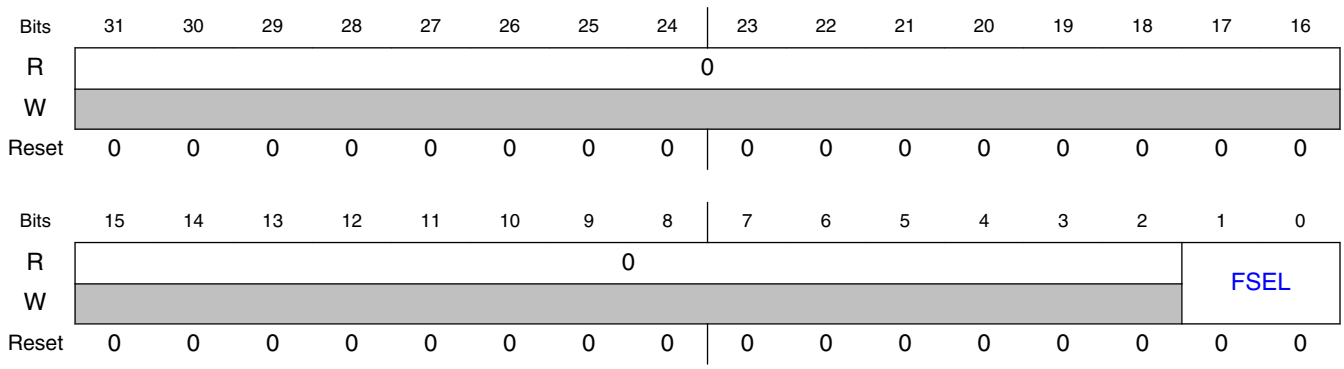
### 29.3.1.22.1 Offset

| Register | Offset |
|---|---|
| LPFLLCFG | 508h |

### 29.3.1.22.2 Function

The LPFLLCFG register cannot be changed when the LPFLL is enabled. When the LPFLL is enabled, writes to this register are ignored, and there is no transfer error.

### 29.3.1.22.3 Diagram




### 29.3.1.22.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>FSEL | Frequency Select<br>00b - LPFLL is trimmed to 48 MHz.<br>01b - LPFLL is trimmed to 72 MHz.<br>10b - Reserved<br>11b - Reserved |

## 29.3.1.23 Low Power FLL Trim Configuration Register (LPFLLTCFG)

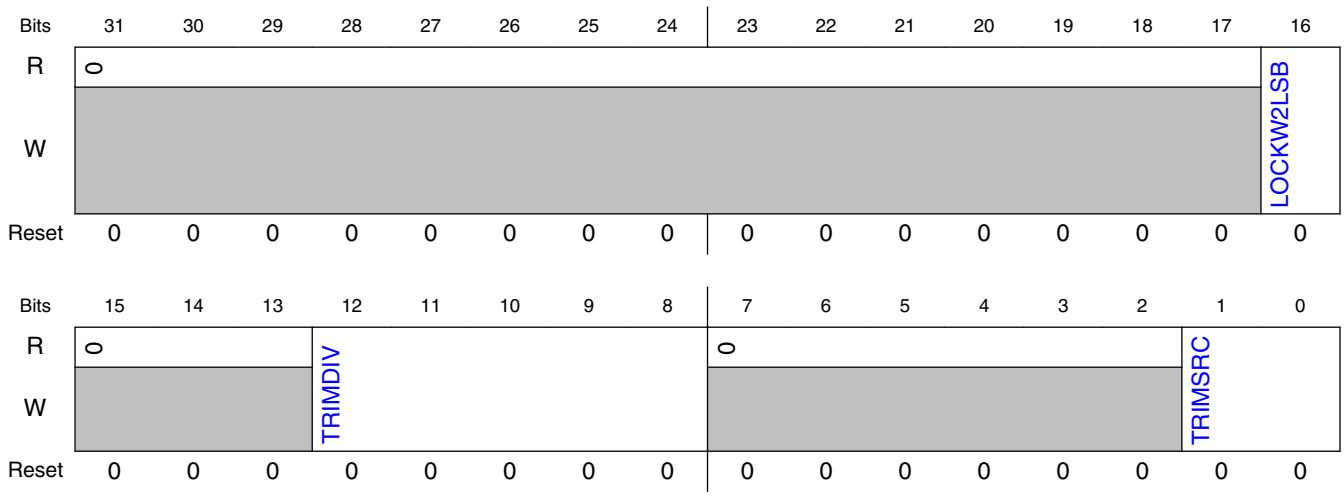
### 29.3.1.23.1 Offset

| Register | Offset |
|---|---|
| LPFLLTCFG | 50Ch |

### 29.3.1.23.2 Function

The LPFLLTCFG register cannot be changed when LPFLL tuning is enabled. When the LPFLL tuning is enabled, writes to this register are ignored, and there is no transfer error.

### 29.3.1.23.3 Diagram




### 29.3.1.23.4 Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved |
| 16<br>LOCKW2LSB | Lock LPFLL with 2 LSBS<br>This bitfield is used to control the condition to set LPFLLTRMLOCK: difference between LPFLL actual clock and target clock (48 MHz, 72 MHz) is within 0.8% or 0.4%;<br>0b - LPFLL locks within 1LSB (0.4%)<br>1b - LPFLL locks within 2LSB (0.8%) |
| 15-13<br>— | Reserved |
| 12-8<br>TRIMDIV | LPFLL Trim Predivide<br>Use to divide the reference clock down for LPFLL trimming by 1,2,4,8,....31,32. The divided frequency should be either 32.768 KHz or 2 MHz.<br>00000 Divide by 1<br>00001 Divide by 2<br>00010 Divide by 3<br>....<br>11110 Divide by 31<br>11111 Divide by 32 |
| 7-2 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 1-0<br>TRIMSRC | Trim Source<br>Configures the reference clock source to tune the LPFLL.<br>00b - SIRC<br>01b - FIRC<br>10b - System OSC<br>11b - RTC OSC |

## 29.3.1.24 Low Power FLL Status Register (LPFLLSTAT)

### 29.3.1.24.1 Offset

| Register | Offset |
|---|---|
| LPFLLSTAT | 514h |

### 29.3.1.24.2 Function

This register is loaded from IFR during reset. These register gets uploaded with the trim values generated by LPFLL auto trimming which is enabled when LPFLL is enabled and LPFLLTREN=1 and LPFLLTRUP=1. When LPFLL auto trimming is enabled and LPFLLTRUP is off, writes to this register are used to trim LPFLL clock.

### 29.3.1.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | AUTOTRIM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u | u | u | u | u |

1. Reset values are loaded out of IFR.

#### 29.3.1.24.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>AUTOTRIM | Auto Tune Trim Status<br>When LPFLL is enabled and auto trimming is enabled (LPFLLTREN=1 and LPFLLTRUP=1) these register gets uploaded with the trimmed value. When LPFLLTRUP=0, these register is writeable to allow user programming of trim values. |

## 29.4 Functional description

### 29.4.1 SCG Clock Mode Transitions

The following figure shows the valid clock mode transitions supported by SCG.

Slow IRC (SIRC) boot mode is not supported on this device.

## SCG Valid Mode Transitions

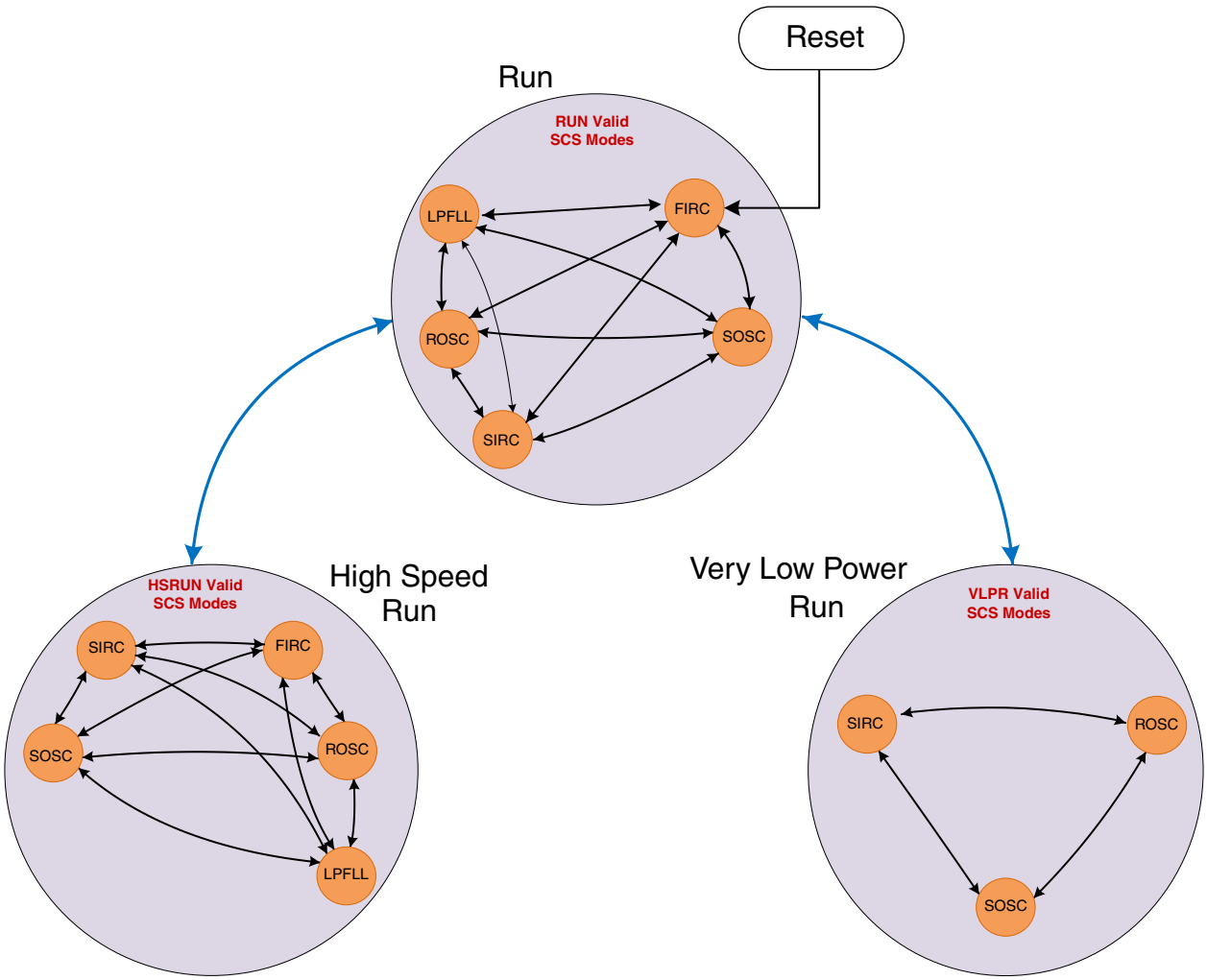



**Figure 29-2. SCG Valid Mode Transition Diagram**

The SCG will restrict programming into invalid clock modes and writes to SCS bits will be ignored. When a transition between run modes or a transition into wait mode occurs, the SCG completes the switch to the clock mode as defined in the SCG clock control register. When completed, the system switches to the selected run/wait mode.

The modes of operation listed in the following table are the valid modes for this implementation of the SCG.

## Table 29-1. SCG modes of operation

| Mode | Description |
|---|---|
| System Oscillator Clock (SOSC) | System Oscillator Clock (SOSC) mode is entered when all the following conditions occur:<br>• RUN MODE: 0001 is written to RCCR[SCS].<br>VLRUN MODE: 0001 is written to VCCR[SCS].<br>HSRUN MODE: 0001 is written to HCCR[SCS].<br>• SOSCEN = 1<br>• SOSCVLD = 1<br><br>In SOSC mode, SCSCLKOUT and system clocks are derived from the external System Oscillator Clock (SOSC). |
| Slow Internal Reference Clock (SIRC) | Slow Internal Reference Clock (SIRC) mode is entered when all the following conditions occur:<br>• RUN MODE: 0010 is written to RCCR[SCS].<br>VLRUN MODE: 0010 is written to VCCR[SCS] and 1 is written to SIRCCSR[SIRCLPEN].<br>HSRUN MODE: 0010 is written to HCCR[SCS].<br>• SIRCEN = 1<br>• SIRCVLD = 1<br><br>In SIRC mode, SCSCLKOUT and system clocks are derived from the slow internal reference clock. Two frequency ranges are available for SIRC clock as described in the SIRCCFG[RANGE] register definition. Changes to SIRC range settings will be ignored when SIRC clock is enabled. |
| Fast Internal Reference Clock (FIRC) | Fast Internal Reference Clock (FIRC) mode is the default clock mode of operation and is entered when all the following conditions occur:<br>• RUN MODE: 0011 is written to RCCR[SCS].<br>VLRUN MODE: Invalid mode. Programming SCG into FIRC mode will be ingored.<br>HSRUN MODE: 0011 is written to HCCR[SCS].<br>• FIRCEN = 1<br>• FIRCVLD = 1<br><br>In FIRC mode, SCSCLKOUT and system clocks are derived from the fast internal reference clock. Two frequency range settings are available for FIRC clock as described in the FIRC[RANGE] register definition. Changes to FIRC range settings will be ignored when FIRC clock is enabled. |
| Real Time Clock Oscillator (RTCOSC) | Real Time Clock Oscillator (RTCOSC) mode is entered when all the following conditions occur:<br>• RUN MODE: 0100 is written to RCCR[SCS].<br>VLRUN MODE: 0100 is written to VCCR[SCS].<br>HSRUN MODE: 0100 is written to HCCR[SCS].<br>• ROSCVLD = 1<br><br>In RTCOSC mode, SCSCLKOUT and systems clocks are derived from the external RTC Oscillator Clock (RTCOSC). |
| Low Power FLL (LPFLL) | Low Power FLL (LPFLL) mode is entered when all the following conditions occur:<br>• RUN MODE: 0101 is written to RCCR[SCS].<br>VLRUN MODE: Invalid mode. Programming SCG into LPFLL mode will be ingored. |

*Table continues on the next page...*

## Table 29-1. SCG modes of operation (continued)

| Mode | Description |
|---|---|
| | HSRUN MODE: 0101 is written to HCCR[SCS].<br>• LPFLLEN = 1<br>• LPFLLVLD = 1<br><br>In LPFLL mode, SCSCLKOUT and system clocks are derived from the Low Power FLL (LPFLL). By default the LPFLL will be running in open-loop mode using default trim values. In closed-loop mode (LPFLLTREN=1 and LPFLLTRUP=1) LPFLL will be auto trimmed using a selectable reference clock as specified by its corresponding SCG_LPFLLTCFG[TRIMSRC]. The LPFLL will lock its frequency to the LPFLL factor, as specified by the SCG_LPFLLCFG[FSEL]. |
| Stop | Entered whenever the MCU enters a Stop state. The power modes are chip specific. For power mode assignments, see the chapter that describes how modules are configured and SCG behaviour during Stop recovery. Entering Stop mode, all SCG clock signals are static including SCG system clocks (DIVCORE,,DIVEXT,DIVBUS,DIVSLOW).<br><br>There are exceptions - the following clocks can continue to run and stay enabled in the following cases:<br><br>SIRC is available in Normal Stop and VLPS mode when all the following conditions become true:<br>• SIRCCSR[SIRCEN] = 1<br>• SIRCCSR[SIRCSTEN] = 1<br>• SIRCCSR[SIRCLPEN] = 1 in VLPS<br><br>FIRC is available only in Normal Stop mode when all the following conditions become true:<br>• FIRCCSR[FIRCEN] = 1<br>• FIRCCSR[FIRCSTEN] = 1<br><br>SOSC is available in following low power stop modes (Normal Stop, VLPS, LLS) when all the below conditions are true. In VLLS stop mode, SOSCLK is disabled.<br>• SOSCCSR[SOSCEN] = 1<br>• SOSCCSR[SOSCSTEN] = 1<br>• SOSCCSR[SOSCLPEN] = 1 (required only for Low Power Stop modes (VLPS and LLS)<br><br>LPFLL is available in Normal Stop mode when all the following conditions become true:<br>• LPFLLCSR[LPFLLEN] = 1<br>• LPFLLCSR[LPFLLSTEN] = 1 |

# Chapter 30
# Peripheral Clock Controller (PCC)

## 30.1 Chip-specific PCC information

### 30.1.1 Peripheral source to the PCC

The PCC module has the following peripheral clock sources as input:

- Clock option 0: Clock is off / External clock selected (if available)

  Only LPADC and I2S support external clock:

  - The external clock to I2S is I2S0_MCLK from PTB5 or PTE19.
  - The external clock to LPADC is the output of LPADCIRC.
- Clock option 1: System OSC
- Clock option 2: SIRC
- Clock option 3: FIRC
- Clock option 4: Reserved
- Clock option 5: Reserved
- Clock option 6: LPFLL
- Clock option 7: Reserved

## 30.2 Introduction

The Peripheral Clock Control (PCC) module provides clock control and configuration for on-chip peripherals. Each peripheral has its own clock control and configuration register.

## 30.3 Features

The PCC module enables software to configure the following clocking options for each peripheral:

- Interface clock gating
- Functional clock source selection
- Functional clock divide values

Below is a block diagram of the PCC module:

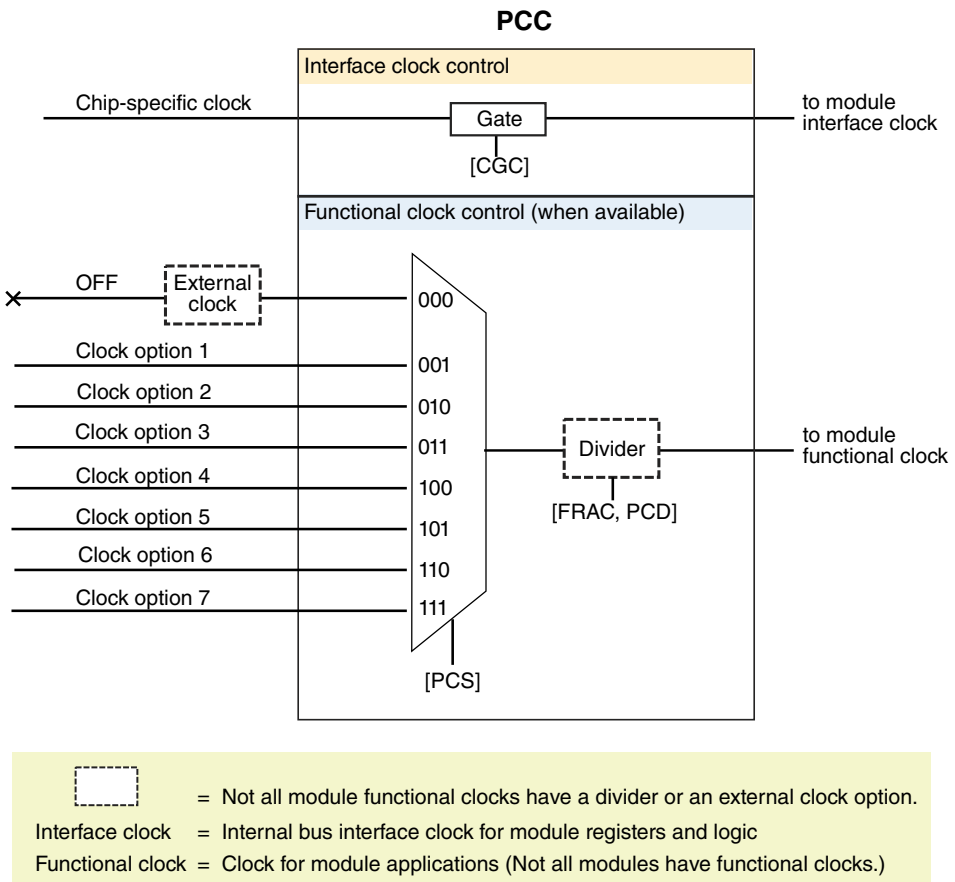



**Figure 30-1. PCC Block Diagram**

## 30.4 Functional description

The PCC module provides on-chip peripherals (modules) their own dedicated PCC registers for clock gating and configuration options. Each module's PCC register contains a clock gating control bit (CGC) for the module's interface clock. Before a module can be used, its interface clock must be enabled (CGC = 1) in the module's PCC register.

If a module has a functional clock, its PCC register may provide options for the clock source, selected by programming the Peripheral Clock Select (PCS) field. Optionally, a module may also have a clock divider, selected by programming the Peripheral Clock Divider (PCD) field along with a Fraction (FRAC) field. Before configuring a functional clock, the module's interface clock must be disabled (CGC = 0).

## 30.5 Memory map and register definition

Each module has its own dedicated PCC register, which controls the clock gating, clock source and divider (when applicable) for that specific module. See each module's PCC register for details.

PCC registers can be written only in supervisor mode using 32-bit accesses.

**NOTE**

To configure the clocking options available to a given module or to modify an existing configuration, first disable the module's interface clock by writing 0 to its CGC bit.

### 30.5.1 PCC0 register descriptions

#### 30.5.1.1 PCC0 Memory map

PCC0 base address: 4002_B000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | PCC MSCM Register (PCC_MSCM) | 32 | RW | 8000_0000h |
| 10h | PCC AXBS0 Register (PCC_AXBS0) | 32 | RW | C000_0000h |
| 20h | PCC DMA0 Register (PCC_DMA0) | 32 | RW | 8000_0000h |
| 30h | PCC FLEXBUS Register (PCC_FLEXBUS) | 32 | RW | 8000_0000h |
| 50h | PCC XRDC_MGR Register (PCC_XRDC_MGR) | 32 | RW | 8000_0000h |
| 58h | PCC XRDC_PAC Register (PCC_XRDC_PAC) | 32 | RW | 8000_0000h |
| 5Ch | PCC XRDC_MRC Register (PCC_XRDC_MRC) | 32 | RW | 8000_0000h |
| 6Ch | PCC SEMA42_0 Register (PCC_SEMA42_0) | 32 | RW | 8000_0000h |
| 84h | PCC DMAMUX0 Register (PCC_DMAMUX0) | 32 | RW | 8000_0000h |
| 88h | PCC EWM Register (PCC_EWM) | 32 | RW | 8000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 94h | PCC MUA Register (PCC_MUA) | 32 | RW | 8000_0000h |
| BCh | PCC CRC0 Register (PCC_CRC0) | 32 | RW | 8000_0000h |
| C0h | PCC LPIT0 Register (PCC_LPIT0) | 32 | RW | 8000_0000h |
| D4h | PCC TPM0 Register (PCC_TPM0) | 32 | RW | 8000_0000h |
| D8h | PCC TPM1 Register (PCC_TPM1) | 32 | RW | 8000_0000h |
| DCh | PCC TPM2 Register (PCC_TPM2) | 32 | RW | 8000_0000h |
| E0h | PCC EMVSIM0 Register (PCC_EMVSIM0) | 32 | RW | 8000_0000h |
| E4h | PCC FLEXIO0 Register (PCC_FLEXIO0) | 32 | RW | 8000_0000h |
| E8h | PCC LPI2C0 Register (PCC_LPI2C0) | 32 | RW | 8000_0000h |
| ECh | PCC LPI2C1 Register (PCC_LPI2C1) | 32 | RW | 8000_0000h |
| F0h | PCC LPI2C2 Register (PCC_LPI2C2) | 32 | RW | 8000_0000h |
| F4h | PCC I2S0 Register (PCC_I2S0) | 32 | RW | 8000_0000h |
| F8h | PCC USDHC0 Register (PCC_USDHC0) | 32 | RW | 8000_0000h |
| FCh | PCC LPSPI0 Register (PCC_LPSPI0) | 32 | RW | 8000_0000h |
| 100h | PCC LPSPI1 Register (PCC_LPSPI1) | 32 | RW | 8000_0000h |
| 104h | PCC LPSPI2 Register (PCC_LPSPI2) | 32 | RW | 8000_0000h |
| 108h | PCC LPUART0 Register (PCC_LPUART0) | 32 | RW | 8000_0000h |
| 10Ch | PCC LPUART1 Register (PCC_LPUART1) | 32 | RW | 8000_0000h |
| 110h | PCC LPUART2 Register (PCC_LPUART2) | 32 | RW | 8000_0000h |
| 114h | PCC USB0 Register (PCC_USB0) | 32 | RW | 8000_0000h |
| 118h | PCC PORTA Register (PCC_PORTA) | 32 | RW | 8000_0000h |
| 11Ch | PCC PORTB Register (PCC_PORTB) | 32 | RW | 8000_0000h |
| 120h | PCC PORTC Register (PCC_PORTC) | 32 | RW | 8000_0000h |
| 124h | PCC PORTD Register (PCC_PORTD) | 32 | RW | 8000_0000h |
| 128h | PCC LPADC0 Register (PCC_LPADC0) | 32 | RW | 8000_0000h |
| 130h | PCC LPDAC0 Register (PCC_LPDAC0) | 32 | RW | 8000_0000h |
| 134h | PCC VREF Register (PCC_VREF) | 32 | RW | 8000_0000h |
| 200h | PCC TRACE Register (PCC_TRACE) | 32 | RW | 8000_0000h |

## 30.5.1.2 PCC MSCM Register (PCC_MSCM)

### 30.5.1.2.1 Offset

| Register | Offset |
|---|---|
| PCC_MSCM | 4h |

### 30.5.1.2.2 Function

This register is for the MSCM module.

### 30.5.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PR | CGC | INUSE | 0 | | 0 | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | 0 | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 30.5.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.3 PCC AXBS0 Register (PCC_AXBS0)

### 30.5.1.3.1 Offset

| Register | Offset |
|---|---|
| PCC_AXBS0 | 10h |

### 30.5.1.3.2 Function

This register is for the AXBS0 module.

### 30.5.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PR | CGC | INUSE | 0 | | 0 | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | 0 | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 30.5.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.4 PCC DMA0 Register (PCC_DMA0)

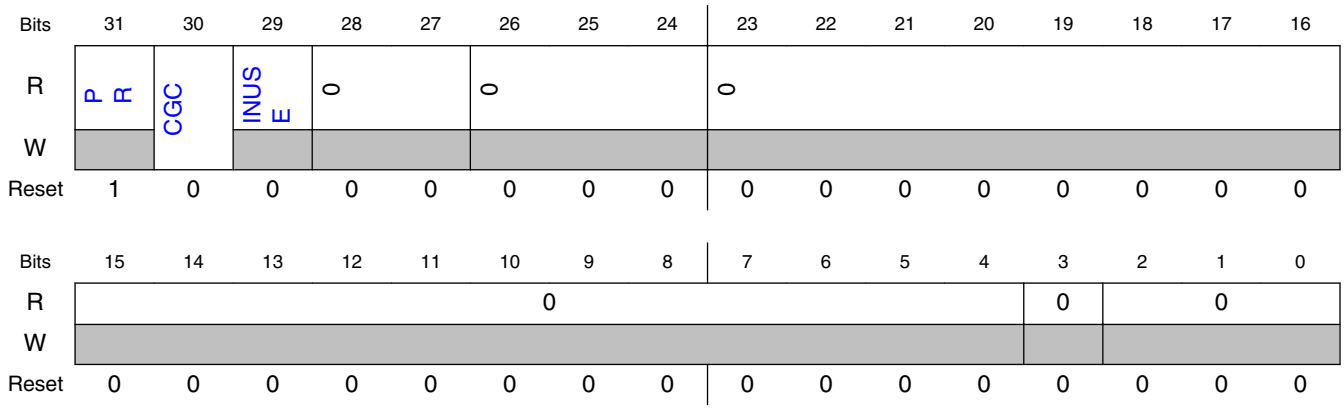
### 30.5.1.4.1 Offset

| Register | Offset |
|---|---|
| PCC_DMA0 | 20h |

### 30.5.1.4.2 Function

This register is for the DMA0 module.

### 30.5.1.4.3 Diagram




### 30.5.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.5 PCC FLEXBUS Register (PCC_FLEXBUS)

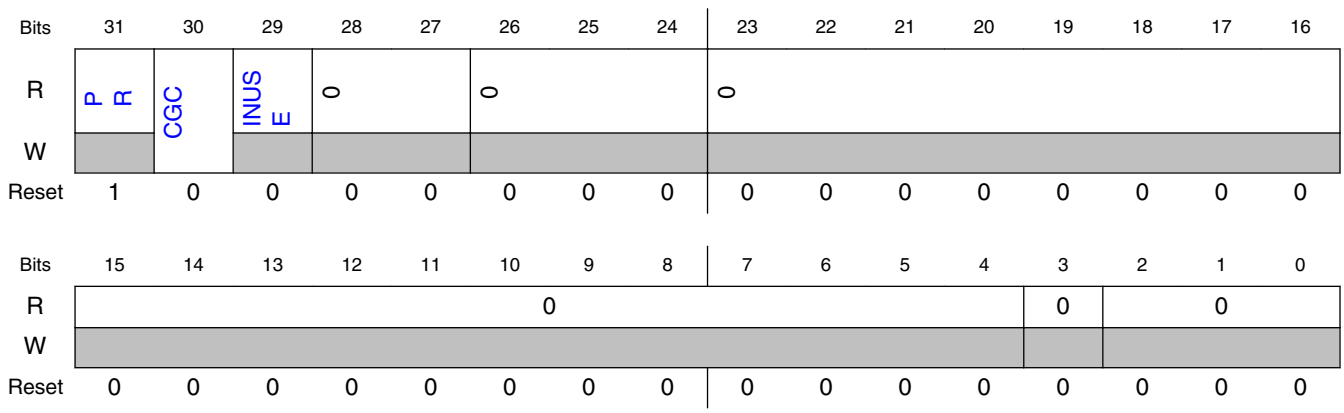
### 30.5.1.5.1 Offset

| Register | Offset |
|---|---|
| PCC_FLEXBUS | 30h |

### 30.5.1.5.2 Function

This register is for the FLEXBUS module.

### 30.5.1.5.3 Diagram




### 30.5.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.6 PCC XRDC_MGR Register (PCC_XRDC_MGR)

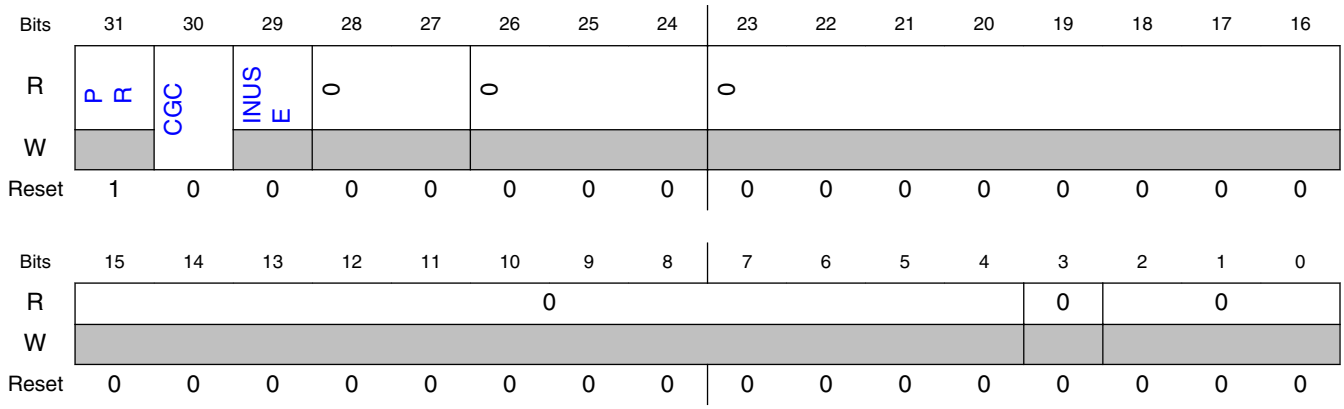
### 30.5.1.6.1 Offset

| Register | Offset |
|---|---|
| PCC_XRDC_MGR | 50h |

### 30.5.1.6.2 Function

This register is for the XRDC_MGR module.

### 30.5.1.6.3 Diagram




### 30.5.1.6.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.7 PCC XRDC_PAC Register (PCC_XRDC_PAC)

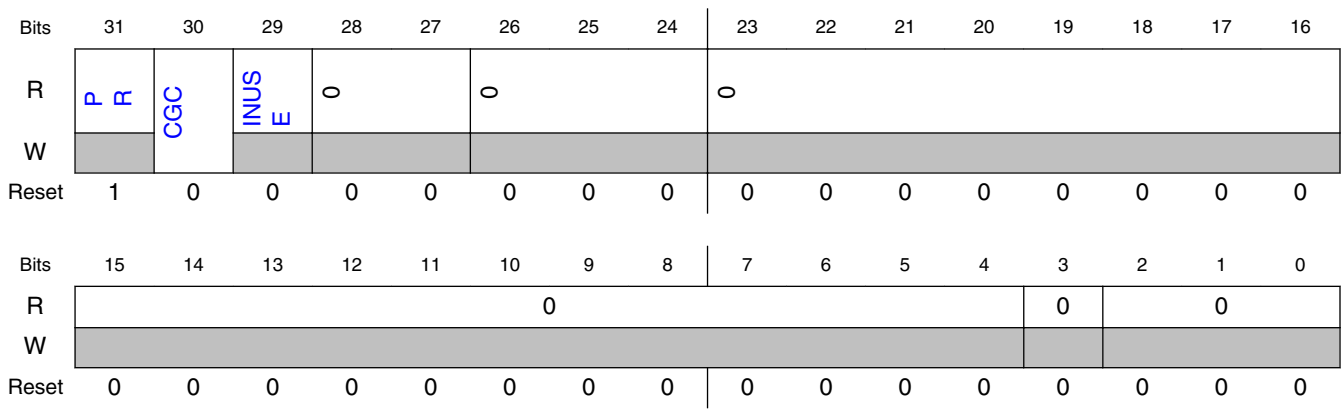
### 30.5.1.7.1 Offset

| Register | Offset |
|---|---|
| PCC_XRDC_PAC | 58h |

### 30.5.1.7.2 Function

This register is for the XRDC_PAC module.

### 30.5.1.7.3 Diagram




### 30.5.1.7.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.8 PCC XRDC_MRC Register (PCC_XRDC_MRC)

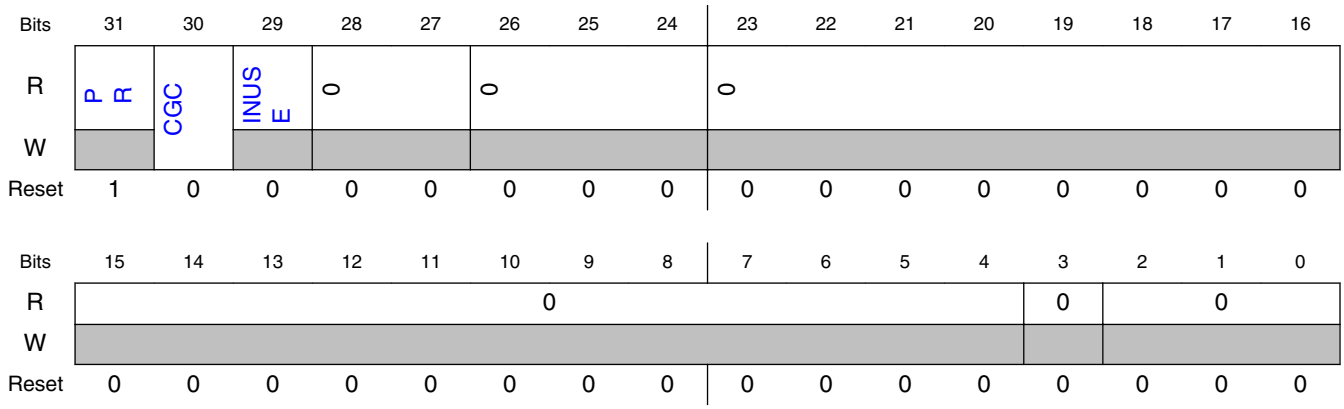
### 30.5.1.8.1 Offset

| Register | Offset |
|---|---|
| PCC_XRDC_MRC | 5Ch |

### 30.5.1.8.2 Function

This register is for the XRDC_MRC module.

### 30.5.1.8.3 Diagram




### 30.5.1.8.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.9 PCC SEMA42_0 Register (PCC_SEMA42_0)

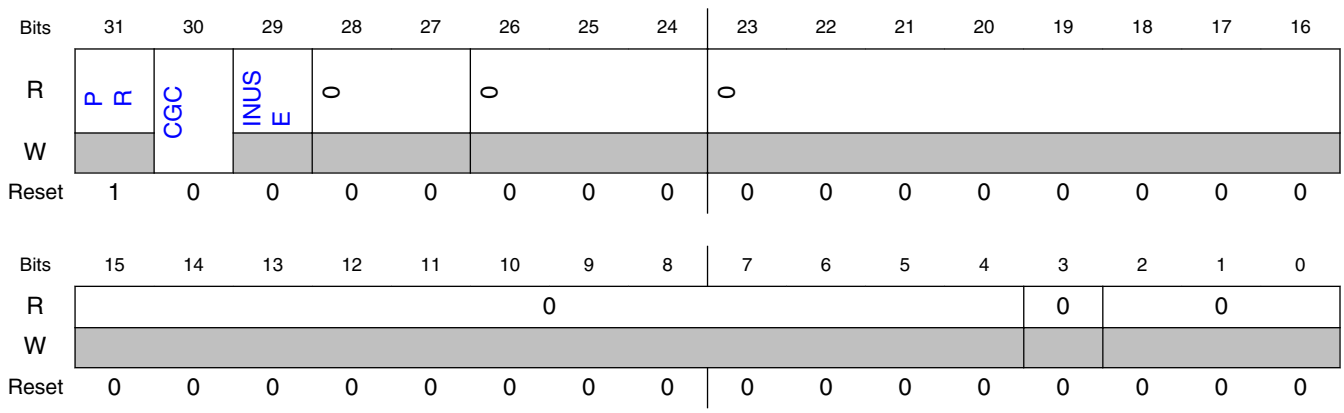
### 30.5.1.9.1 Offset

| Register | Offset |
|---|---|
| PCC_SEMA42_0 | 6Ch |

### 30.5.1.9.2 Function

This register is for the SEMA42_0 module.

### 30.5.1.9.3 Diagram




### 30.5.1.9.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.10 PCC DMAMUX0 Register (PCC_DMAMUX0)

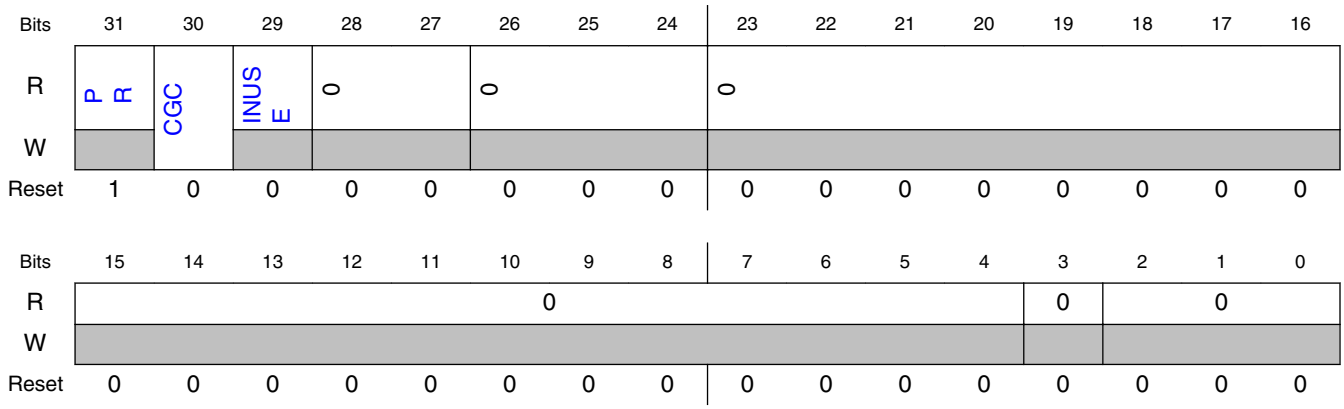
### 30.5.1.10.1 Offset

| Register | Offset |
|---|---|
| PCC_DMAMUX0 | 84h |

### 30.5.1.10.2 Function

This register is for the DMAMUX0 module.

### 30.5.1.10.3 Diagram




### 30.5.1.10.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.11 PCC EWM Register (PCC_EWM)

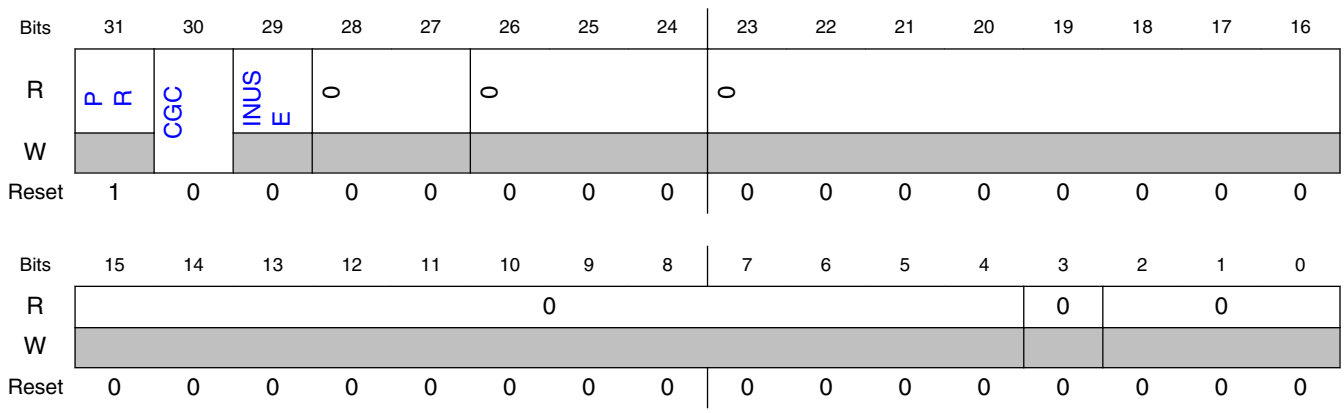
### 30.5.1.11.1 Offset

| Register | Offset |
|---|---|
| PCC_EWM | 88h |

### 30.5.1.11.2 Function

This register is for the EWM module.

### 30.5.1.11.3 Diagram




### 30.5.1.11.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.12 PCC MUA Register (PCC_MUA)

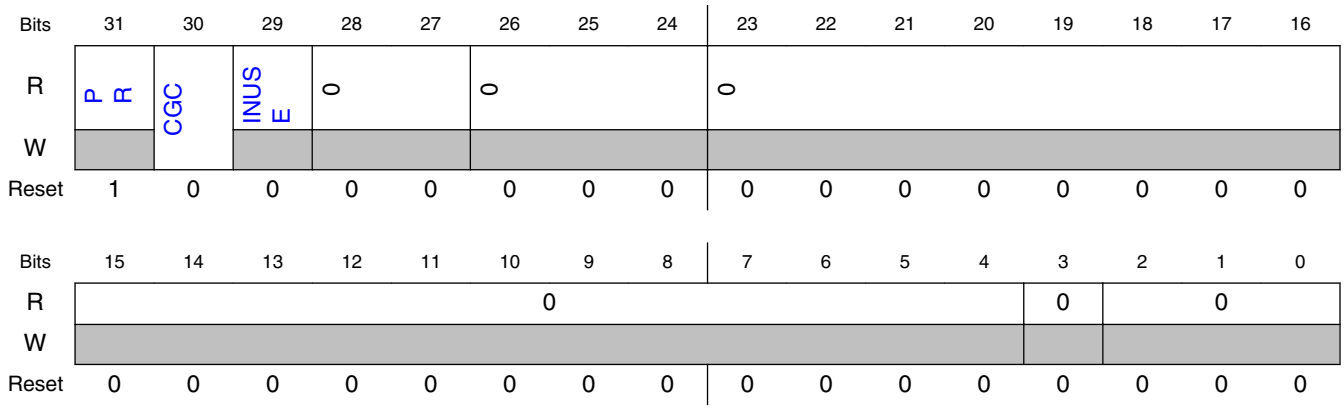
### 30.5.1.12.1 Offset

| Register | Offset |
|---|---|
| PCC_MUA | 94h |

### 30.5.1.12.2 Function

This register is for the MUA module.

## 30.5.1.12.3 Diagram




## 30.5.1.12.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.13 PCC CRC0 Register (PCC_CRC0)

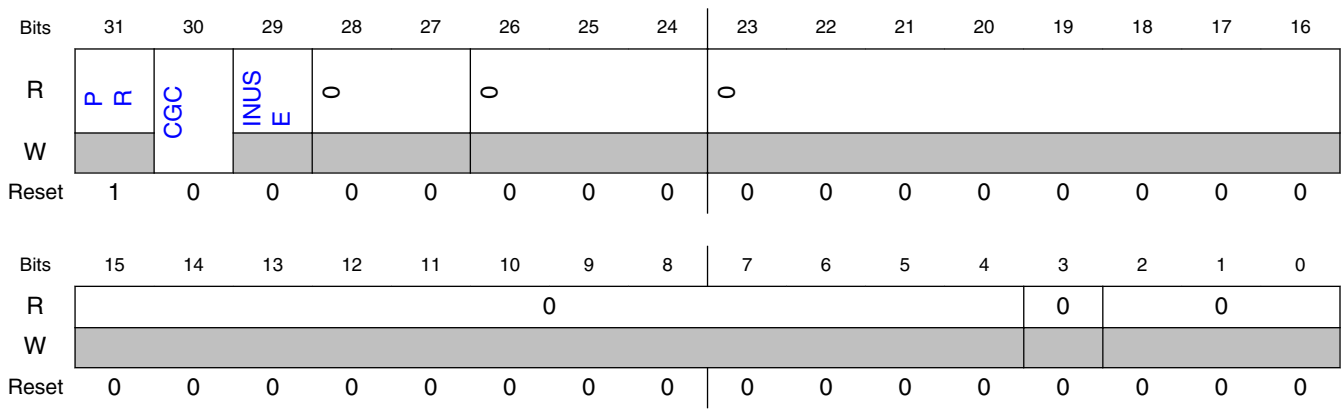
### 30.5.1.13.1 Offset

| Register | Offset |
|---|---|
| PCC_CRC0 | BCh |

### 30.5.1.13.2 Function

This register is for the CRC0 module.

### 30.5.1.13.3 Diagram




### 30.5.1.13.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.14 PCC LPIT0 Register (PCC_LPIT0)

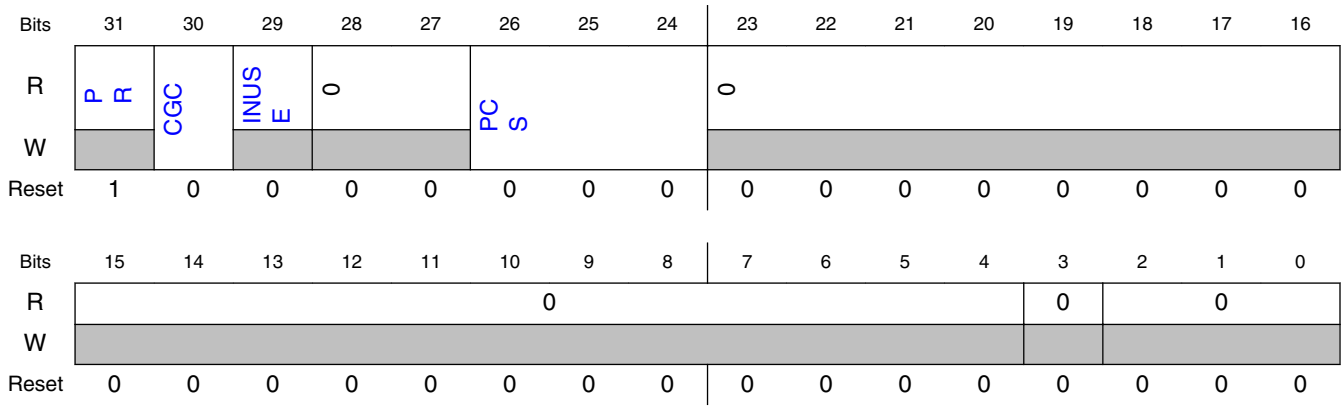
### 30.5.1.14.1 Offset

| Register | Offset |
|---|---|
| PCC_LPIT0 | C0h |

### 30.5.1.14.2 Function

This register is for the LPIT0 module.

### 30.5.1.14.3 Diagram




### 30.5.1.14.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.15 PCC TPM0 Register (PCC_TPM0)

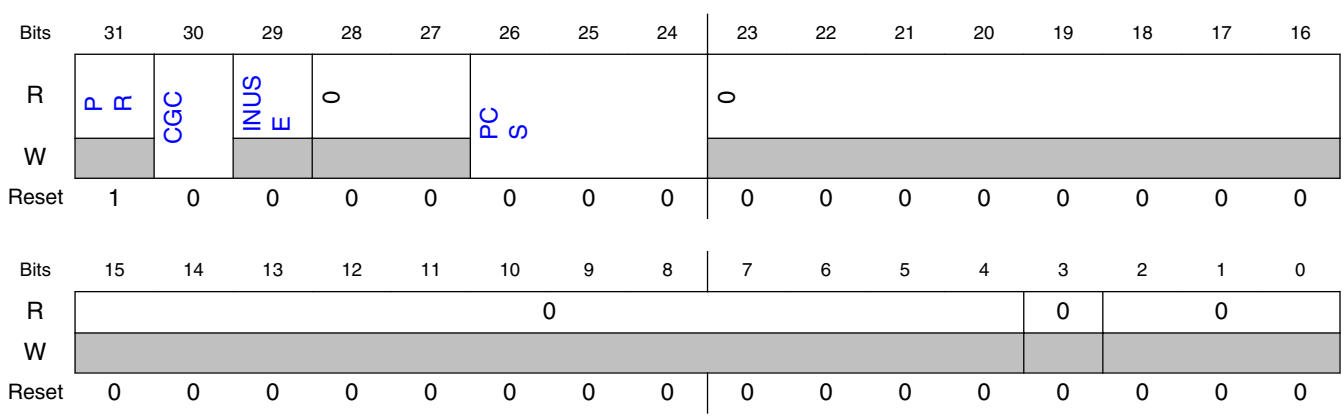
### 30.5.1.15.1 Offset

| Register | Offset |
|---|---|
| PCC_TPM0 | D4h |

### 30.5.1.15.2 Function

This register is for the TPM0 module.

### 30.5.1.15.3 Diagram



### 30.5.1.15.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.16 PCC TPM1 Register (PCC_TPM1)

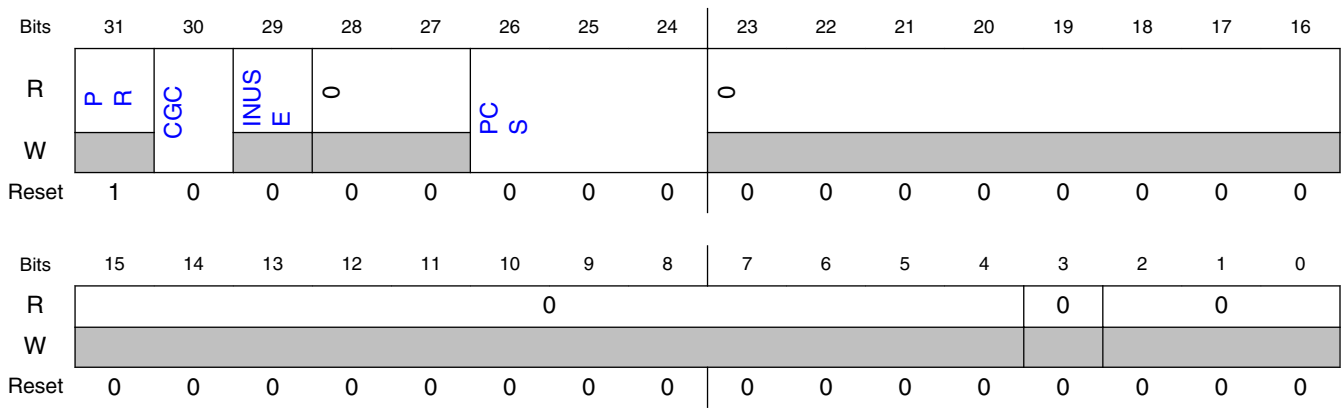
### 30.5.1.16.1 Offset

| Register | Offset |
|---|---|
| PCC_TPM1 | D8h |

### 30.5.1.16.2 Function

This register is for the TPM1 module.

### 30.5.1.16.3 Diagram




### 30.5.1.16.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.17 PCC TPM2 Register (PCC_TPM2)

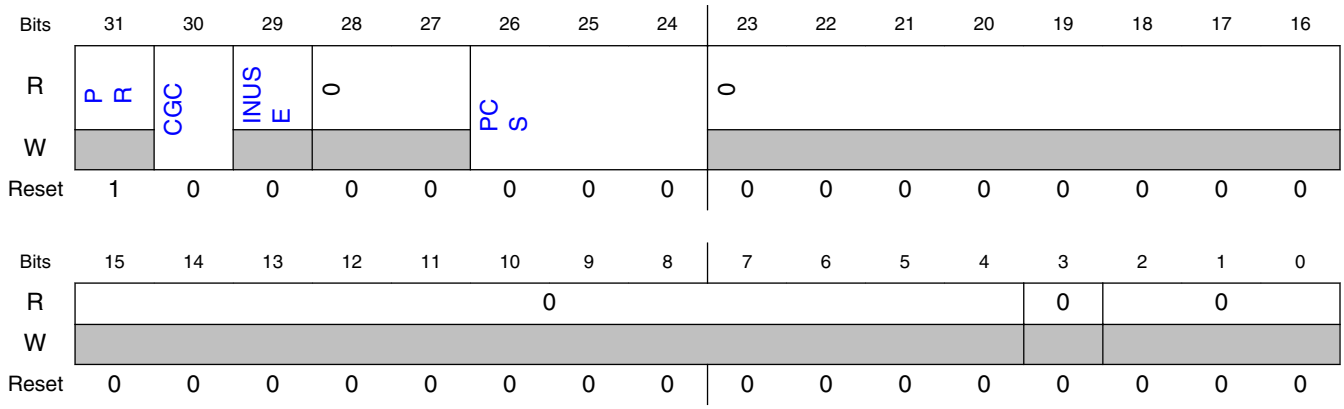
### 30.5.1.17.1 Offset

| Register | Offset |
|---|---|
| PCC_TPM2 | DCh |

### 30.5.1.17.2 Function

This register is for the TPM2 module.

## 30.5.1.17.3 Diagram




## 30.5.1.17.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.18 PCC EMVSIM0 Register (PCC_EMVSIM0)

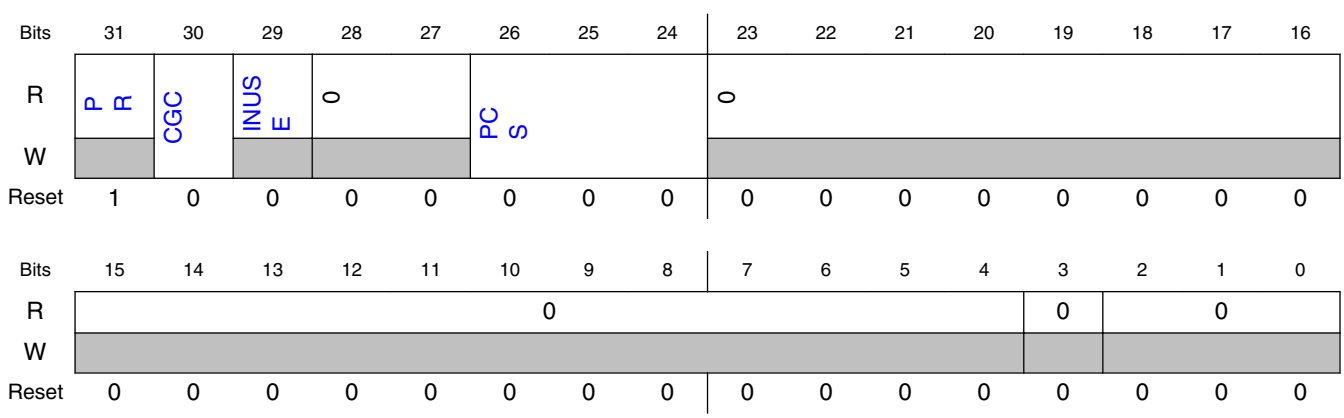
### 30.5.1.18.1 Offset

| Register | Offset |
|---|---|
| PCC_EMVSIM0 | E0h |

### 30.5.1.18.2 Function

This register is for the EMVSIM0 module.

### 30.5.1.18.3 Diagram



### 30.5.1.18.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.19 PCC FLEXIO0 Register (PCC_FLEXIO0)

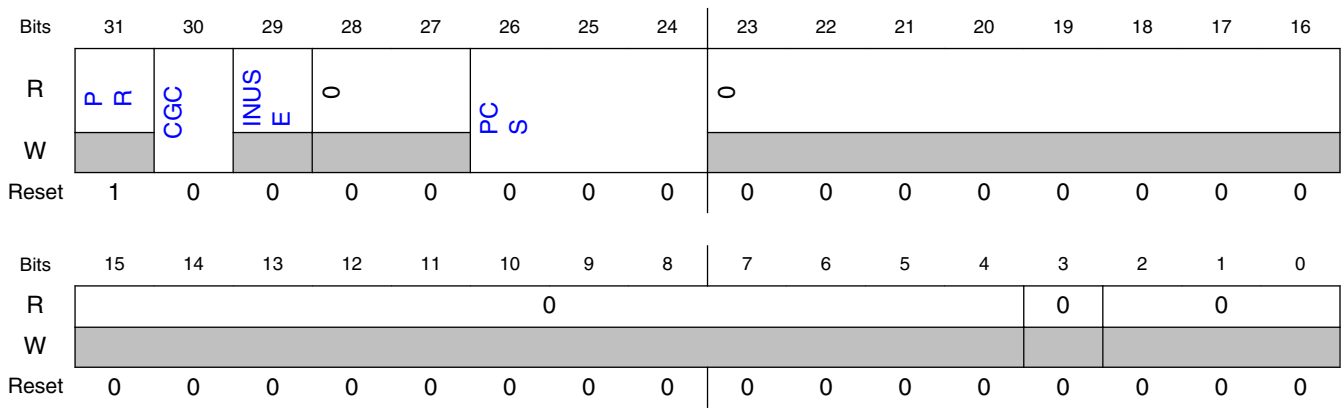
### 30.5.1.19.1 Offset

| Register | Offset |
|---|---|
| PCC_FLEXIO0 | E4h |

### 30.5.1.19.2 Function

This register is for the FLEXIO0 module.

### 30.5.1.19.3 Diagram




### 30.5.1.19.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.20 PCC LPI2C0 Register (PCC_LPI2C0)

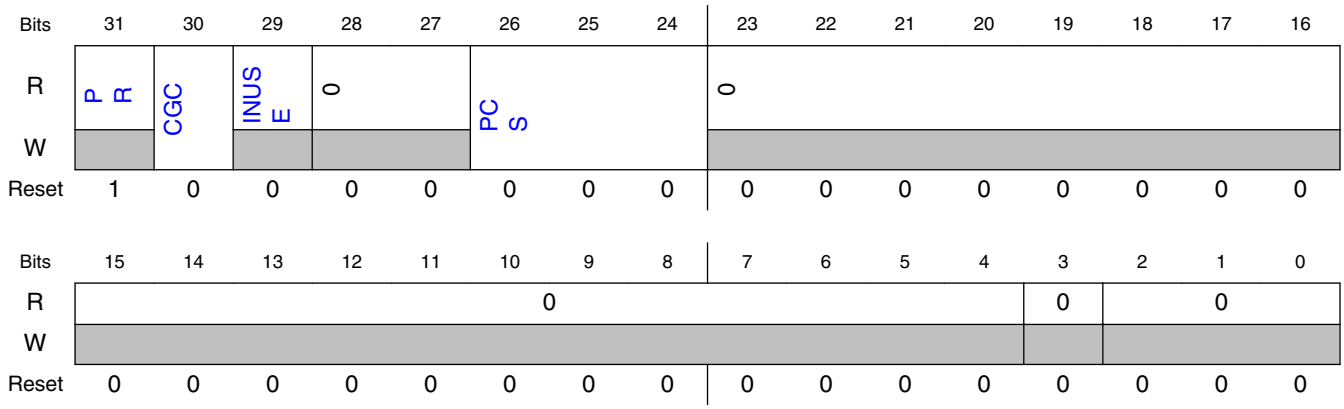
### 30.5.1.20.1 Offset

| Register | Offset |
|---|---|
| PCC_LPI2C0 | E8h |

### 30.5.1.20.2 Function

This register is for the LPI2C0 module.

## 30.5.1.20.3 Diagram




## 30.5.1.20.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.21 PCC LPI2C1 Register (PCC_LPI2C1)

### 30.5.1.21.1 Offset

| Register | Offset |
|---|---|
| PCC_LPI2C1 | ECh |

### 30.5.1.21.2 Function

This register is for the LPI2C1 module.

### 30.5.1.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PR | CGC | INUSE | 0 | | PCS | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | 0 | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 30.5.1.21.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.22 PCC LPI2C2 Register (PCC_LPI2C2)

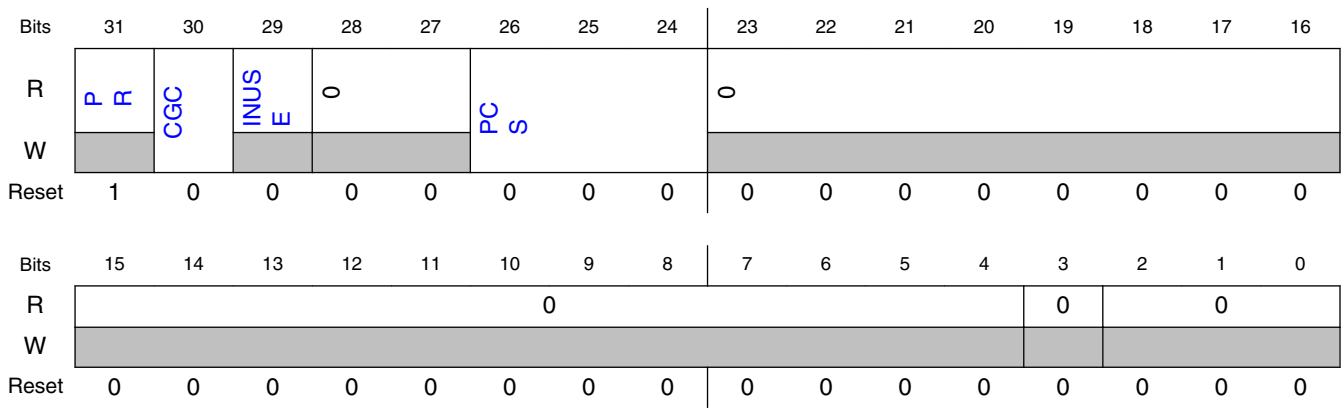
### 30.5.1.22.1 Offset

| Register | Offset |
|---|---|
| PCC_LPI2C2 | F0h |

### 30.5.1.22.2 Function

This register is for the LPI2C2 module.

### 30.5.1.22.3 Diagram




### 30.5.1.22.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.23 PCC I2S0 Register (PCC_I2S0)

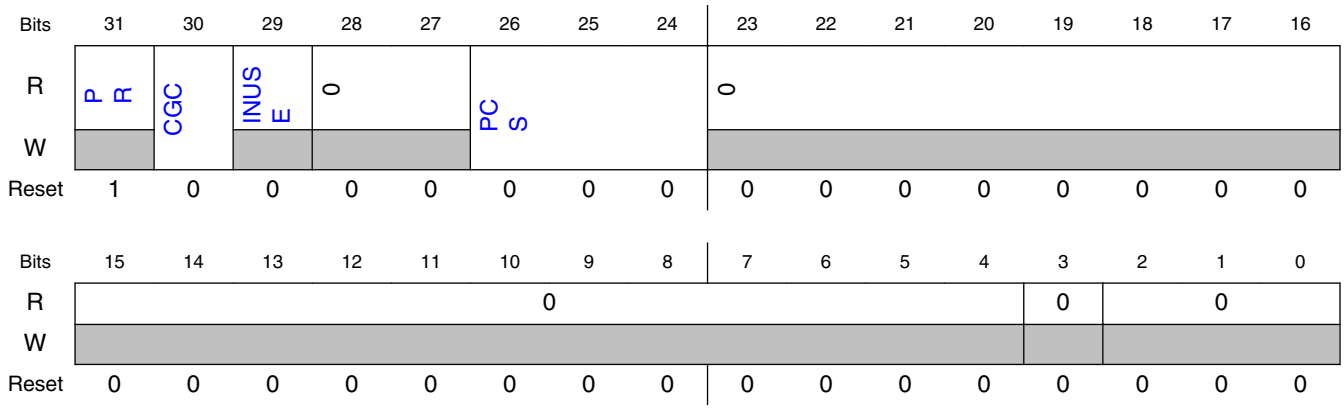
### 30.5.1.23.1 Offset

| Register | Offset |
|---|---|
| PCC_I2S0 | F4h |

### 30.5.1.23.2 Function

This register is for the I2S0 module.

## 30.5.1.23.3 Diagram




## 30.5.1.23.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off. An external clock can be enabled for this peripheral.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.24 PCC USDHC0 Register (PCC_USDHC0)

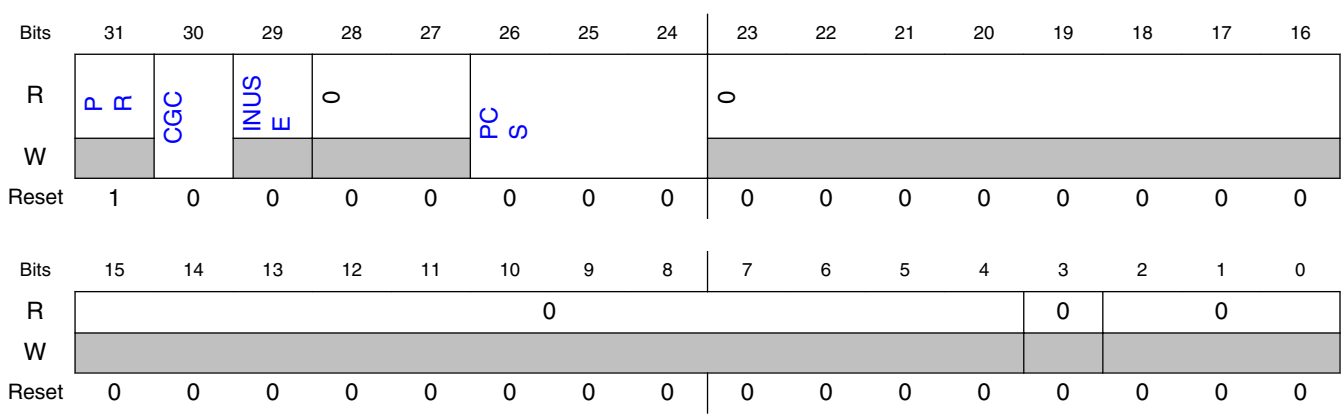
### 30.5.1.24.1 Offset

| Register | Offset |
|---|---|
| PCC_USDHC0 | F8h |

### 30.5.1.24.2 Function

This register is for the USDHC0 module.

### 30.5.1.24.3 Diagram



### 30.5.1.24.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.25 PCC LPSPI0 Register (PCC_LPSPI0)

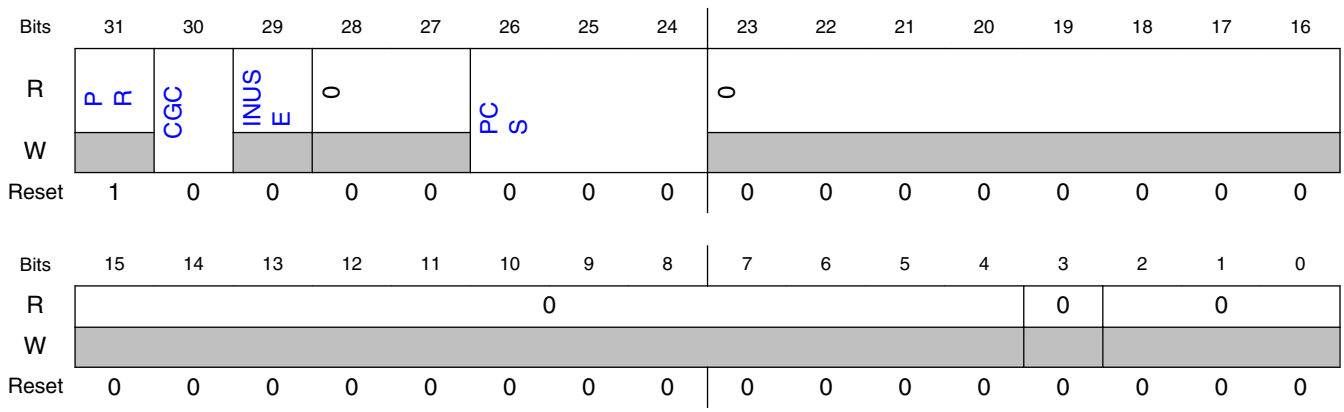
### 30.5.1.25.1 Offset

| Register | Offset |
|---|---|
| PCC_LPSPI0 | FCh |

### 30.5.1.25.2 Function

This register is for the LPSPI0 module.

### 30.5.1.25.3 Diagram




### 30.5.1.25.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.26 PCC LPSPI1 Register (PCC_LPSPI1)

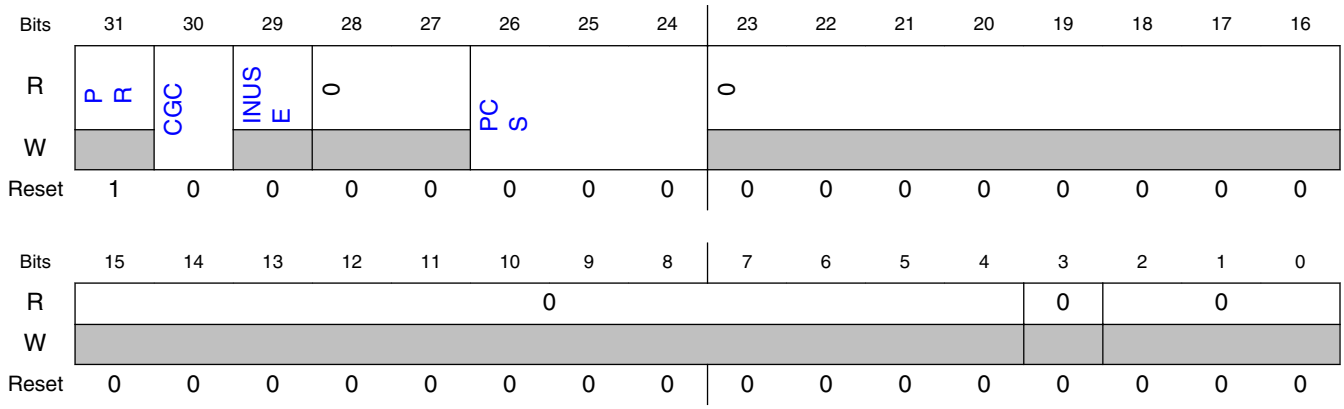
### 30.5.1.26.1 Offset

| Register | Offset |
|---|---|
| PCC_LPSPI1 | 100h |

### 30.5.1.26.2 Function

This register is for the LPSPI1 module.

## 30.5.1.26.3 Diagram




## 30.5.1.26.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.27 PCC LPSPI2 Register (PCC_LPSPI2)

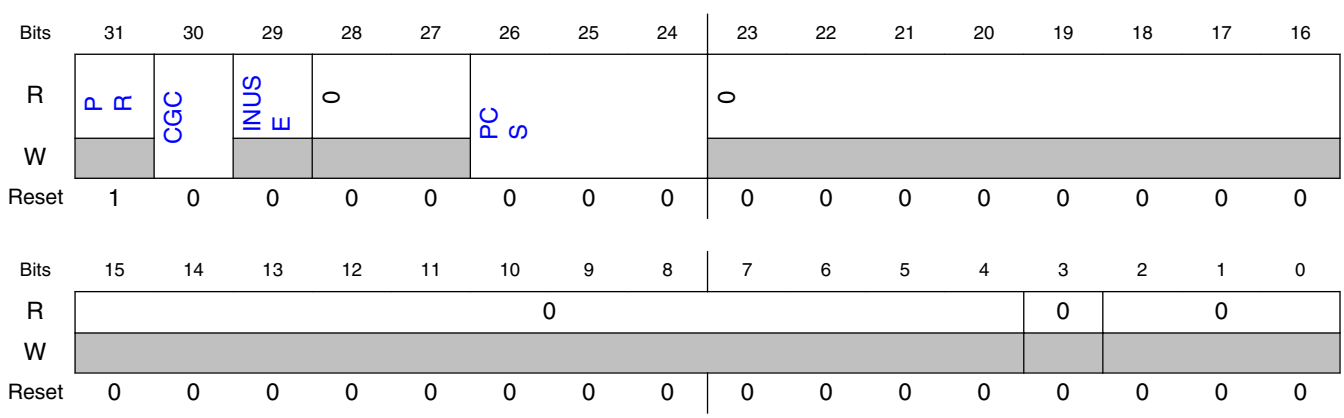
### 30.5.1.27.1 Offset

| Register | Offset |
|---|---|
| PCC_LPSPI2 | 104h |

### 30.5.1.27.2 Function

This register is for the LPSPI2 module.

### 30.5.1.27.3 Diagram



### 30.5.1.27.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.28 PCC LPUART0 Register (PCC_LPUART0)

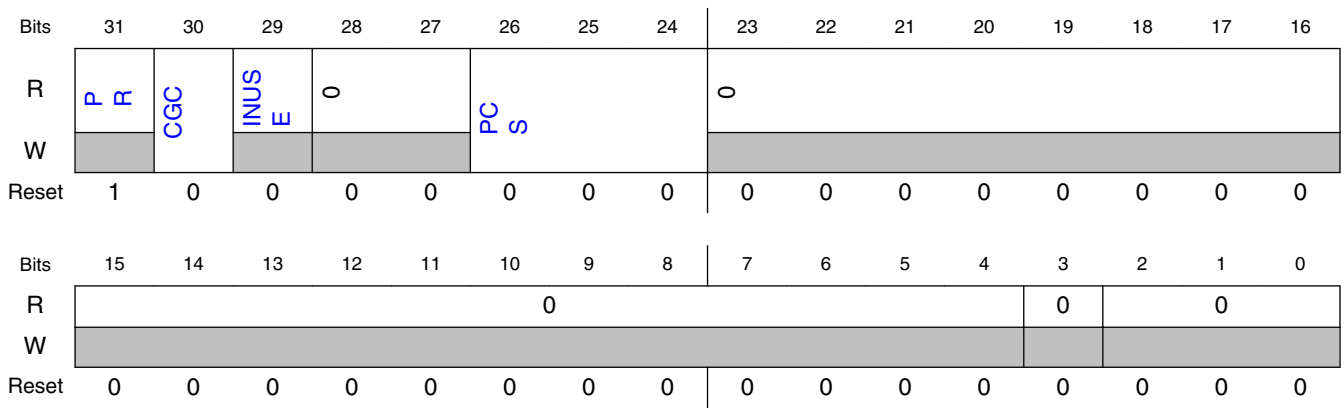
### 30.5.1.28.1 Offset

| Register | Offset |
|---|---|
| PCC_LPUART0 | 108h |

### 30.5.1.28.2 Function

This register is for the LPUART0 module.

### 30.5.1.28.3 Diagram




### 30.5.1.28.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.29 PCC LPUART1 Register (PCC_LPUART1)

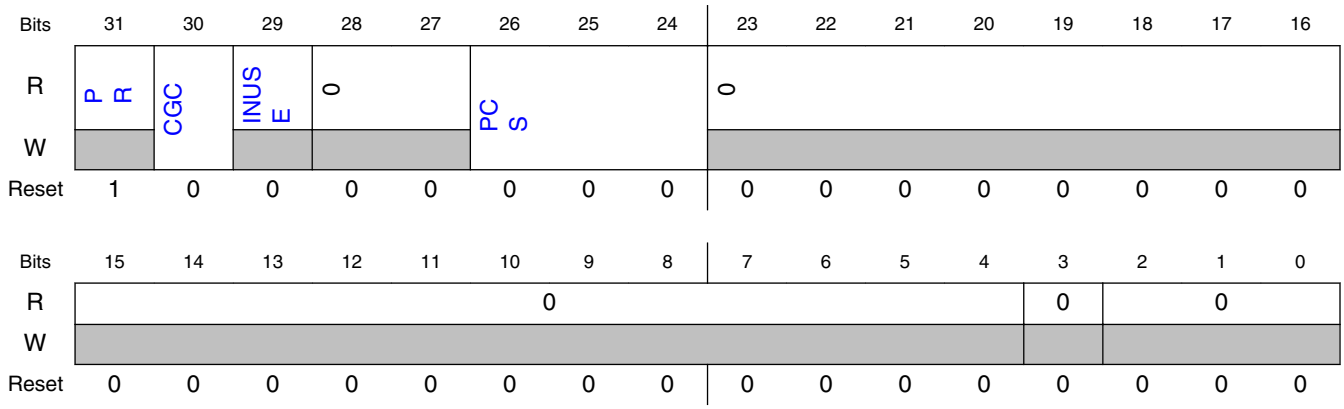
### 30.5.1.29.1 Offset

| Register | Offset |
|---|---|
| PCC_LPUART1 | 10Ch |

### 30.5.1.29.2 Function

This register is for the LPUART1 module.

## 30.5.1.29.3 Diagram




## 30.5.1.29.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.30 PCC LPUART2 Register (PCC_LPUART2)

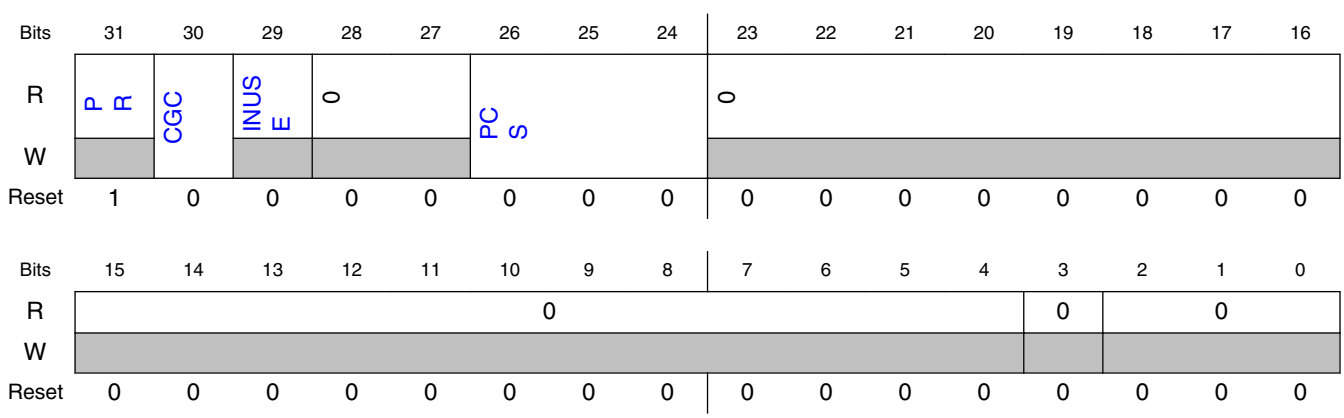
### 30.5.1.30.1 Offset

| Register | Offset |
|---|---|
| PCC_LPUART2 | 110h |

### 30.5.1.30.2 Function

This register is for the LPUART2 module.

### 30.5.1.30.3 Diagram



## 30.5.1.30.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.31 PCC USB0 Register (PCC_USB0)

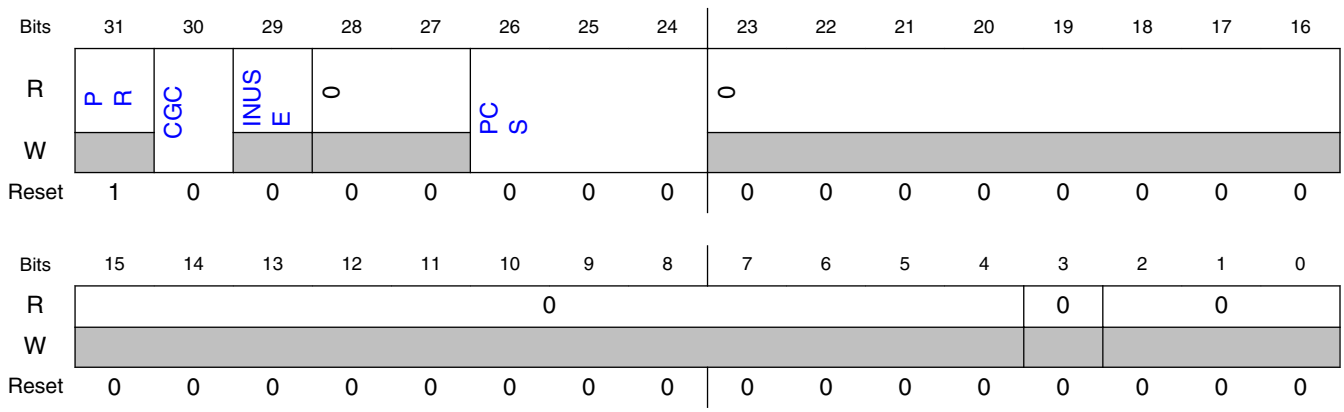
## 30.5.1.31.1 Offset

| Register | Offset |
|---|---|
| PCC_USB0 | 114h |

## 30.5.1.31.2 Function

This register is for the USB0 module.

## 30.5.1.31.3 Diagram




## 30.5.1.31.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.32 PCC PORTA Register (PCC_PORTA)

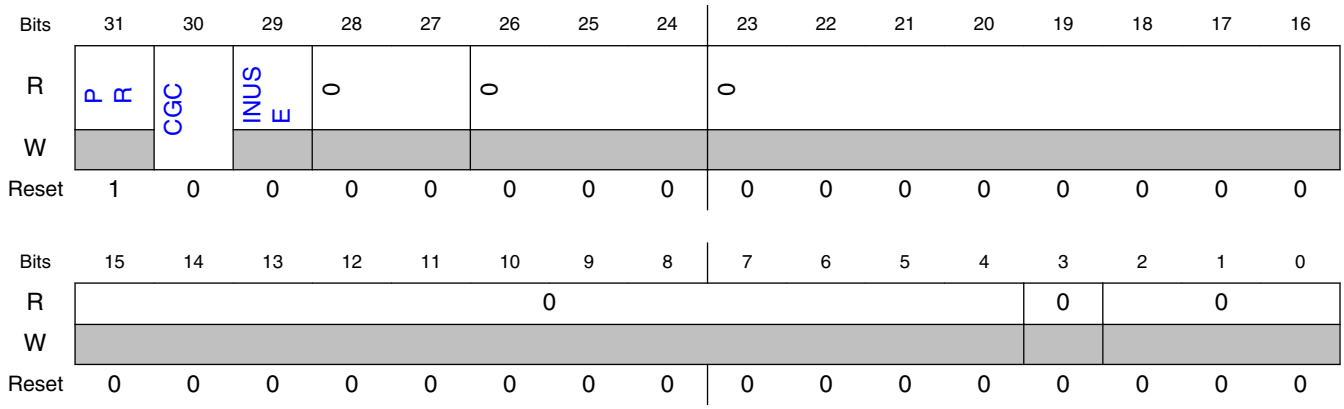
### 30.5.1.32.1 Offset

| Register | Offset |
|---|---|
| PCC_PORTA | 118h |

### 30.5.1.32.2 Function

This register is for the PORTA module.

## 30.5.1.32.3 Diagram




## 30.5.1.32.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.33 PCC PORTB Register (PCC_PORTB)

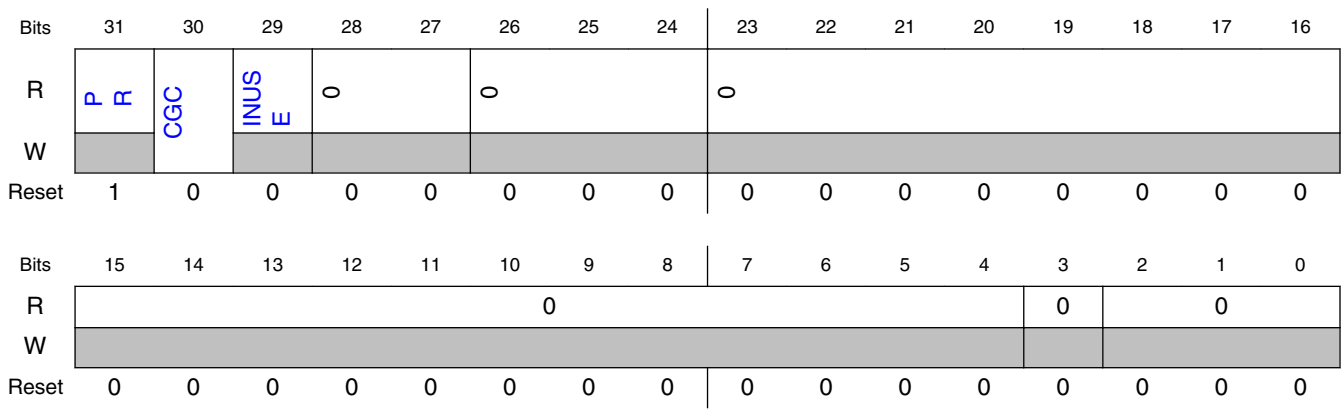
### 30.5.1.33.1 Offset

| Register | Offset |
|---|---|
| PCC_PORTB | 11Ch |

### 30.5.1.33.2 Function

This register is for the PORTB module.

### 30.5.1.33.3 Diagram




### 30.5.1.33.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.34 PCC PORTC Register (PCC_PORTC)

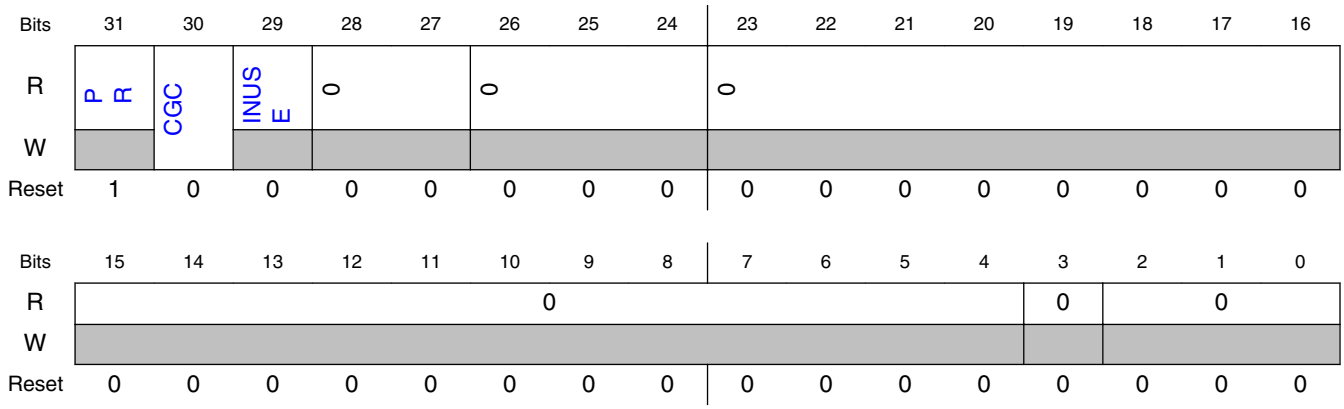
### 30.5.1.34.1 Offset

| Register | Offset |
|---|---|
| PCC_PORTC | 120h |

### 30.5.1.34.2 Function

This register is for the PORTC module.

## 30.5.1.34.3 Diagram




## 30.5.1.34.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.35 PCC PORTD Register (PCC_PORTD)

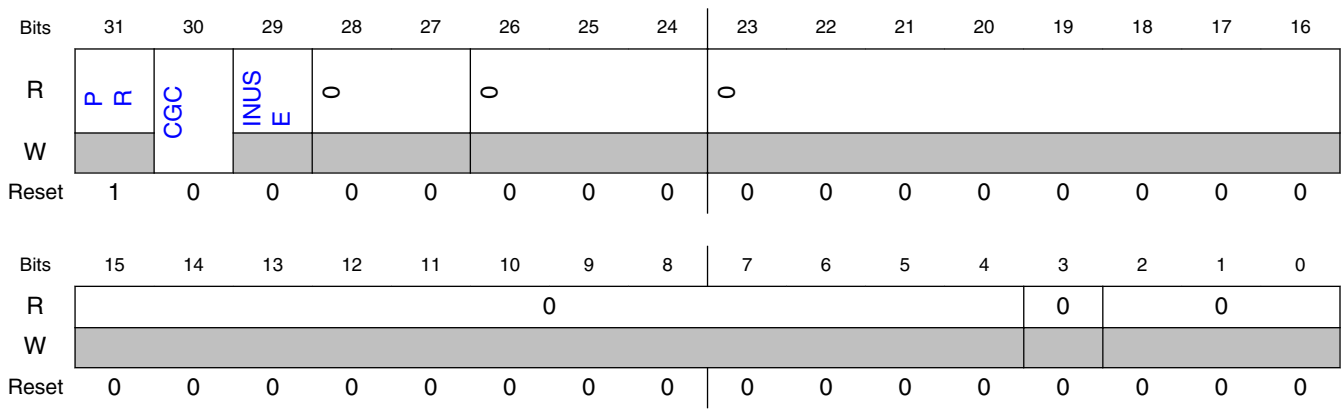
### 30.5.1.35.1 Offset

| Register | Offset |
|---|---|
| PCC_PORTD | 124h |

### 30.5.1.35.2 Function

This register is for the PORTD module.

### 30.5.1.35.3 Diagram




### 30.5.1.35.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.36 PCC LPADC0 Register (PCC_LPADC0)

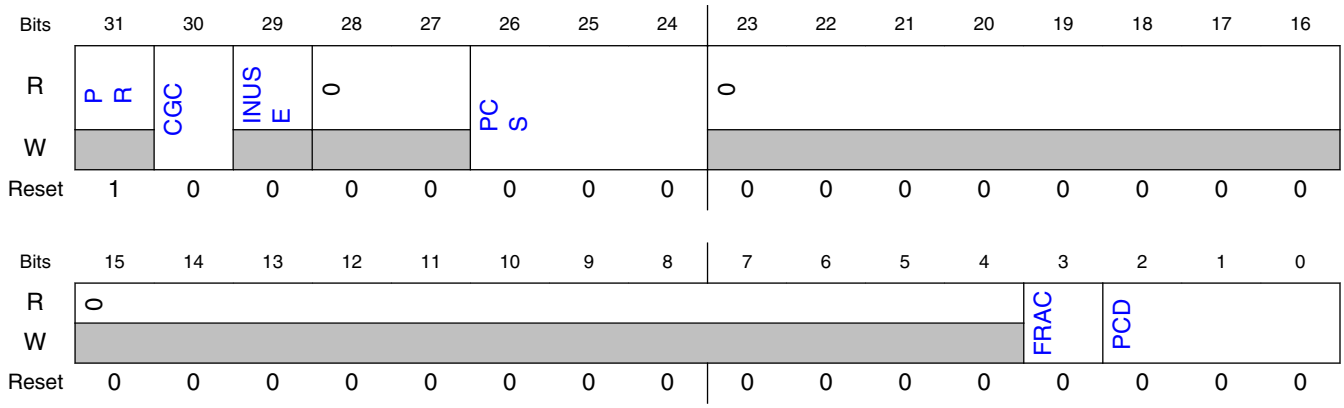
### 30.5.1.36.1 Offset

| Register | Offset |
|---|---|
| PCC_LPADC0 | 128h |

### 30.5.1.36.2 Function

This register is for the LPADC0 module.

## 30.5.1.36.3 Diagram




## 30.5.1.36.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off. An external clock can be enabled for this peripheral.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>FRAC | Peripheral Clock Divider Fraction<br>This read/write bit field sets the fraction multiply value for the fractional clock divider used as a clock source. Divider output clock = Divider input clock x [(FRAC+1)/(PCD+1)].<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>**NOTE:** When dividing by 1 (PCD = 000), do not set the FRAC bit; otherwise, the output clock is disabled.<br>0b - Fractional value is 0.<br>1b - Fractional value is 1. |
| 2-0<br>PCD | Peripheral Clock Divider Select<br>This read/write bit field is used for peripherals that require a clock divider. Divider output clock = Divider input clock x [(FRAC+1)/(PCD+1)].<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

## 30.5.1.37 PCC LPDAC0 Register (PCC_LPDAC0)

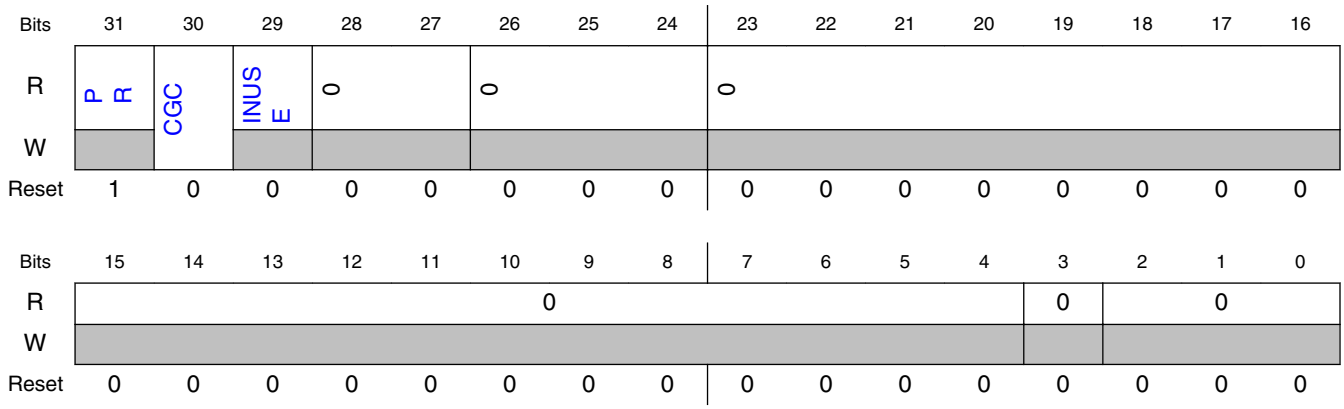
### 30.5.1.37.1 Offset

| Register | Offset |
|---|---|
| PCC_LPDAC0 | 130h |

### 30.5.1.37.2 Function

This register is for the LPDAC0 module.

### 30.5.1.37.3 Diagram




### 30.5.1.37.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.38 PCC VREF Register (PCC_VREF)

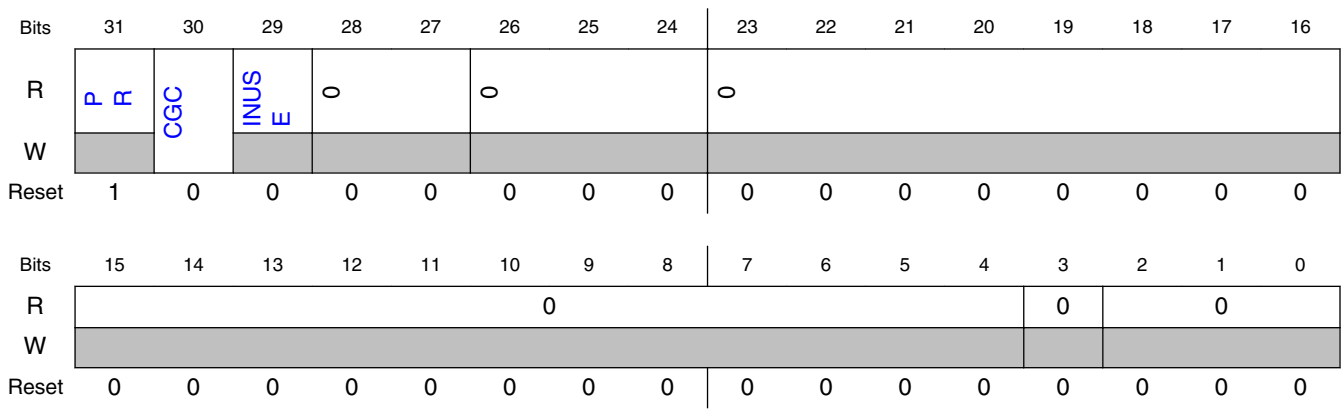
### 30.5.1.38.1 Offset

| Register | Offset |
|---|---|
| PCC_VREF | 134h |

### 30.5.1.38.2 Function

This register is for the VREF module.

### 30.5.1.38.3 Diagram




### 30.5.1.38.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.1.39 PCC TRACE Register (PCC_TRACE)

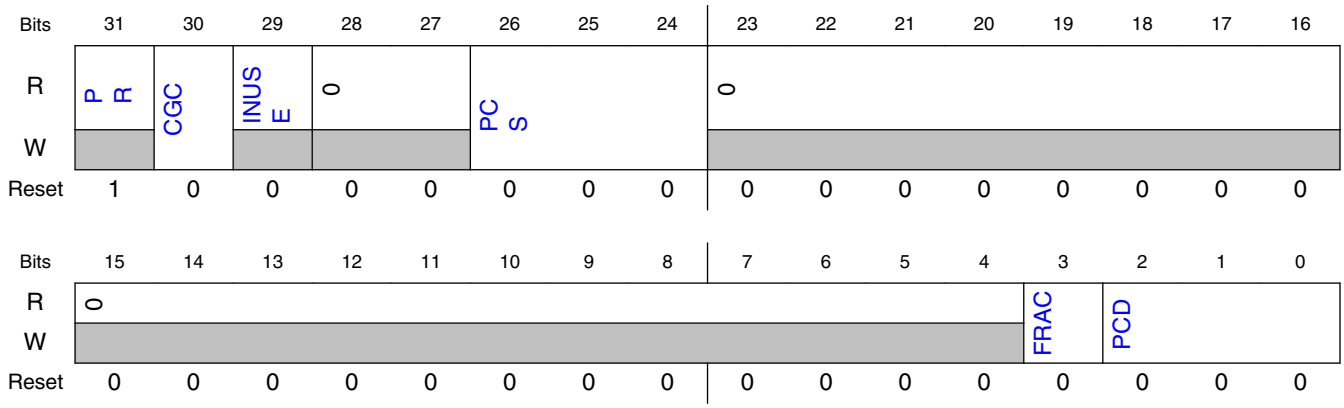
### 30.5.1.39.1 Offset

| Register | Offset |
|---|---|
| PCC_TRACE | 200h |

### 30.5.1.39.2 Function

This register is for the TRACE module.

## 30.5.1.39.3 Diagram




## 30.5.1.39.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>FRAC | Peripheral Clock Divider Fraction<br>This read/write bit field sets the fraction multiply value for the fractional clock divider used as a clock source. Divider output clock = Divider input clock x [(FRAC+1)/(PCD+1)].<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>**NOTE:** When dividing by 1 (PCD = 000), do not set the FRAC bit; otherwise, the output clock is disabled.<br>0b - Fractional value is 0.<br>1b - Fractional value is 1. |
| 2-0<br>PCD | Peripheral Clock Divider Select<br>This read/write bit field is used for peripherals that require a clock divider. Divider output clock = Divider input clock x [(FRAC+1)/(PCD+1)].<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

## 30.5.2 PCC1 register descriptions

### 30.5.2.1 PCC1 Memory map

PCC1 base address: 4102_7000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20h | PCC DMA1 Register (PCC_DMA1) | 32 | RW | 8000_0000h |
| 3Ch | PCC GPIOE Register (PCC_GPIOE) | 32 | RW | 8000_0000h |
| 58h | PCC XRDC_PAC Register (PCC_XRDC_PAC) | 32 | RW | 8000_0000h |
| 5Ch | PCC XRDC_MRC Register (PCC_XRDC_MRC) | 32 | RW | 8000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 6Ch | PCC SEMA42_1 Register (PCC_SEMA42_1) | 32 | RW | 8000_0000h |
| 84h | PCC DMAMUX1 Register (PCC_DMAMUX1) | 32 | RW | 8000_0000h |
| 88h | PCC INTMUX1 Register (PCC_INTMUX1) | 32 | RW | 8000_0000h |
| 90h | PCC MUB Register (PCC_MUB) | 32 | RW | 8000_0000h |
| A0h | PCC CAU3 Register (PCC_CAU3) | 32 | RW | 8000_0000h |
| A4h | PCC TRNG Register (PCC_TRNG) | 32 | RW | 8000_0000h |
| A8h | PCC LPIT1 Register (PCC_LPIT1) | 32 | RW | 8000_0000h |
| B4h | PCC TPM3 Register (PCC_TPM3) | 32 | RW | 8000_0000h |
| B8h | PCC LPI2C3 Register (PCC_LPI2C3) | 32 | RW | 8000_0000h |
| D4h | PCC LPSPI3 Register (PCC_LPSPI3) | 32 | RW | 8000_0000h |
| D8h | PCC LPUART3 Register (PCC_LPUART3) | 32 | RW | 8000_0000h |
| DCh | PCC PORTE Register (PCC_PORTE) | 32 | RW | 8000_0000h |
| 200h | PCC MTB Register (PCC_MTB) | 32 | RW | 8000_0000h |
| 204h | PCC EXT_CLK Register (PCC_EXT_CLK) | 32 | RW | 8000_0000h |

## 30.5.2.2 PCC DMA1 Register (PCC_DMA1)

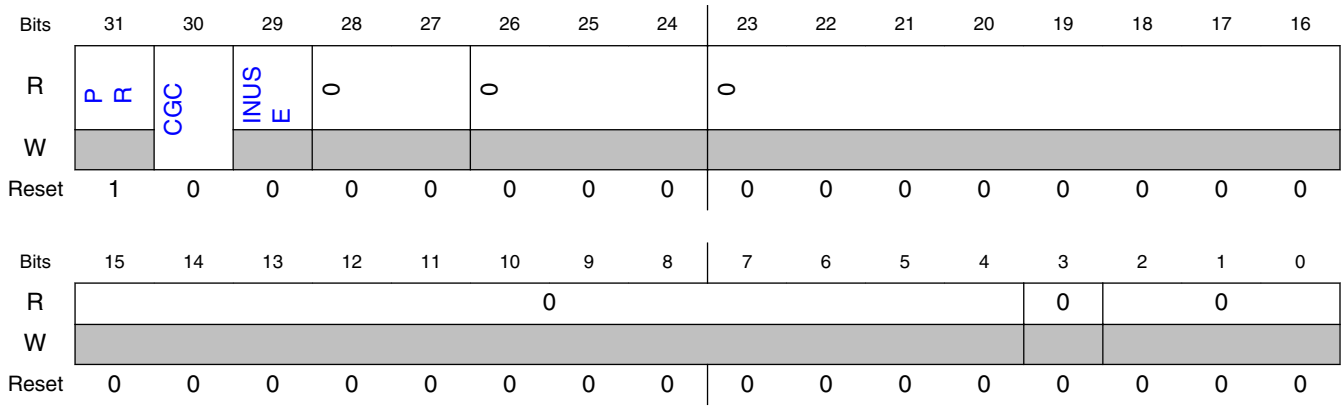
### 30.5.2.2.1 Offset

| Register | Offset |
|---|---|
| PCC_DMA1 | 20h |

### 30.5.2.2.2 Function

This register is for the DMA1 module.

### 30.5.2.2.3 Diagram




### 30.5.2.2.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.3 PCC GPIOE Register (PCC_GPIOE)

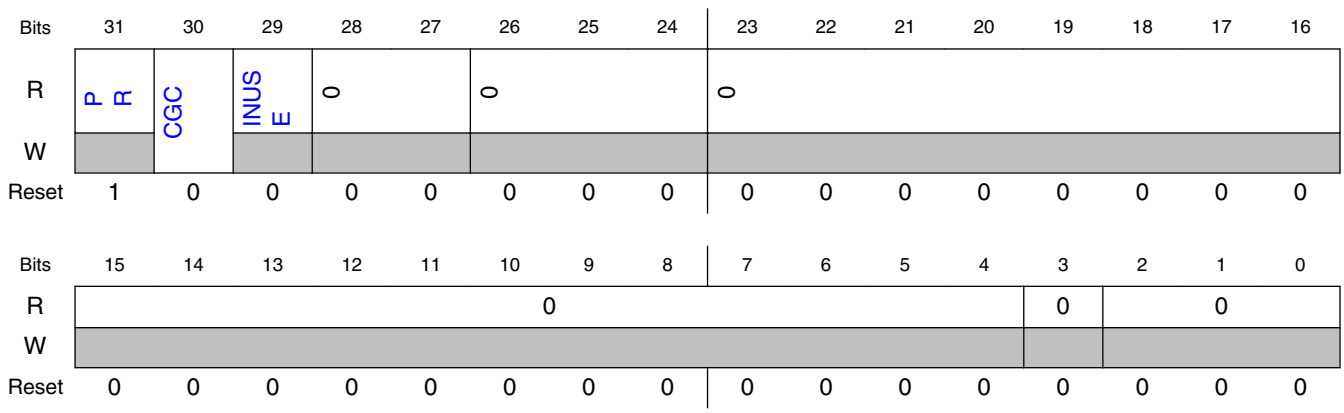
### 30.5.2.3.1 Offset

| Register | Offset |
|---|---|
| PCC_GPIOE | 3Ch |

### 30.5.2.3.2 Function

This register is for the GPIOE module.

### 30.5.2.3.3 Diagram




### 30.5.2.3.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.4 PCC XRDC_PAC Register (PCC_XRDC_PAC)

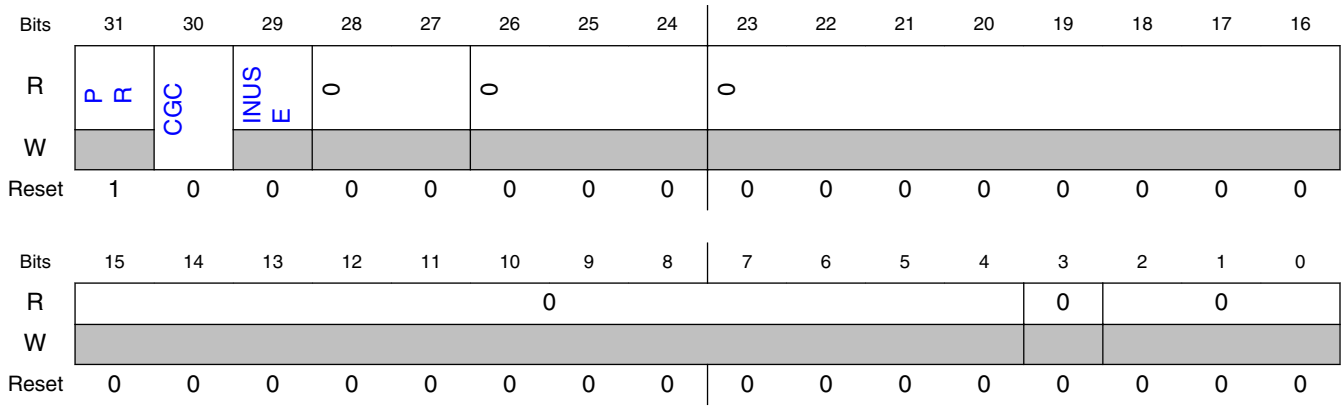
### 30.5.2.4.1 Offset

| Register | Offset |
|---|---|
| PCC_XRDC_PAC | 58h |

### 30.5.2.4.2 Function

This register is for the XRDC_PAC module.

### 30.5.2.4.3 Diagram




### 30.5.2.4.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.5 PCC XRDC_MRC Register (PCC_XRDC_MRC)

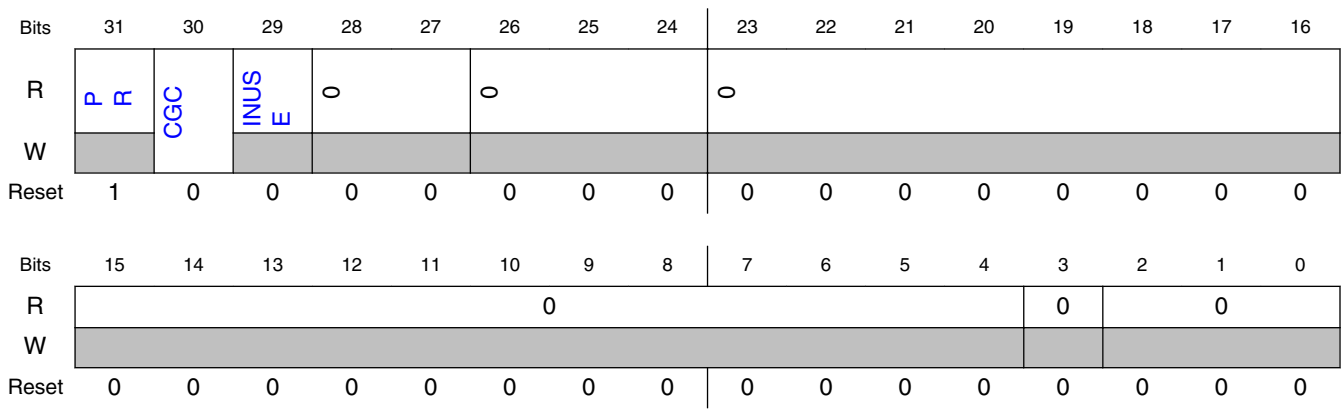
### 30.5.2.5.1 Offset

| Register | Offset |
|---|---|
| PCC_XRDC_MRC | 5Ch |

### 30.5.2.5.2 Function

This register is for the XRDC_MRC module.

### 30.5.2.5.3 Diagram




### 30.5.2.5.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.6 PCC SEMA42_1 Register (PCC_SEMA42_1)

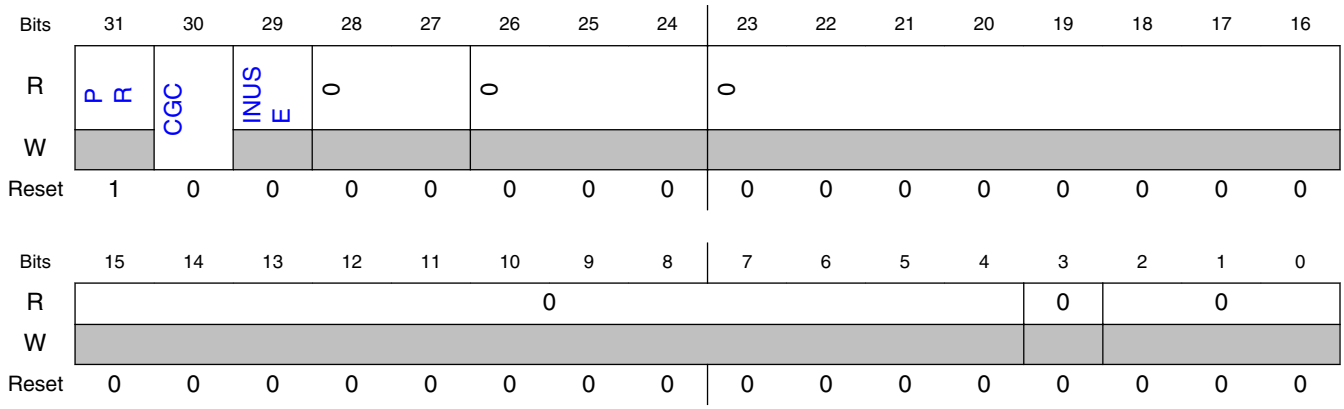
### 30.5.2.6.1 Offset

| Register | Offset |
|---|---|
| PCC_SEMA42_1 | 6Ch |

### 30.5.2.6.2 Function

This register is for the SEMA42_1 module.

### 30.5.2.6.3 Diagram




### 30.5.2.6.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.7 PCC DMAMUX1 Register (PCC_DMAMUX1)

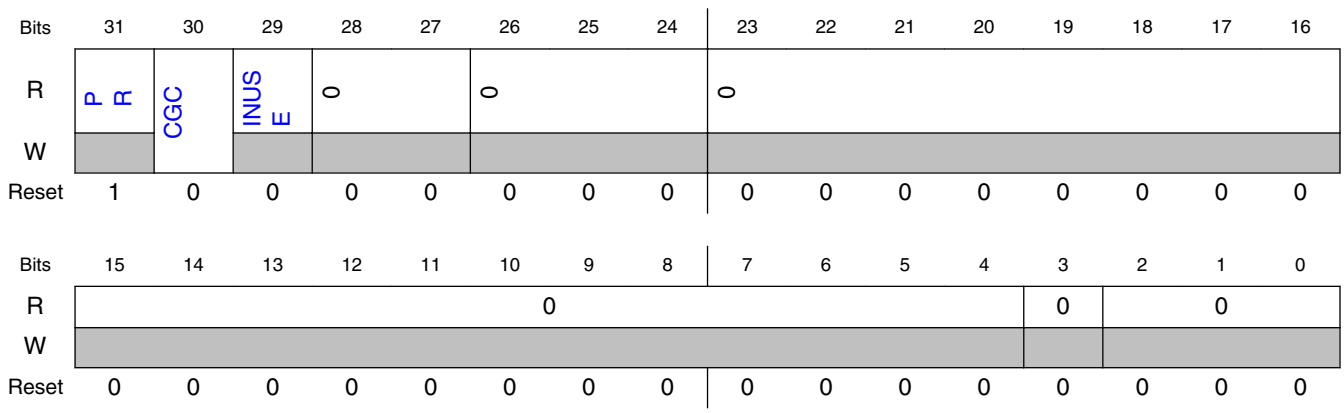
### 30.5.2.7.1 Offset

| Register | Offset |
|---|---|
| PCC_DMAMUX1 | 84h |

### 30.5.2.7.2 Function

This register is for the DMAMUX1 module.

### 30.5.2.7.3 Diagram




### 30.5.2.7.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.8 PCC INTMUX1 Register (PCC_INTMUX1)

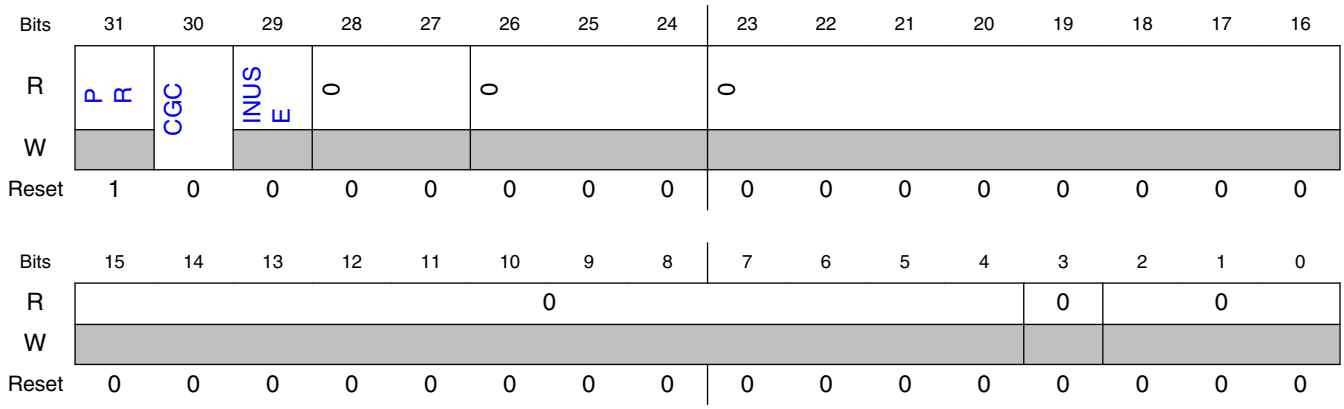
### 30.5.2.8.1 Offset

| Register | Offset |
|---|---|
| PCC_INTMUX1 | 88h |

### 30.5.2.8.2 Function

This register is for the INTMUX1 module.

## 30.5.2.8.3 Diagram




## 30.5.2.8.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.9 PCC MUB Register (PCC_MUB)

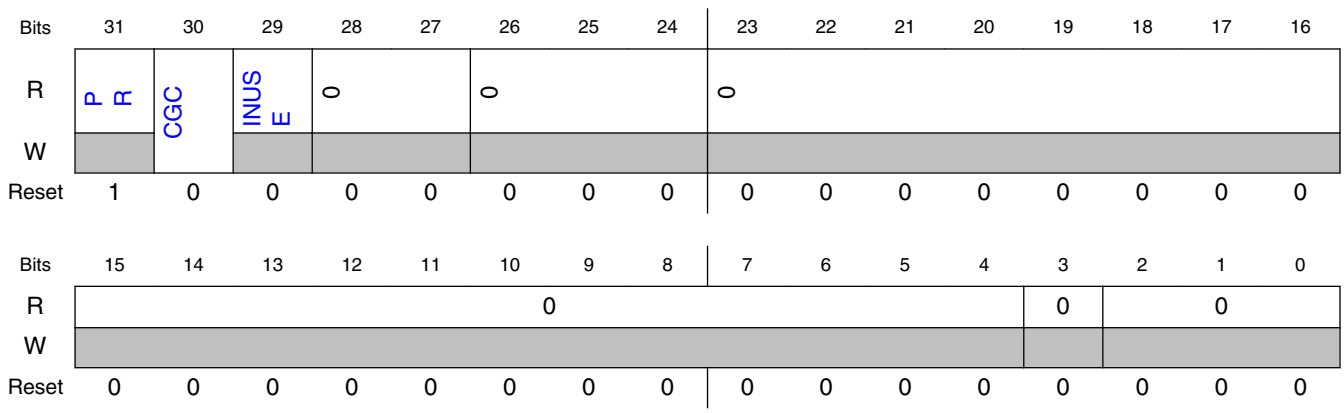
### 30.5.2.9.1 Offset

| Register | Offset |
|---|---|
| PCC_MUB | 90h |

### 30.5.2.9.2 Function

This register is for the MUB module.

### 30.5.2.9.3 Diagram




### 30.5.2.9.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.10 PCC CAU3 Register (PCC_CAU3)

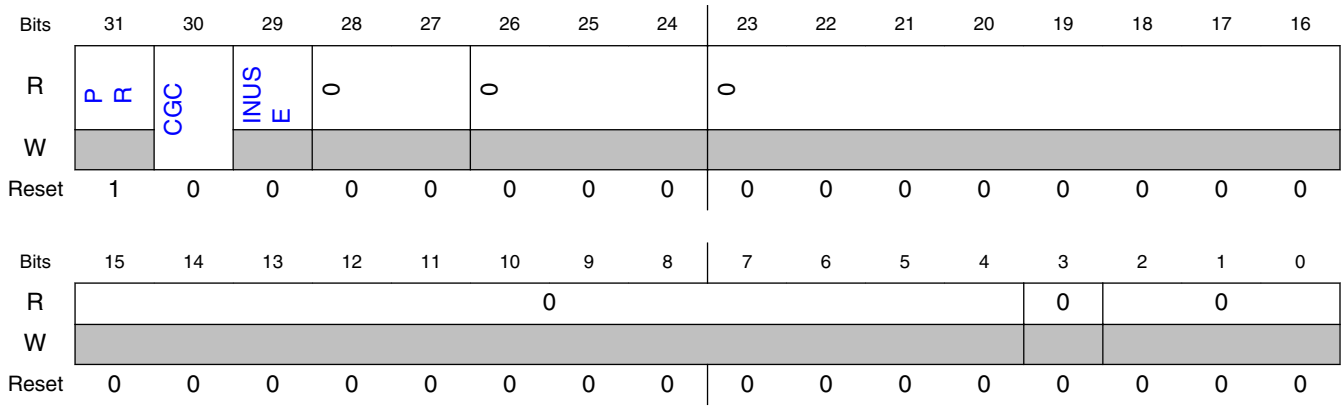
### 30.5.2.10.1 Offset

| Register | Offset |
|---|---|
| PCC_CAU3 | A0h |

### 30.5.2.10.2 Function

This register is for the CAU3 module.

### 30.5.2.10.3 Diagram




### 30.5.2.10.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.11 PCC TRNG Register (PCC_TRNG)

### 30.5.2.11.1 Offset

| Register | Offset |
|---|---|
| PCC_TRNG | A4h |

### 30.5.2.11.2 Function

This register is for the TRNG module.

### 30.5.2.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PR | CGC | INUSE | 0 | | 0 | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | 0 | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 30.5.2.11.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.12 PCC LPIT1 Register (PCC_LPIT1)

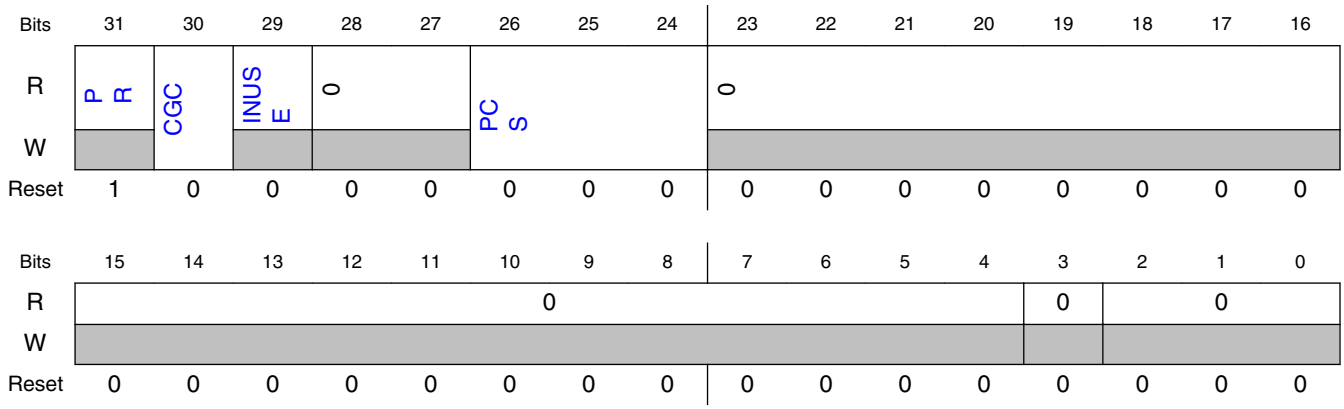
### 30.5.2.12.1 Offset

| Register | Offset |
|---|---|
| PCC_LPIT1 | A8h |

### 30.5.2.12.2 Function

This register is for the LPIT1 module.

## 30.5.2.12.3 Diagram




## 30.5.2.12.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.13 PCC TPM3 Register (PCC_TPM3)

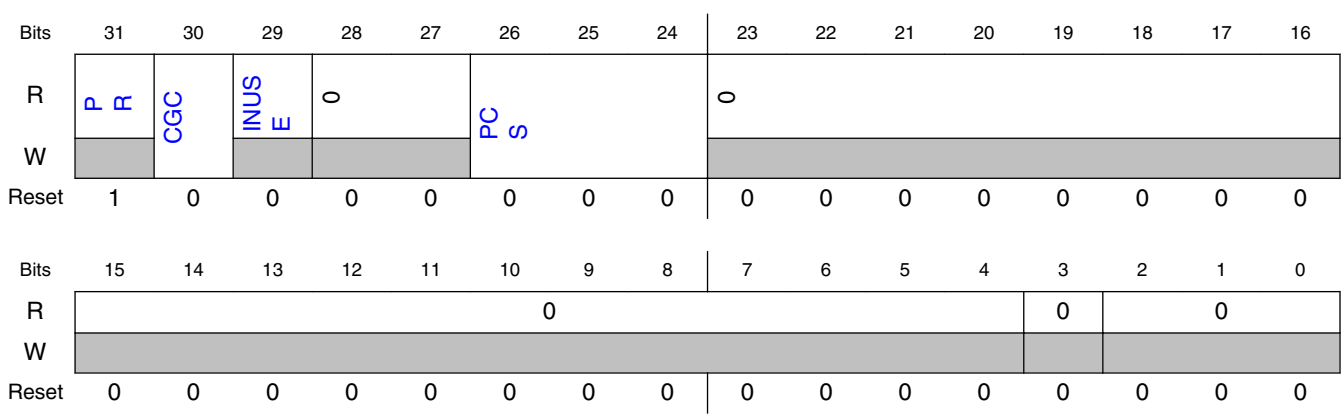
### 30.5.2.13.1 Offset

| Register | Offset |
|---|---|
| PCC_TPM3 | B4h |

### 30.5.2.13.2 Function

This register is for the TPM3 module.

### 30.5.2.13.3 Diagram



### 30.5.2.13.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.14 PCC LPI2C3 Register (PCC_LPI2C3)

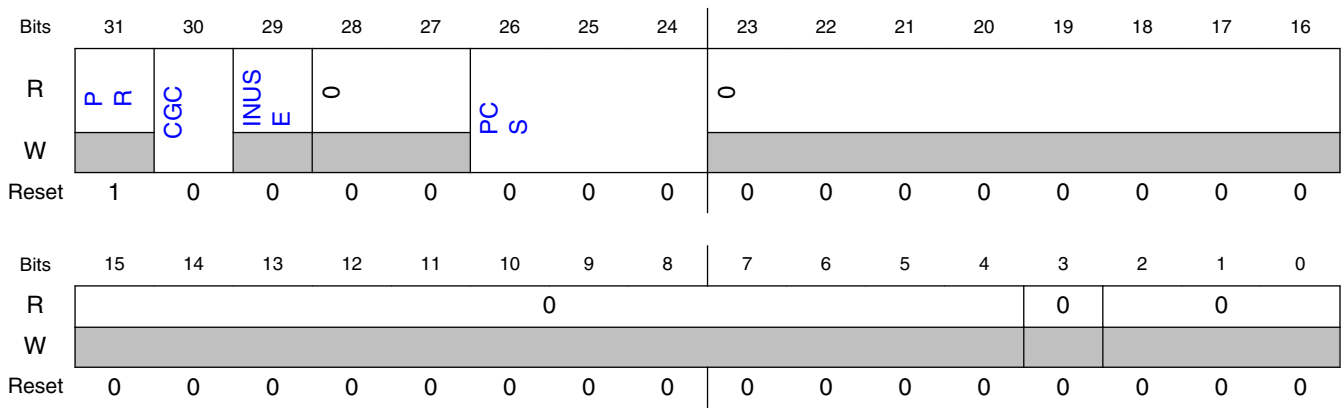
### 30.5.2.14.1 Offset

| Register | Offset |
|---|---|
| PCC_LPI2C3 | B8h |

### 30.5.2.14.2 Function

This register is for the LPI2C3 module.

### 30.5.2.14.3 Diagram




### 30.5.2.14.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.15 PCC LPSPI3 Register (PCC_LPSPI3)

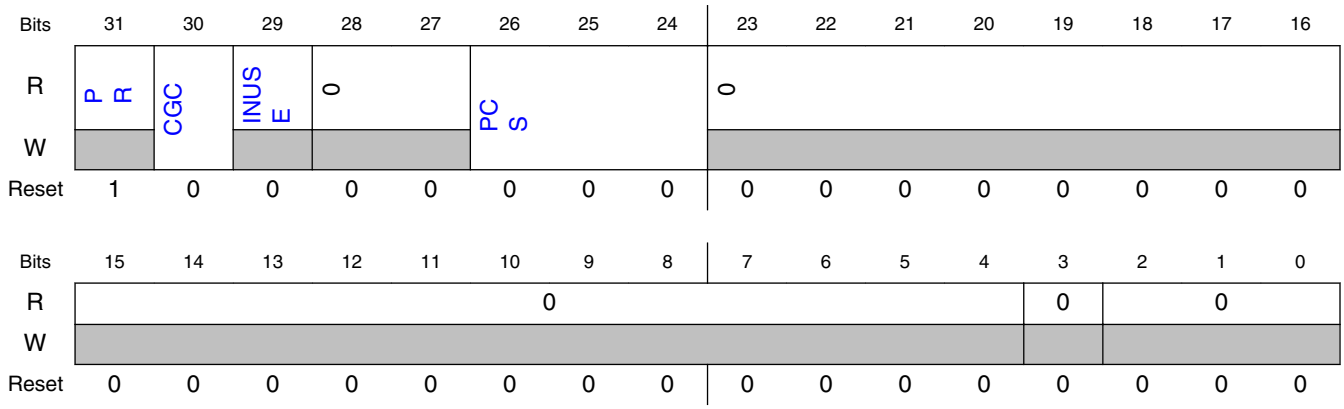
### 30.5.2.15.1 Offset

| Register | Offset |
|---|---|
| PCC_LPSPI3 | D4h |

### 30.5.2.15.2 Function

This register is for the LPSPI3 module.

## 30.5.2.15.3 Diagram




## 30.5.2.15.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.16 PCC LPUART3 Register (PCC_LPUART3)

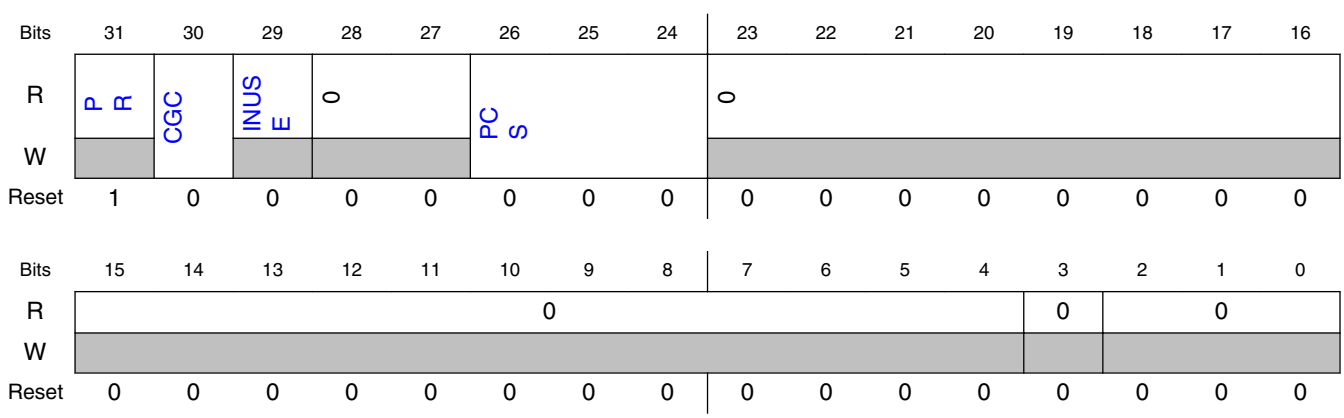
### 30.5.2.16.1 Offset

| Register | Offset |
|---|---|
| PCC_LPUART3 | D8h |

### 30.5.2.16.2 Function

This register is for the LPUART3 module.

### 30.5.2.16.3 Diagram



### 30.5.2.16.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>PCS | Peripheral Clock Source Select<br>This read/write bit field is used for peripherals that support various clock selections.<br>This field can be written only when the clock is disabled (CGC = 0). Similarly, if the INUSE flag is set, this field is locked.<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.17 PCC PORTE Register (PCC_PORTE)

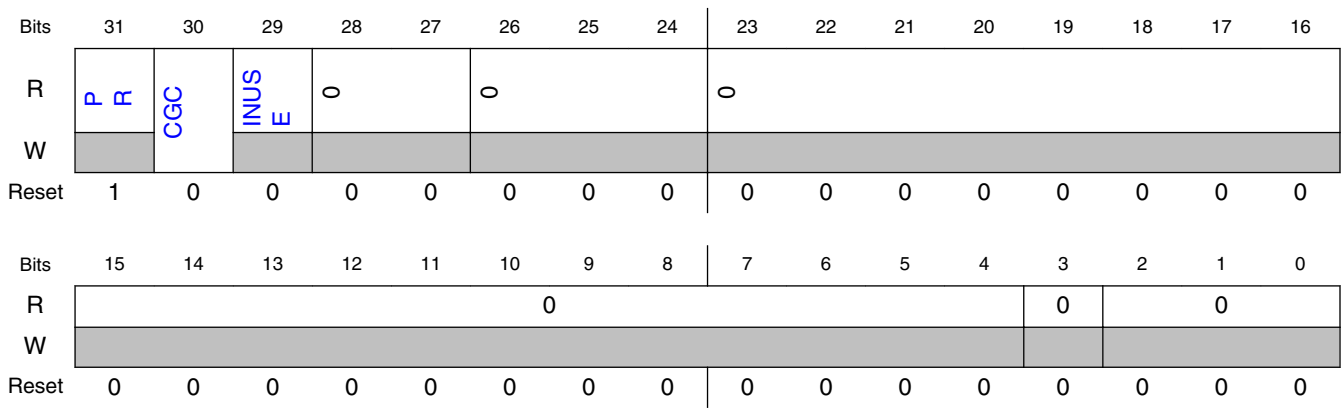
### 30.5.2.17.1 Offset

| Register | Offset |
|---|---|
| PCC_PORTE | DCh |

### 30.5.2.17.2 Function

This register is for the PORTE module.

### 30.5.2.17.3 Diagram




### 30.5.2.17.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.18 PCC MTB Register (PCC_MTB)

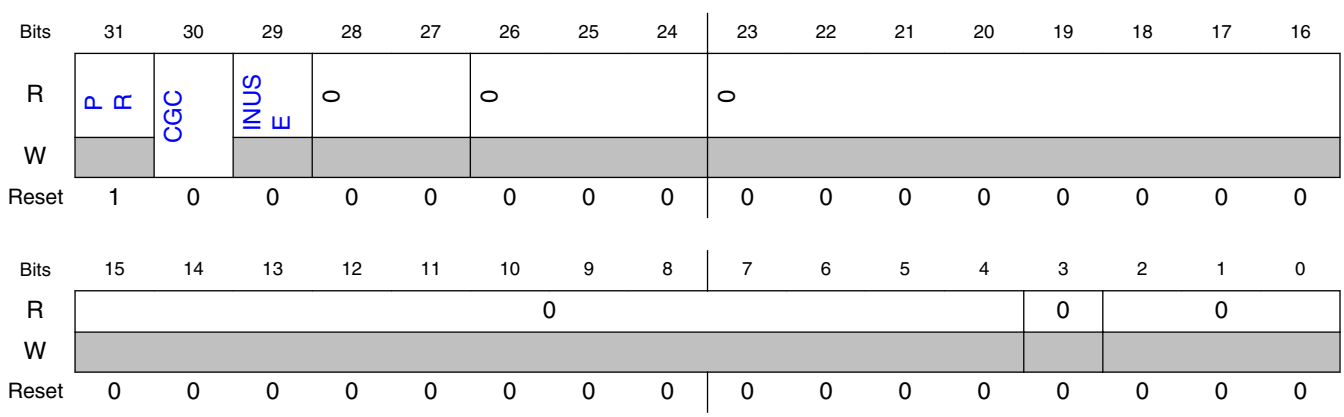
### 30.5.2.18.1 Offset

| Register | Offset |
|---|---|
| PCC_MTB | 200h |

### 30.5.2.18.2 Function

This register is for the MTB module.

### 30.5.2.18.3 Diagram



### 30.5.2.18.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

## 30.5.2.19 PCC EXT_CLK Register (PCC_EXT_CLK)

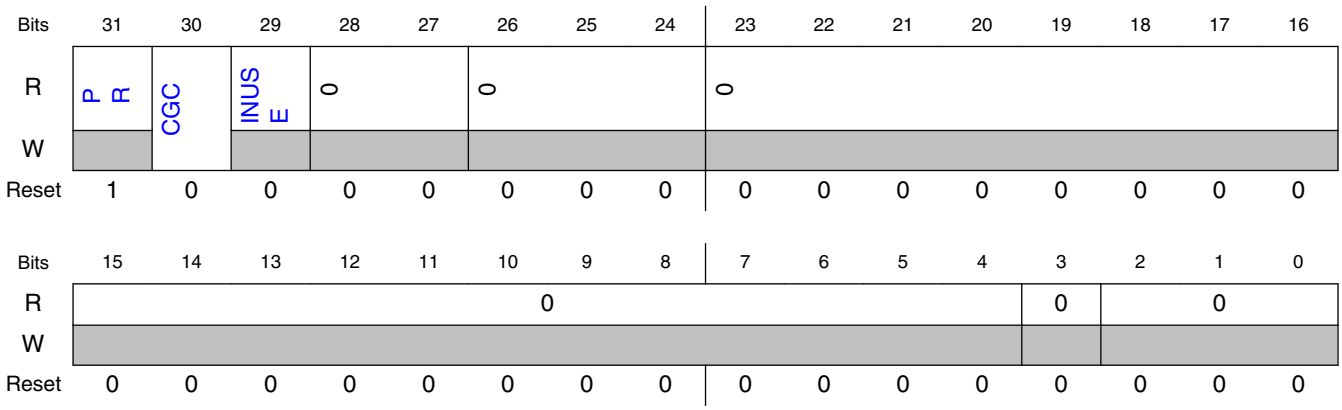
### 30.5.2.19.1 Offset

| Register | Offset |
|---|---|
| PCC_EXT_CLK | 204h |

### 30.5.2.19.2 Function

This register is for the EXT_CLK module.

### 30.5.2.19.3 Diagram




### 30.5.2.19.4 Fields

| Field | Function |
|---|---|
| 31<br>PR | Present<br>This bit shows whether the peripheral is present on this device.<br>0b - Peripheral is not present.<br>1b - Peripheral is present. |
| 30<br>CGC | Clock Gate Control<br>This read/write bit enables the interface clock for the peripheral, allowing access to the module's registers. It also controls whether the clock selection and divider options can be modified.<br>0b - Clock disabled. The current clock selection and divider options are not locked and can be modified.<br>1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified. |
| 29<br>INUSE | In use flag<br>This read-only bit shows that this peripheral is being used by another core. That is, software on another core has already configured the clocking options of this peripheral. For example, if CPU0 software has set this peripheral's CGC bit first, it sees the INUSE bit as cleared. However, if software on another core has set this peripheral's CGC bit first, CPU0 software sees the INUSE bit as set.<br>0b - Peripheral is not being used.<br>1b - Peripheral is being used. Software cannot modify the existing clocking configuration. |
| 28-27<br>— | This read-only bit field is reserved and always has the value 0. |
| 26-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23-4<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3<br>— | This read-only bit field is reserved and always has the value 0. |
| 2-0<br>— | This read-only bit field is reserved and always has the value 0. |

# Chapter 31
# Power Management

## 31.1 Introduction

This chapter describes various chip power modes and functionality of the each modules in these modes.

## 31.2 Power supply and power domain summary

This section describes the power supply and the power domain of this device.

### 31.2.1 Power supply

This device requires a 1.71 V to 3.6 V $V_{DD}$ operating voltage supply. Several independent supplies can be provided to specific peripherals.

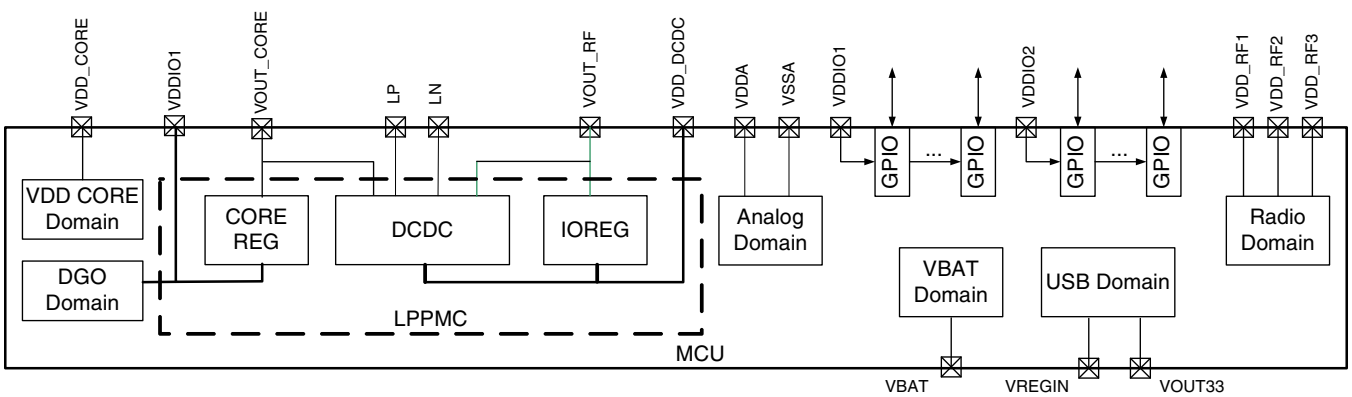



**Figure 31-1. Power supply overview**

#### 31.2.1.1 VDD_DCDC

VDD_DCDC is the power supply input for DCDC and 1.8 V IOREG regulator.

#### 31.2.1.2 VDD_CORE

VDD_CORE is the power supply input for internal VDD core domain. It powers the core, memory which includes flash and SRAM, and other digital peripherals.

VDD_CORE can be from external or internal power regulator, such as DCDC or PMC internal regulator.

VDD_CORE supports input voltage range from 1.2±5% V to 1.45 V.

- When 1.3 V < VDD_CORE_IN < 1.45 V, the device can support high speed run mode, and the maximum core clock can be 72 MHz.
- When 1.14 V < VDD_CORE_IN < 1.3 V, the device can support only the normal run mode, the maximum core clock can be 48 MHz.

#### 31.2.1.3 VDDIOx

VDDIOx power the I/O rings.

- VDDIO1 powers the I/O ring 1, including PTA, PTB, PTC, and PTD.
- VDDIO2 powers the I/O ring 2, including PTE.

VDDIO1 can also power the core regulator and DGO domain.

VDDIO1 and VDDIO2 can be powered independently.

Each VDDIOx can be powered from an external power supply or from an internal IO regulator.

#### 31.2.1.4 VDDA/VSSA

VDDA is the power supply for analog peripherals, such as LPADC, VREF, and LPDAC. It is the maximum of VDDIO1 and VDDIO2.

VSSA is the analog ground for analog peripherals, such as LPADC, VREF, and LPDAC.

#### 31.2.1.5 VDD_RFx

VDD_RF1 (VDD_RFSY), VDD_RF2 (VDD_RFHF) and VDD_RF3 (VDD_RFLF) are the power supplies for the 2.4 GHz radio module.

The radio module power supplies can be powered from external source or generated by the on-chip DCDC.

VDD_RFx input voltage range is from 1.5 V to 3.6 V.

#### 31.2.1.6 VBAT

VBAT is the power supply input for peripherals in VBAT domain.

The peripheral function and register data inside VBAT domain can be maintained when all the other powers are OFF.

#### 31.2.1.7 VREGIN

VREGIN is the power supply input for USB regulator. The USB regulator output can power the USB transceiver or serve as the other power supply like VDD.

VREGIN input voltage range is from 2.7 V to 5.5 V.

#### 31.2.1.8 VOUT_CORE

VOUT_CORE is the power output from the LDO regulator (core regulator) or one of the outputs from the DCDC regulator.

The VOUT_CORE can be connected to VDD_CORE_IN as the power supply for core domain. VOUT_CORE provides an output voltage of 1.2 V to 1.4 V, and a maximum output current of 100 mA.

Both the LPPMC and DCDC voltage outputs can provide 100 mA output current. The default, after a power-on-reset, the VOUT_CORE will be from the LPPMC.

#### 31.2.1.9 VOUT_RF

VOUT_RF is the power output from the internal IO regulator (IOREG) or one of the dual outputs of the DCDC regulator. The VOUT_RF provides the supply voltage for an internal or external RF module. The VOUT_RF provides an output voltage of 1.5 V to 1.8 V, with a maximum output current of 100 mA on either the IOREG or DCDC outputs.

The VOUT_RF can be connected to VDDIO1 or VDDIO2 as the power supply for padring 1 or padring 2. The VOUT_RF can be connected to VDDA as the power supply for analog modules.

The VOUT_RF can be connected to VDD_RFx as the power supply for 2.4 GHz radio module.

#### 31.2.1.10 LP and LN

The LP and LN pins require an external inductor to be connected to the DCDC regulator module.

### 31.2.2 Power domains

The following table describe the power domain of this device.

**Table 31-1. Modules power domain summary**

| Module | Domain |
|---|---|
| Digital Tamper | $V_{BAT}$ |
| Low Voltage Power Regulator (LVPREG) | $V_{BAT}$ |
| Real Time Clock (RTC) | $V_{BAT}$ |
| Real Time Clock Oscillator (RTCOSC) | $V_{BAT}$ |
| Flash (FTFE) | $V_{DD_CORE}$ |
| $V_{BAT}$ Register File – 32 Bytes | $V_{BAT}$ |
| Low-Power Analog-to-Digital Converter 0 (LPADC0) | $V_{DDA}$ |
| 12-bit Low Power Digital-to-Analog Converter 0 (LPDAC0) | $V_{DDA}$ |
| VREF 2.1V/1.2V | $V_{DDA}$ |
| System Power Manager (SPM) Drivers | $V_{DDIO1}$ |
| Multi System Mode Controller (MSMC) | $V_{DDIO1}$ (DGO) |
| Low-Leakage Wake-up Unit 0 (LLWU0) | $V_{DDIO1}$ (DGO) |
| Low-Leakage Wake-up Unit 1 (LLWU1) | $V_{DDIO1}$ (DGO) |
| Low-Power Frequency Locked Loop (LPFLL) | $V_{DDIO1}$ (DGO) |
| Fast Internal Resister/Capaciter Oscillator (FIRC) | $V_{DDIO1}$ (DGO) |

*Table continues on the next page...*

## Table 31-1. Modules power domain summary (continued)

| Module | Domain |
|---|---|
| Slow Internal Resister/Capaciter Oscillator (SIRC) | $V_{DD_CORE}$ |
| Low-Power Timer 0 (LPTMR0) | $V_{DDIO1}$ (DGO) |
| Low-Power Timer 1 (LPTMR1) | $V_{DDIO1}$ (DGO) |
| Low-Power Timer 2 (LPTMR2) | $V_{DDIO1}$ (DGO) |
| System Power Management (SPM) | $V_{DDIO1}$ (DGO) |
| GPIO pins Port A, B, C, D | $V_{DDIO1}$ |
| Low-Power Comparator 0 (LPCMP0) | $V_{DDIO1}$ (DGO) |
| GPIO pins Port E | $V_{DDIO2}$ |
| Low-Power Comparator 1 (LPCMP1) | $V_{DDIO2}$ (DGO) |
| Input/Output Regulator (IOREG) | $V_{DD_DCDC}$ |
| Joint Test Action Group (JTAG) | $V_{DD_CORE}$ |
| Serial Wire Debug version 2.0 (SWD) | $V_{DD_CORE}$ |
| External Watchdog Monitor (EWM) | $V_{DD_CORE}$ |
| System Clock Generator (SCG) | $V_{DD_CORE}$ |
| AXBS | $V_{DD_CORE}$ |
| Resource Domain Controller (RDC) | $V_{DD_CORE}$ |
| Read-Only Memory 0 (ROM0) | $V_{DD_CORE}$ |
| ROM Resource Domain Controller MRC | $V_{DD_CORE}$ |
| M4 AIPS 0 | $V_{DD_CORE}$ |
| M0+ AIPS 0 | $V_{DD_CORE}$ |
| M4 Core | $V_{DD_CORE}$ |
| M0+ Core | $V_{DD_CORE}$ |
| M4 Instruction/Data Cache (LPLMEM) 4kB | $V_{DD_CORE}$ |
| M0+ Instruction/Data Cache (LPLMEM) 4kB | $V_{DD_CORE}$ |
| Asynchronous Wake-up Interrupt Controller (AWIC) | $V_{DD_CORE}$ |
| Memory Mapped Division / Square Root (MMDVSQ) | $V_{DD_CORE}$ |
| SRAM | $V_{DD_CORE}$ |
| Direct Memory Access Controller 0 (DMA0) - 16ch | $V_{DD_CORE}$ |
| Direct Memory Access Controller 1 (DMA1) - 8ch | $V_{DD_CORE}$ |
| Direct Memory Access Multiplexer 0 (DMAMUX0) | $V_{DD_CORE}$ |
| Direct Memory Access Multiplexer 1 (DMAMUX1) | $V_{DD_CORE}$ |
| Peripheral Clock Control (PCCx) | $V_{DD_CORE}$ |
| Watchdog 0 (WDOG0) | $V_{DD_CORE}$ |
| Watchdog 1 (WDOG1) | $V_{DD_CORE}$ |
| True Random Number Generator (TRNG) | $V_{DD_CORE}$ |
| Cryptographic Acceleration Engine version 3 (CAEv3) | $V_{DD_CORE}$ |
| Flexible Input/Ouput 0 (FlexIO0) | $V_{DD_CORE}$ |
| Low-Power Inter-Integrated Circuit 0 (LPI2C0) | $V_{DD_CORE}$ |
| Low-Power Inter-Integrated Circuit 1 (LPI2C1) | $V_{DD_CORE}$ |

*Table continues on the next page...*

**Table 31-1. Modules power domain summary (continued)**

| Module | Domain |
|---|---|
| Low-Power Inter-Integrated Circuit 2 (LPI2C2) | $V_{DD_CORE}$ |
| Low-Power Inter-Integrated Circuit 3 (LPI2C3) | $V_{DD_CORE}$ |
| Low-Power Serial Peripheral Interface 0 (LPSPI0) | $V_{DD_CORE}$ |
| Low-Power Serial Peripheral Interface 1 (LPSPI1) | $V_{DD_CORE}$ |
| Low-Power Serial Peripheral Interface 2 (LPSPI2) | $V_{DD_CORE}$ |
| Low-Power Serial Peripheral Interface 3 (LPSPI3) | $V_{DD_CORE}$ |
| Low-Power Universal Asynchronous Receive/Transmit 0 (LPUART0) | $V_{DD_CORE}$ |
| Low-Power Universal Asynchronous Receive/Transmit 1 (LPUART1) | $V_{DD_CORE}$ |
| Low-Power Universal Asynchronous Receive/Transmit 2 (LPUART2) | $V_{DD_CORE}$ |
| Low-Power Universal Asynchronous Receive/Transmit 3 (LPUART3) | $V_{DD_CORE}$ |
| Low-Power Periodic Interrupt Timer 0 (LPIT0) | $V_{DD_CORE}$ |
| Low-Power Periodic Interrupt Timer 1 (LPIT1) | $V_{DD_CORE}$ |
| Timer / PWM Module 0 (TPM0) - 6ch | $V_{DD_CORE}$ |
| Timer / PWM Module 1 (TPM1) - 2ch | $V_{DD_CORE}$ |
| Timer / PWM Module 2 (TPM2) - 6ch | $V_{DD_CORE}$ |
| Timer / PWM Module 3 (TPM3) - 2ch | $V_{DD_CORE}$ |
| Rapid GPIO (RGPIO) | $V_{DD_CORE}$ |
| PORTA – E logic | $V_{DD_CORE}$ |
| Trigger Multiplexer 0 (TRGMUX0) | $V_{DD_CORE}$ |
| Time Stamp Timer Module (TSTMR) | $V_{DD_CORE}$ |
| USB0 | $V_{DD_CORE}$ |
| USB SRAM | $V_{DD_CORE}$ |
| Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) | $V_{DD_CORE}$ |
| Serial Audio Interface 0 (SAI) | $V_{DD_CORE}$ |
| 2.4 GHz Radio Digital Logic | $V_{DD_CORE}$ |
| Single Inductor Multiple Ouput DC/DC (SIMO DCDC) | $V_{DD_DCDC}$ |
| 2.4 GHz Radio Analog | $V_{DD_RF}$ |
| Radio OSC | $V_{DD_RF}$ |
| USB VREG | $V_{REGIN}$ |
| USB LS/FS Tranceiver | $V_{OUT33}$ |

## 31.3 On-chip regulator summary

This section describes the various on-chip regulators of this device.

### 31.3.1 DCDC regulator

The dual output DCDC regulator requires one single external inductor to generate two outputs. The external inductor must be connected between LP and LN.

The DCDC input power supply is from VDD_DCDC. It supports input voltage range from 2.1 V to 4.25 V.

One output is configurable from 1.0 V to 1.4 V. It is connected to the VOUT_CORE. The other output is configurable from 1.5 V to 1.8 V. It is connected to the VOUT_RF.

### 31.3.2 LPPMC regulator

The LPPMC integrates a 1.2 V output linear regulator.

The power input to this regulator is from VDDIO1 and ranges from 1.71 V to 3.6 V.

The output of this regulator is connected to the VOUT_CORE which shares the output pin with one output of the DCDC.

### 31.3.3 IOREG regulator

The IOREG is a linear regulator that provides a 1.8 V or 1.5 V output. Its output is connected to VOUT_RF.

This IOREG regulator's power input is from VDD_DCDC. It supports input voltage range from 1.95 V to 4.25 V.

The VOUT_RF is shared with one of the outputs of the DCDC.

### 31.3.4 USB regulator

The USB voltage regulator module is a LDO linear voltage regulator to provide 3.3 V power from an input power supply varying from 2.7 V to 5.5 V. It is the power supply for USB domain. Its input is VREGIN and its output is VOUT33.

## 31.4 Typical power supply schematic

This section describes the chip power modes and functionality of the each modules in these modes.

### 31.4.1 Typical power supply schematic 1

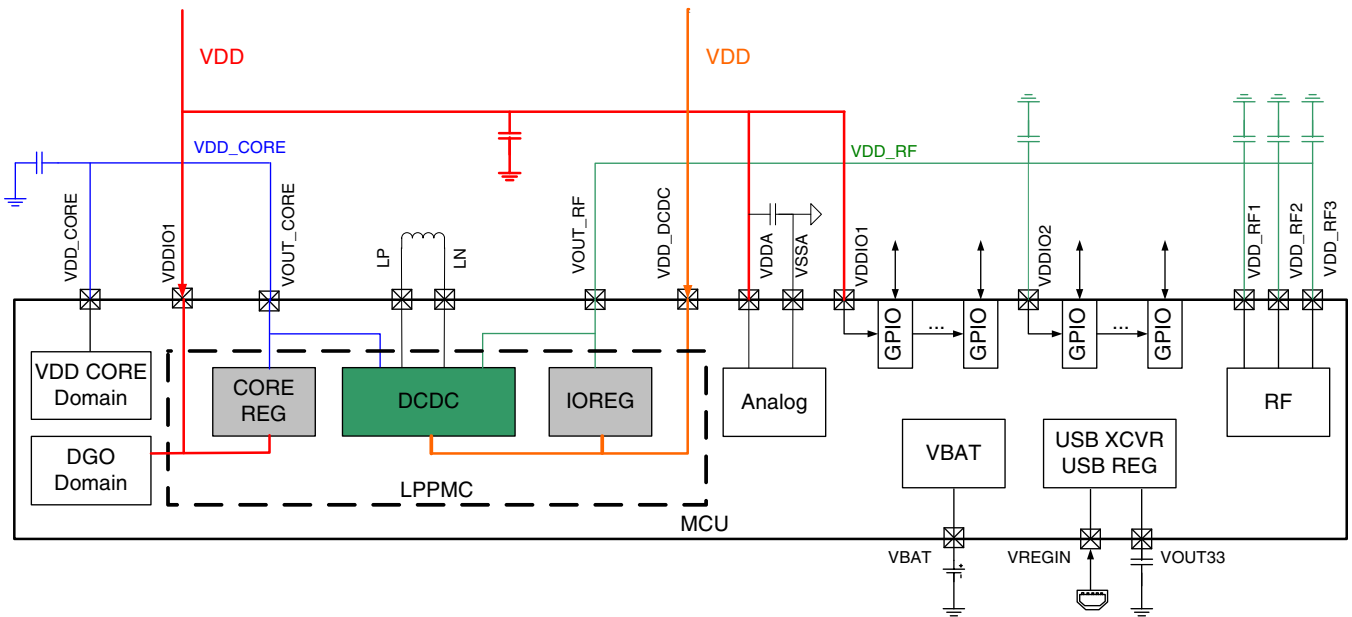



**Figure 31-2. MCU core powered by DCDC**

In this configuration, the VDDIO1 input provides power to the DGO domain modules, and the GPIO pins powered by VDDIO. These GPIO pins interface with 1.8 V to 3.6 V circuits. The LPPMC is also powered by the VDDIO1 input pin and from a POR, the LPPMC provides the VDD_CORE supply. After POR, the application code places the LPPMC in a standby/powered off mode.

The VDD_DCDC input provides power to the DCDC and the IOREG. The MCU's core ( CPU platform and SoC logic) is powered from the DCDC which provides a high efficiency buck output from VDD_DCDC input.

The other output of the DCDC provides VOUT_RF which powers the radio module, via the VDD_RF1/2/3 inputs. The VOUT_RF is also used to power GPIO pins that are powered via the VDDIO2 input. These GPIO pins interface with 1.71 V to 1.8 V circuits.

## 31.4.2 Typical power supply schematic 2

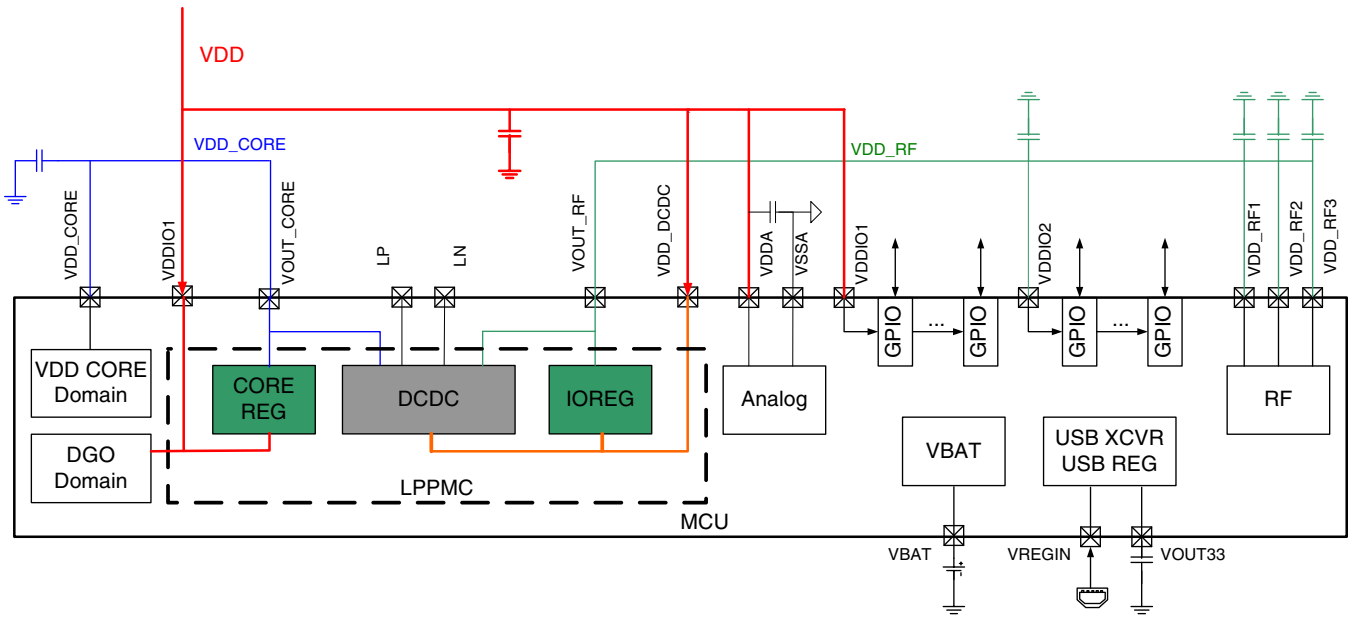



**Figure 31-3. MCU core powered by LPPMC with DCDC bypassed**

In this configuration, VDDIO1, VDD_DCDC, and VDDA are supplied with the same external voltage source in the range of 1.8 V -3.6 V. The LPPMC provides the VOUT_CORE which powers the core and the SoC logic. The IOREG provides the VOUT_RF. Both the radio module and the GPIO powered by VDDIO2 are powered from the VOUT_RF output from the IOREG. This would be a typical pin configuration for lower pin count devices and simpler designs with two external voltage sources.

### 31.4.3 Typical power supply schematic 3

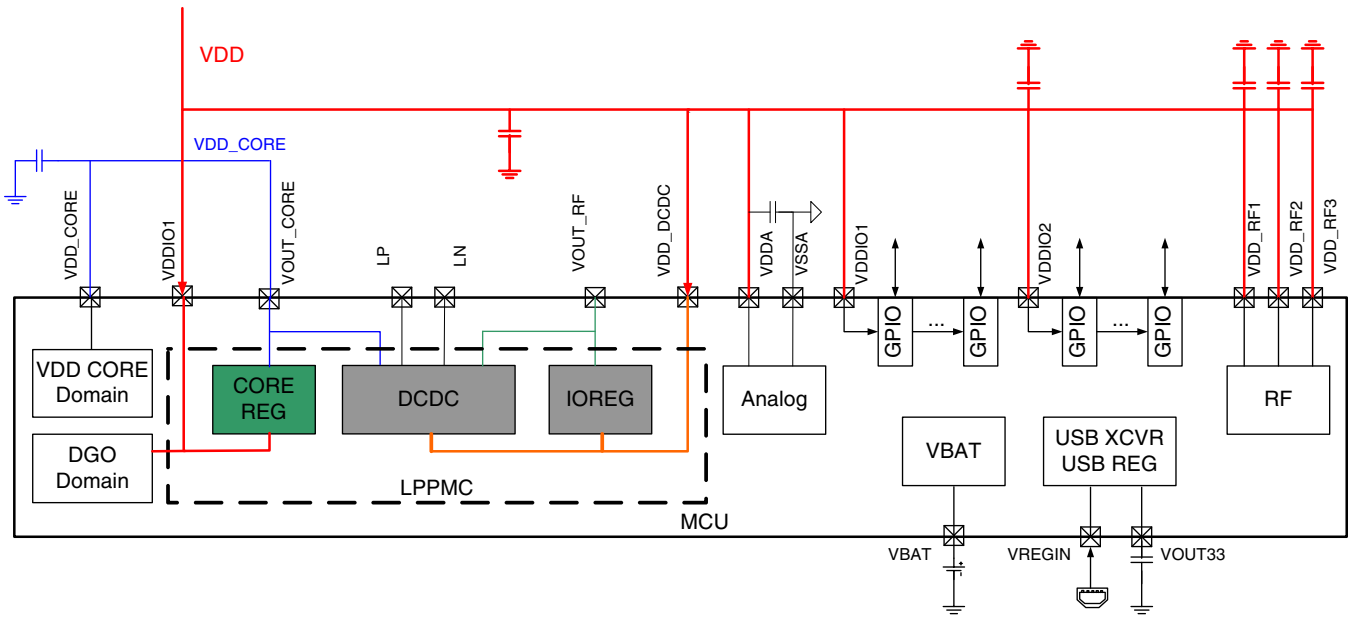



**Figure 31-4. MCU powered by one external supply**

This configuration is based on a simple application where only one external voltage supply is available.

## 31.5 Power up state

When power supply is applied on VDD_DCDC, the DCDC is bypassed by default. The VOUT_CORE is supplied by the LPPMC 1.2 V regulator and the VOUT_RF is supplied by the 1.8 V IOREG. The DCDC can be enabled as the main regulator.

## 31.6 Multi-Core power mode description

This device includes CM4F and CM0+ subsystem as well as RF subsystem.

For CM4F/CM0+ power mode, each includes HSRUN, RUN, PSTOP, STOP, VLPR/ VLPW/VLPS, LLS, VLLS, etc. Please refer to Power modes for details.

For RF power mode description, it includes RRUN, RLLS, RVLLS power modes. Please refer to Section Radio Power for details.

## 31.6.1 Core power structure overview

The following diagram is the overview of core power structure for this device.

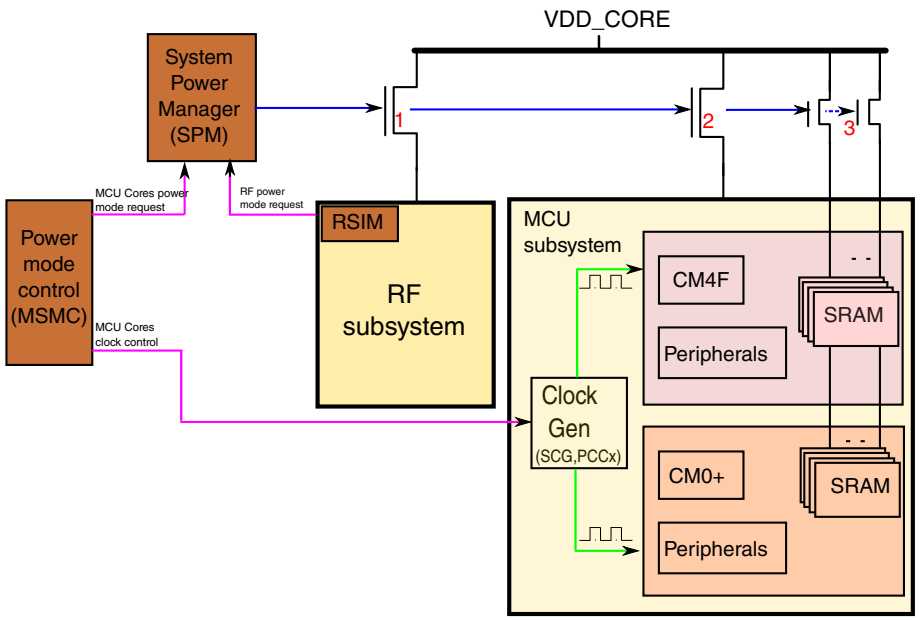



**Figure 31-5. VDD_CORE power structure**

Generally speaking, RF and MCU digital parts are powered from the power source (VDD_CORE) through independent power switches (1 and 2 in the diagram). The SPM controls them according to the CM4F/CM0+ power mode configuration in MSMC and RF power mode configuration in RSIM.

Besides above power control, clock control is the second level of power mode implementation. The clock generation modules like SCG, PCC0 and 1 can automatically switch ON/OFF clocks for CM4F and CM0+ subsystem based on their power modes.

Also separate power switches (3 in the diagram) are used to control each SRAM power in specific MCU low-power modes. See details in Power modes.

## 31.6.2 Power mode combinations

There are various power modes available across the subsystems for the user. The system power management (SPM) deals with the power mode requests from RF and MCU subsystems and select appropriate power control scheme. Clock control is implemented as fine optimization. Usually there are different targets across subsystem like high-performance in CM4F and low-power in RF/CM0+. To meet whole system target, there are general principles to determine whole system power status according to MCU/RF power mode. Below lists two general principles.

- Individual power switch control applies for MCU and RF. Basically it allows MCU and RF power mode to be independent. However, due to sharing same power source, some schemes adopted in the low-power mode, like voltage level adjustment in one subsystem, could be disabled if another subsystem needs to work at different voltage level; but the other irrelevant low-power schemes are kept. It might cause the power larger than the expectation. For example, if RF in RRUN mode, the total power is almost the same between MCU in VLPR mode and RUN with 8 MHz. In this case, MCU VLPR is fake VLPR mode to maintain RF in RRUN mode by keeping normal voltage level which is the same in MCU RUN mode.
- The same power supply applies for whole MCU subsystem including CM4F and CM0+ part. The clock control switches on/off clocks to account for the majority of MCU power difference. The MCU power mode is determined by the highest one from CM4F and CM0+ power modes with following order: HSRUN > RUN/STOP/PSTOP > VLPR/VLPW/VLPS > LLS > VLLS. For example, when CM4F in RUN while CM0+ in VLPS, MCU power mode is in RUN mode. In this case, trivial power difference could be observed with CM0+ in VLPS or Stop mode when CM4F is in RUN mode. Because both VLPS and Stop mode are the same to disable all clocks to have the same clock control behavior.

Generally speaking, it is two step determination. In the first step, it's clock control according to CM4F and CM0+ power mode to determine MCU expected power mode. In the second step, it's power control based on MCU and RF power mode to determine system power mode.

The following tables list all power mode combinations and corresponding power behaviors.

**Table 31-2. MCU power mode combinations**

| CM4F power mode[1] | CM0+ power mode[2] | Max CM4F core clock frequency | Max CM0+ core clock frequency | MCU power mode[3] |
|---|---|---|---|---|
| RUN | RUN | 48MHz | 48MHz | RUN |
| RUN | HSRUN | 72MHz | 72MHz | HSRUN |
| RUN | VLPR | 48MHz | 48MHz | RUN |
| RUN | VLPW | 48MHz | OFF | RUN |
| RUN | VLPS | 48MHz | OFF | RUN |
| RUN | STOP/PSTOP | 48MHz | OFF | RUN |
| RUN | LLS | 48MHz | OFF | RUN |
| RUN | VLLS | 48MHz | OFF | RUN |
| HSRUN | RUN | 72MHz | 72MHz | HSRUN |
| HSRUN | HSRUN | 72MHz | 72MHz | HSRUN |
| HSRUN | VLPR | 72MHz | 72MHz | HSRUN |
| HSRUN | VLPW | 72MHz | OFF | HSRUN |

*Table continues on the next page...*

## Table 31-2. MCU power mode combinations (continued)

| CM4F power mode[1] | CM0+ power mode[2] | Max CM4F core clock frequency | Max CM0+ core clock frequency | MCU power mode[3] |
|---|---|---|---|---|
| HSRUN | VLPS | 72MHz | OFF | HSRUN |
| HSRUN | STOP/PSTOP | 72MHz | OFF | HSRUN |
| HSRUN | LLS | 72MHz | OFF | HSRUN |
| HSRUN | VLLS | 72MHz | OFF | HSRUN |
| VLPR | RUN | 48MHz | 48MHz | RUN |
| VLPR | HSRUN | 72MHz | 72MHz | HSRUN |
| VLPR | VLPR | 8MHz | 8MHz | VLP |
| VLPR | VLPW | 8MHz | OFF | VLP |
| VLPR | VLPS | 8MHz | OFF | VLP |
| VLPR | STOP/PSTOP | 48MHz | OFF | RUN |
| VLPR | LLS | 8MHz | OFF | VLP |
| VLPR | VLLS | 8MHz | OFF | VLP |
| VLPW | RUN | OFF | 48MHz | RUN |
| VLPW | HSRUN | OFF | 72MHz | HSRUN |
| VLPW | VLPR | OFF | 8MHz | VLP |
| VLPW | VLPW | OFF | OFF | VLP |
| VLPW | VLPS | OFF | OFF | VLP |
| VLPW | STOP/PSTOP | OFF | OFF | RUN |
| VLPW | LLS | OFF | OFF | VLP |
| VLPW | VLLS | OFF | OFF | VLP |
| VLPS | RUN | OFF | 48MHz | RUN |
| VLPS | HSRUN | OFF | 72MHz | HSRUN |
| VLPS | VLPR | OFF | 8MHz | VLP |
| VLPS | VLPW | OFF | OFF | VLP |
| VLPS | VLPS | OFF | OFF | VLP |
| VLPS | STOP/PSTOP | OFF | OFF | RUN |
| VLPS | LLS | OFF | OFF | VLP |
| VLPS | VLLS | OFF | OFF | VLP |
| STOP/PSTOP | RUN | OFF | 48MHz | RUN |
| STOP/PSTOP | HSRUN | OFF | 72MHz | HSRUN |
| STOP/PSTOP | VLPR | OFF | 8MHz | RUN |
| STOP/PSTOP | VLPW | OFF | OFF | RUN |
| STOP/PSTOP | VLPS | OFF | OFF | RUN |
| STOP/PSTOP | STOP/PSTOP | OFF | OFF | RUN |
| STOP/PSTOP | LLS | OFF | OFF | RUN |
| STOP/PSTOP | VLLS | OFF | OFF | RUN |
| LLS | RUN | OFF | 48MHz | RUN |
| LLS | HSRUN | OFF | 72MHz | HSRUN |
| LLS | VLPR | OFF | 8MHz | VLP |

*Table continues on the next page...*

## Table 31-2. MCU power mode combinations (continued)

| CM4F power mode[1] | CM0+ power mode[2] | Max CM4F core clock frequency | Max CM0+ core clock frequency | MCU power mode[3] |
|---|---|---|---|---|
| LLS | VLPW | OFF | OFF | VLP |
| LLS | VLPS | OFF | OFF | VLP |
| LLS | STOP/PSTOP | OFF | OFF | RUN |
| LLS | LLS | OFF | OFF | LLS |
| LLS | VLLS | OFF | OFF | LLS |
| VLLS | RUN | OFF | 48MHz | RUN |
| VLLS | HSRUN | OFF | 72MHz | HSRUN |
| VLLS | VLPR | OFF | 8MHz | VLP |
| VLLS | VLPW | OFF | OFF | VLP |
| VLLS | VLPS | OFF | OFF | VLP |
| VLLS | STOP/PSTOP | OFF | OFF | RUN |
| VLLS | LLS | OFF | OFF | LLS |
| VLLS | VLLS | OFF | OFF | VLLS |

1. It is configured by SMC0_PMCTRL register of MSMC.
2. It is configured by SMC1_PMCTRL register of MSMC.
3. It can be read out from SMCx_PMCSTAT register of MSMC. Please note the PMSTAT[7:0] bits in both SMC0_PMCSTAT and SMC1_PMCSTAT registers refer to MCU power mode so their value are identical. The STOPSTAT[31:24] bits in SMCx_PMCSTAT register reflect separate CM4F or CM0+ clock status, so can be different.

## Table 31-3. System power mode

| MCU power mode[1] | RF power mode[2] | System power mode | |
|---|---|---|---|
| | | MCU power status | RF power status |
| RUN | RRUN | RUN | RRUN |
| RUN | RLLS | RUN | RLLS |
| RUN | RVLLS | RUN | RVLLS |
| HSRUN | RRUN | HSRUN | RRUN |
| HSRUN | RLLS | HSRUN | RLLS |
| HSRUN | RVLLS | HSRUN | RVLLS |
| VLP | RRUN | RUN | RRUN |
| VLP | RLLS | VLP | RLLS |
| VLP | RVLLS | VLP | RVLLS |
| LLS | RRUN | LLS | RRUN |
| LLS | RLLS | LLS | RLLS |
| LLS | RVLLS | LLS | RVLLS |
| VLLS | RRUN | VLLS | RRUN |
| VLLS | RLLS | VLLS | RLLS |
| VLLS | RVLLS | VLLS | RVLLS |

1. It can be read out from SMCx_PMCSTAT register of MSMC. Please note the PMSTAT[7:0] bits in both SMC0_PMCSTAT and SMC1_PMCSTAT registers refer to MCU power mode so their value are identical. The STOPSTAT[31:24] bits in SMCx_PMCSTAT register reflect separate CM4F or CM0+ clock status, so can be different.

2. It is configured by RSIM Power Control register. The user can read out relative bit in this register to determine RF power mode status.

**NOTE**

When RF OSC is used in MCU subsystem, RF must be in RRUN to keep RF OSC alive.

Due to the multi-core power mode combination, users must be careful to select appropriate power modes to meet the target. Please keep in mind – it takes some time for power mode transition depending on which kinds of power mode transition is ongoing. To guarantee the entry or exiting procedure complete, it's recommended to read relative power status register mentioned above.

**NOTE**

The stack and all interrupt handlers must be placed in Flash / RAM memory local to the core that will access the stack space or execute the interrupt vectors. In addition, all functions that execute the WFI or WFE instruction must also be placed in Flash / RAM memory local to the core that will execute the function.

To help the user debug these kinds of procedures via external debugger, this device also puts all key power status information (power mode, regulator mode, power supply status, etc) in MDM-AP System Power Status Register. Please refer to MDM-AP system Power Status register in debug chapter for details.

## 31.6.3 Recommended power mode combination

Considering the complexity of power design, below power mode combinations in Table 31-2 are recommended to easy the user for start-up. It's simplified into several combinations to either achieve high performance or low-power goal in one subsystem with minimum impact on the rest of this device.

**Table 31-4. Recommended power mode combination**

| CM4F power mode | CM0+ power mode | RF allowed mode |
|---|---|---|
| HSRUN | HSRUN | RRUN, RLLS, RVLLS |
| HSRUN | STOP/PSTOP | RRUN, RLLS, RVLLS |
| STOP/PSTOP | HSRUN | RRUN, RLLS, RVLLS |
| RUN | RUN | RRUN, RLLS, RVLLS |
| RUN | STOP/PSTOP | RRUN, RLLS, RVLLS |
| STOP/PSTOP | RUN | RRUN, RLLS, RVLLS |
| STOP/PSTOP | STOP/PSTOP | RRUN, RLLS, RVLLS |

*Table continues on the next page...*

**Table 31-4. Recommended power mode combination (continued)**

| CM4F power mode | CM0+ power mode | RF allowed mode |
|---|---|---|
| VLPR | VLPR/VLPW/VLPS | RLLS, RVLLS |
| VLPR/VLPW/VLPS | VLPR | RLLS, RVLLS |
| VLPS | VLPS | RLLS, RVLLS |
| LLS | LLS | RLLS, RVLLS |
| VLLS | VLLS | RVLLS |

## 31.7 Power mode description

### 31.7.1 RUN mode

RUN mode is the default mode after reset.

In RUN mode, user is allowed to adapt following different configurations to balance the performance and power.

#### 31.7.1.1 HSRUN

When the VIN_CORE voltage is higher than 1.3 V, the MCU VDD core domain is fully powered. The MCU can run at a faster frequency compared with in the normal RUN mode.

#### 31.7.1.2 VLPR

The regulator is in low power mode with limited output capability. Thus the core, system, bus, and flash clock maximum frequencies are restricted in this mode.

#### 31.7.1.3 Computation operation

Compute operation is an execution or compute-only mode of operation that keeps the CPU enabled with full access to the SRAM and flash read port, but places all other bus masters and bus slaves into their stop mode.

## 31.7.2 Wait mode

Wait mode allows peripherals to function while the core is in sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked.

### 31.7.2.1 VLPW

The regulator is in low power mode with limited output capability. The core clock is gated off. The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted.

## 31.7.3 Stop mode

Stop mode places chip in static state. LVD in low power modes depends on user configuration and MCU/RF arbitration.

### 31.7.3.1 PSTOP1

When configured for PSTOP1, both the system clock and bus clock are gated. All bus masters and bus slaves enter Stop mode, but the clock generators in the MCG and the on-chip regulator in the PMC remain in Run (or VLP Run) mode. Exit from PSTOP1 can be initiated by a reset or an asynchronous interrupt from a bus master or bus slave. If configured, an asynchronous DMA request can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP1.

### 31.7.3.2 PSTOP2

When configured for PSTOP2, only the core and system clocks are gated and the bus clock remains active. The bus masters and bus slaves clocked by the system clock enter Stop mode, but the bus slaves clocked by bus clock remain in Run (or VLP Run) mode. The clock generators in the MCG and the on-chip regulator in the PMC also remain in Run (or VLP Run) mode. Exit from PSTOP2 can be initiated by a reset, an asynchronous interrupt from a bus master or bus slave clocked by the system clock, or a synchronous interrupt from a bus slave clocked by the bus clock. If configured, a DMA request (using the asynchronous DMA wakeup) can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP2.

#### 31.7.3.3 PSTOP3

When configured for PSTOP3, only the core clocks are gated and the system and bus clocks remain active. The bus masters and bus slaves remain in Run (or VLP Run) mode. The clock generators in the SCG and the on-chip regulator in the SPM also remain in Run (or VLP Run) mode. Exit from PSTOP3 can be initiated by a reset, or a synchronous interrupt from a bus master or slave.

#### 31.7.3.4 VLPS

The regulator is in low power mode with limited output capability. The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.

### 31.7.4 Standby (LLS) mode

Standby mode is the state retention power mode.It permits following configurations to achieve various functionalities and power levels.

#### 31.7.4.1 SRAM retention

Each SRAM array can be optional configured to retain its content.

### 31.7.5 Shutdown (VLLSx) mode

Shutdown mode permits the lowest power level. Most peripherals are disabled (powered OFF). Only some modules can be used.

It permits following configurations to achieve various functionalities and power levels.

#### 31.7.5.1 SRAM retention

Each SRAM array can be optional configured to retain its content in VLLS2/3 modes.

#### 31.7.5.2 LPO off

The 1 kHz LPO clock can be optional off to further reduce the power consumption in shutdown mode.

#### 31.7.5.3 POR off

The POR circuit can be optional off to further reduce the power consumption in shutdown mode.

## 31.8 Entering and exiting power modes

Low power modes are entered by executing the WFI or WFE instruction. Which particular mode (Wait, STOP, Standby, or Shutdown) is entered depends on the configuration of the SMC and the status of the other core and Radio. For more information on this, refer to the SMC descriptions in the Multi System Mode Controller (MSMC), Table 31-2 and Table 31-3.

Upon execution of the WFI or WFE command, the core executing the command will internally signal all of its peripherals and wait for an acknowledge signal before fully moving into the commanded low power mode. Since the two cores execute code independently and have their own peripherals on independent crossbars, the low power entry times may not be exactly the same in complex applications that use multiple peripherals. This is important to remember and account for when designing and troubleshooting your applications.

Exiting low power modes differs depending on the mode entered. For WAIT (WAIT and VLPW) and STOP (STOP and VLPS) modes, the exit mechanism is via interrupt if the WFI instruction was executed or if the WFE instruction was executed when SEVONPEND = 1. For Standby (LLS) and Shutdown (VLLSx) modes, the wakeup sources are limited to LLWU generated wakeups (for a complete list of possible LLWU wakeup sources, consult the LLWU chapter), NMI_b, or RESET_b pin assertions. When the NMI_b pin or RESET_b pin has been disabled through the associated FTFE_FOPT settings, then these pins are ignored as wakeup sources. The wakeup flow from Shutdown or from RESET_b is always through reset. The wakeup flow from Standby will be the next instruction cycle or through the IRQ handler of the interrupt that causes the wakeup provided it is properly configured and interrupts are not globally masked.

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode. Failure to do this may result in the low power mode not being entered correctly.

On Shutdown recoveries, the I/O pins continue to be held in a static state after code execution begins, allowing software to reconfigure the system before unlocking the I/O. RAM can be optionally retained.

## 31.9 Power mode transitions

The following figure shows the power mode transitions. Any reset always brings the chip back to the normal run state. The Standby and Shutdown modes are the lowest power stop modes based on amount of logic or memory that is required to be retained by the application.

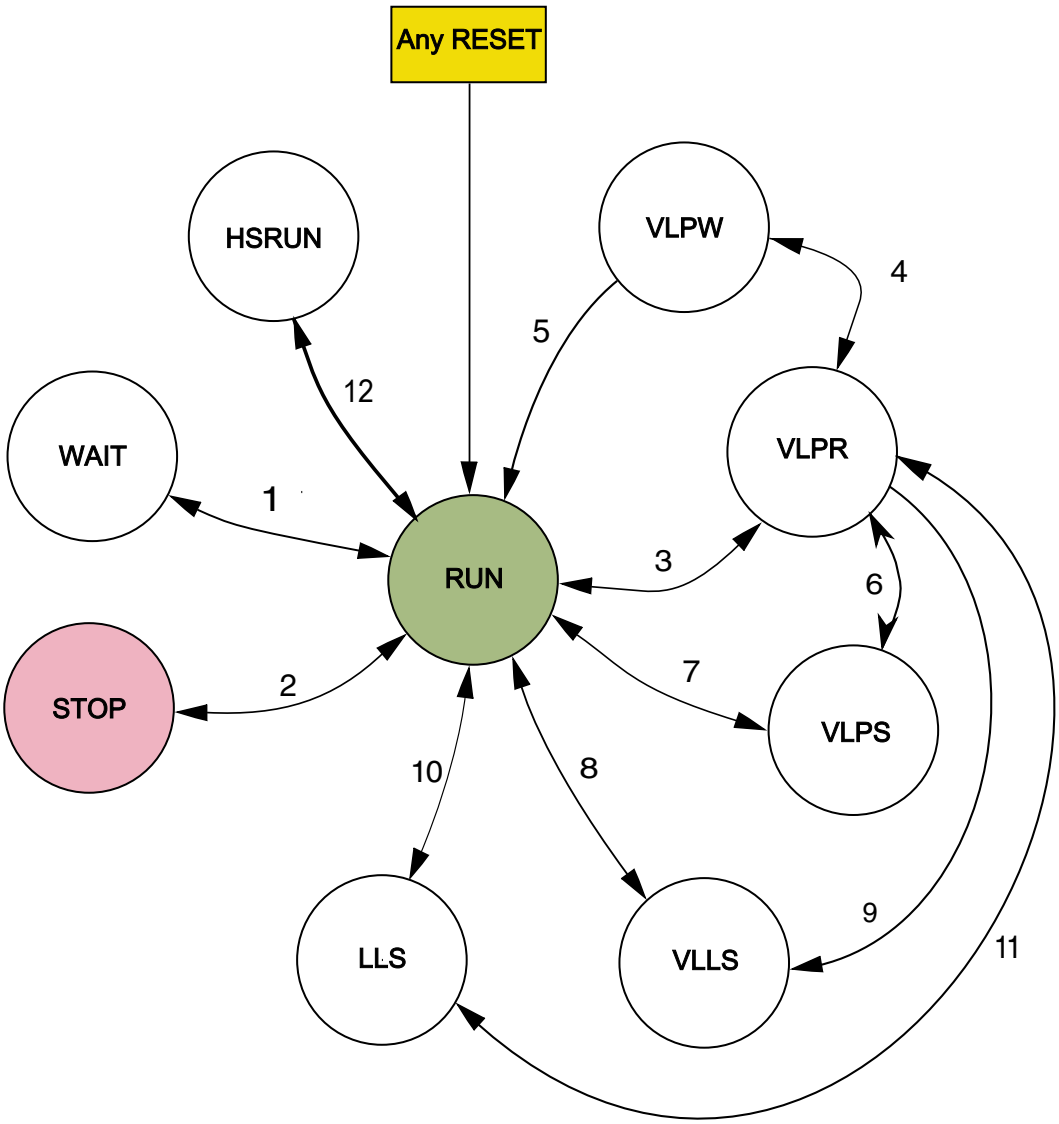



**Figure 31-6. Power mode state transition diagram**

## 31.10 Power modes shutdown sequencing

When entering Stop or other low-power modes, the clocks are shut off in an orderly sequence to safely place the chip in the targeted low-power state. All low-power entry sequences are initiated by the core executing a WFI instruction. The ARM core's outputs, SLEEPDEEP and SLEEPING, trigger entry to the various low-power modes:

- System level wait and VLPW modes equate to: SLEEPING & !SLEEPDEEP
- All other low power modes equate to: SLEEPING & SLEEPDEEP

When entering the non-wait modes, the chip performs the following sequence:

- Shuts off core clock and system clock to the ARM Cortex-M0+ core and Cortex-M4 core immediately.
- Polls stop acknowledge indications from the non-core crossbar masters (DMA), supporting peripherals (SPI, PIT, TRNG, etc.,) and the flash controller for indications that system clocks, bus clock and/or flash clock need to be left enabled to complete a previously initiated operation, effectively stalling entry to the targeted low power mode. When all acknowledges are detected, system clock, bus clock and flash clock are turned off at the same time.
- SCG and mode controller shut off clock sources and/or the internal supplies driven from the on-chip regulator as defined for the targeted low power mode.

In Wait mode, most of the system clocks are not affected by the low power mode entry. The core clock to the ARM Cortex-M0+ core and/or the ARM Cortex-M4 core is shut off. Some modules support stop-in wait functionality and have their clocks disabled under these configurations.

The debugger modules support a transition from Stop, Wait, VLPS, and VLPW back to a halted state when the debugger is enabled. This transition is initiated by setting the Debug Request bit in MDM-AP control register. As part of this transition, system clocking is re-established and is equivalent to normal Run/VLPR mode clocking configuration.

## 31.11 Flash program restrictions

The flash memory on this device must not be programmed or erased while operating in VLPR mode or HSRUN mode.

## 31.12 Module operation in low power modes

The following table illustrates the functionality of each modules while the chip is in each of the low power modes. The standard behavior is shown with some exceptions for Compute Operation (CPO) and Partial Stop2 (PSTOP2).

(Debug modules are discussed separately; see Debug in low power modes.) Number ratings (such as 2 MHz and 1 Mbit/s) represent the maximum frequencies or maximum data rates per mode. Also, these terms are used:

- FF = Full functionality. In VLPR and VLPW, the system frequency is limited, but if a module does not have a limitation in its functionality, it is still listed as FF.
- Async = Fully functional with alternate clock source, provided the selected clock source remains enabled
- SR = Module state is retained but not functional.

- LP = Module operates in a lower power state
- OFF = Modules are powered off or disabled.

**Table 31-5. Module operation in low power modes**

| Modules | VLPR | VLPW | STOP | VLPS | LLS | VLLS2/3 | VLLS0/1 |
|---|---|---|---|---|---|---|---|
| Core modules | | | | | | | |
| NVIC | FF | FF | SR | SR | SR | OFF | OFF |
| System modules | | | | | | | |
| MSMC | FF | FF | FF | FF | SR/FF[1] | OFF/FF[2] | OFF/FF[2] |
| LLWUx | FF | FF | FF | FF | FF | FF | FF |
| Core regulator (1.2 V) | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] |
| IOREG 1.8 V | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] | Optional[3] |
| HVD/LVD | Optional[4] | Optional[4] | Optional[4] | Optional[4] | Optional[4] | Optional[4] | Optional[4] |
| POR (Brown-Out) | FF | FF | FF | FF | FF | FF | Optional[5] |
| DMA | FF | FF | Async | Async | SR | OFF | OFF |
| Watchdog | FF | FF | Optional on with available clock source | Optional on with available clock source | SR | OFF | OFF |
| Clocks | | | | | | | |
| 1 kHz LPO | FF | FF | FF | FF | FF | FF | Optional[6] |
| Radio OSC | Optional[7] | Optional[7] | Optional[7] | Optional[7] | Optional[7] | Optional[7] | Optional[7] |
| SCG | 8 MHz SIRC | 8 MHz SIRC | static – SIRC, FIRC, LPFLL optional | static – SIRC optional | static | OFF | OFF |
| Core clock | FF | OFF | OFF | OFF | OFF | OFF | OFF |
| Plat clock | FF | FF | OFF | OFF | OFF | OFF | OFF |
| Bus clock | FF | FF | OFF | OFF | OFF | OFF | OFF |
| Slow clock | FF | FF | OFF | OFF | OFF | OFF | OFF |
| Memory | | | | | | | |
| Flash | 1 MHz No program | LP | LP | LP | SR | The registers are retained | OFF |
| SRAM | LP | LP | SR | Optional[8] | Optional[8] | Optional[8] | OFF |
| LPLMEM (Cache) | FF | FF | SR | SR | Optional retention | Optional[8] | OFF |
| LPCAC (Cache) | FF | FF | SR | SR | Optional retention | Optional[8] | OFF |
| System Register File | FF | FF | FF | FF | FF | FF | FF |
| VBAT Register File | Optional[9] | Optional[9] | Optional[9] | Optional[9] | Optional[9] | Optional[9] | Optional[9] |
| Communication Interface | | | | | | | |
| USB | static | static | static | static | static | OFF | OFF |
| USB Reg | Optional | Optional | Optional | Optional | Optional | Optional | Optional |
| LPUARTx | 1 Mbit/s | 1 Mbit/s | Async | Async | static | OFF | OFF |

*Table continues on the next page...*

## Table 31-5. Module operation in low power modes (continued)

| Modules | VLPR | VLPW | STOP | VLPS | LLS | VLLS2/3 | VLLS0/1 |
|---|---|---|---|---|---|---|---|
| LPSPIx | Master 2 Mbit/s<br>Slave 1 Mbit/s | Master 2 Mbit/s<br>Slave 1 Mbit/s | Static, slave mode receive | Static, slave mode receive | static | OFF | OFF |
| LPI2Cx | 1 Mbit/s | 1 Mbit/s | Async | Async | static | OFF | OFF |
| I2S | FF | FF | Async | Async | static | OFF | OFF |
| EMVSIM | FF | FF | Static, card detect wakeup | Static, card detect wakeup | static | OFF | OFF |
| Timer modules | | | | | | | |
| LPTPMx | FF | FF | Async | Async | static | OFF | OFF |
| LPIT | FF | FF | Async | Async | static | OFF | OFF |
| LPTMRx | FF | FF | Async | Async | Async | FF | FF |
| RTC | FF | FF | Async | Async | Async | Optional[9] | Optional[9] |
| TSTMR | FF | FF | static | static | static | OFF | OFF |
| Security modules | | | | | | | |
| CAU3 | FF | FF | static | static | static | OFF | OFF |
| CRC | FF | FF | static | static | static | OFF | OFF |
| Digital Tamper | FF | FF | Async | Async | Async | Optional[9] | Optional[9] |
| TRNG | FF | FF | static | static | static | OFF | OFF |
| Analog | | | | | | | |
| LPADC | FF | FF | LPADC internal clock | LPADC internal clock | SR | OFF | OFF |
| LPCMP0 | FF | FF | FF | FF or Low power Nano mode Compare | FF or Low power Nano mode Compare | FF | FF[10] |
| LPCMP1 | FF | FF | FF | FF or Low power Nano mode Compare | FF or Low power Nano mode Compare | Optional[11] | Optional[11] |
| 6-bit DAC | FF | FF | static | static | static | OFF | OFF |
| 12-bit LPDAC | FF | FF | static | static | static | OFF | OFF |
| VREF | FF | FF | static | static | static | OFF | OFF |
| HMI | | | | | | | |
| GPIO | FF | FF | Static output, wake up input | Static output, wake up input | Static, pins latched | OFF, pins latched | OFF, pins latched |
| FlexIO | FF | FF | Async | Async | SR | OFF | OFF |

1. SR for reset related control and FF for power model related control
2. OFF for reset related control and FF for power model related control
3. It depends on MCU/RF arbitration and SPM LPSEL/BGEN bit.
4. It depends on MCU/RF arbitration and SPM LVDEN/BGEN/ALLREFN bit
5. It depends on MCU/RF arbitration and SPM POREN bit.
6. It depends on MCU/RF arbitration and SPM LPOEN/ALLREFN bit.

7. It depends on MCU/RF configuration and arbitration.
8. It depends on the configurations, see Table 1 for details.
9. Requires the VBAT supply to be properly powered.
10. It depends on SPM_CORELPCNFG[ALLREFEN] =1.
11. It depends on SPM_CORESC[VDDIOOVRIDE] = 1, SPM_CORELPCNFG[ALLREFEN] =1, and VDDIO2.

# Chapter 32
# Multi System Mode Controller (MSMC)

## 32.1 Chip-specific MSMC information

### 32.1.1 SRAM blocks with corresponding control bit

See SRAM memory map for the control bits of SRAMLPR and SRAMDSR.

### 32.1.2 Power mode transition in dual core application

In dual core applications, either core can initiate a transition to HSRUN mode. However, once in HSRUN, the same core that initiated the entry to HSRUN must also initiate the transition back to normal run mode.

When transitioning to Deep Sleep power modes (STOP mode and below), both cores must execute the WFI or WFE instruction. When this happens, the respective platform of each core will not necessarily move to the commanded low power mode at the same time. This is due to peripherals attached to said platform must acknowledge the low power mode transition command. Likewise, the platforms may not exit the low power mode at exactly the same time either. It is for this reason that all interrupt handlers and stack space for each core must reside in the respective core's local Flash / RAM memory space. Inter-domain accesses are possible but in some cases, a hard fault exception can be thrown as one core can become active and attempt to access memory in the inactive platform. Likewise, any function that executes the WFI or WFE command should be placed in local Flash / RAM memory.

## 32.2 Introduction

The Multi-System Mode Controller (MSMC) is responsible for sequencing the MCU into and out of all stop and run power modes.

Specifically, it monitors events to trigger transitions between power modes while controlling the power, clocks, and memories of the MCU to achieve the power consumption and functionality of that mode. Additionally, the MSMC will arbitrate between multiple cores in the MCU to provide each with the most optimal power mode without negatively impacting the functionality of other cores.

This chapter describes all the available power modes, the sequence followed to enter/exit each mode, and the functionality available while in each of the modes.

The following diagram illustrates the connections of the MSMC with other components in the system that are necessary to sequence the system through all power modes.

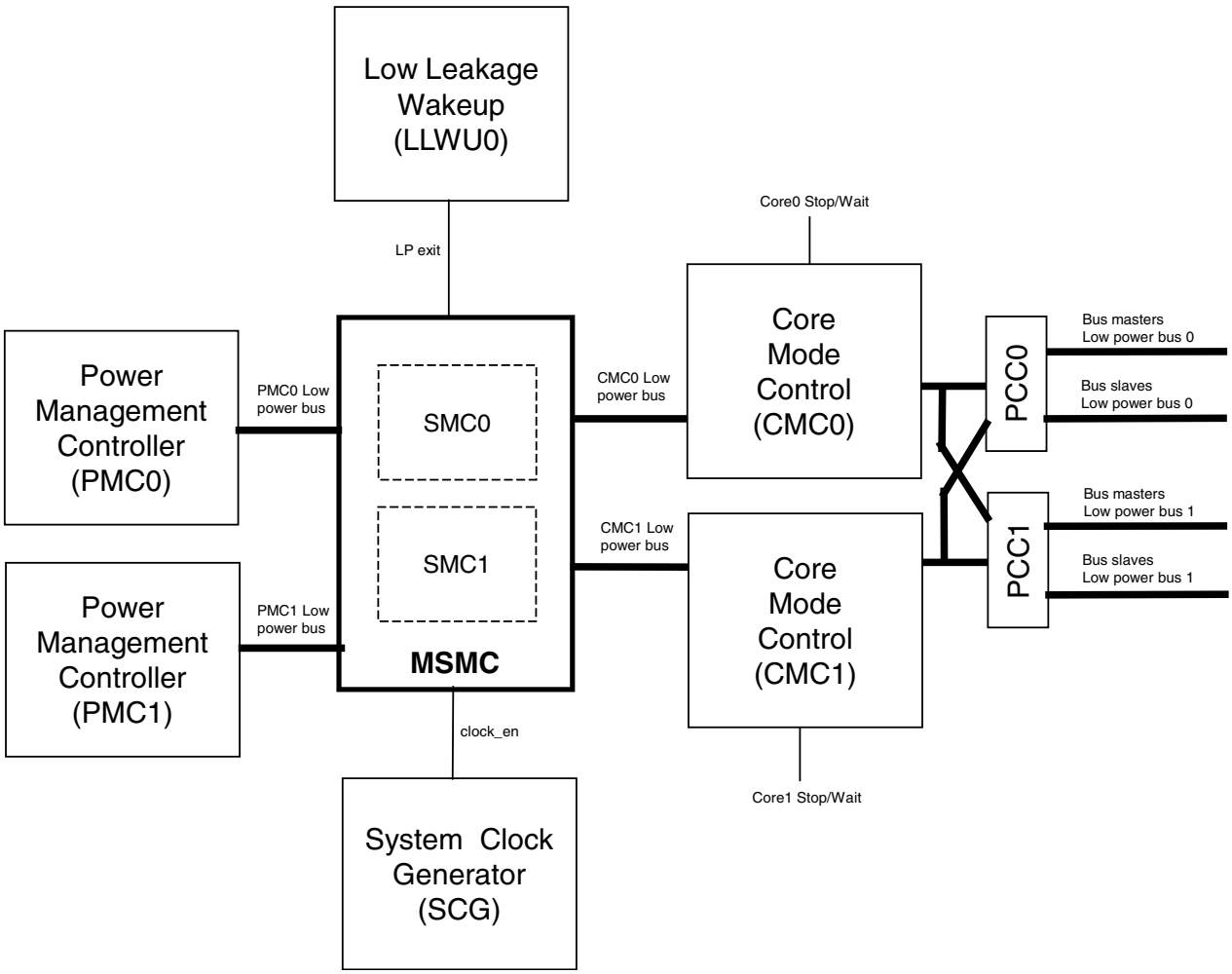



**Figure 32-1. Power management system components and connections**

## 32.3 Modes of operation

An ARM Cortex-M CPU has three primary modes of operation:

- Run
- Sleep
- Deep Sleep

The WFI or WFE instruction is used to invoke a Sleep, Deep Sleep or Standby modes for either Cortex family. Run, Wait, and Stop are the common terms used for the primary operating modes of the MSMC.

The following table shows the translation between the ARM CPU modes and MSMC power modes:

| ARM CPU mode | MSMC mode |
|---|---|
| Sleep | Wait |
| Deep Sleep | Stop |

In addition, the MSMC also augments Stop, Wait, and Run modes in a number of ways. Depending on the needs of the user application, a variety of run and stop modes are available that allow the user to adjust functionality, performance and power consumption.

The following table describes the power modes available to each core within the MCU.

**Table 32-1. Power modes**

| Mode | Description |
|---|---|
| Run (RUN) | The core can be run at full frequency and all functionality is available. This mode is also referred to as Normal Run mode. |
| High Speed Run (HSRUN) | Provides the highest performance mode at the fastest supported frequency. The core, system, bus, and flash clock maximum frequencies are unrestricted in this mode.[1] |
| Wait (WAIT) | Allows a core to enter a static, low power state with instant wakeup time, while still allowing peripherals to operate with full functionality. The core clock is gated off. The system clock continues to operate. Bus clocks, if enabled, continue to operate. |
| Stop (STOP) | Allows a core and its peripherals to enter a static, low power state with relatively fast wakeup times, while still allowing peripherals to operate with full asynchronous functionality. The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. |
| Very Low Power Run (VLPR) | Provides the lowest power operating mode to the core at a reduced frequency. The core, system, bus, and flash clock maximum frequencies are restricted in this mode.[1] |
| Very Low Power Wait (VLPW) | Allows a core to enter a static, very low power state, while still allowing peripherals to operate with full functionality at a reduced frequency. The core clock is gated off. The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted.[1] |
| Very Low Power Stop (VLPS) | Allows a core and its peripherals to enter a static, very low power state with relatively fast wakeup times, while still allowing peripherals to operate with limited asynchronous functionality. The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. |
| Low Leakage Stop (LLS) | Allows a core and its peripherals to enter a low leakage, power gated state with relatively fast wakeup times, while still retaining the state of all logic. The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. Logic and I/O are retained, with wakeup only supported via the LLWU, NMI or Reset pins. |
| Very Low Leakage Stop (VLLS) | Allows a core and its peripherals to enter a very low leakage, power gated state.The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. Logic is power gated, with wakeup only supported via the LLWU, NMI or Reset pins. I/O states are latched. |

1. See the devices' chip configuration details for the maximum allowable frequencies.

## 32.4 Memory map and register descriptions

Information about the registers related to the system mode controller can be found here.

## 32.4.1 SMC0 register descriptions

Different MSMC registers reset on different reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

**NOTE**

The MSMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**NOTE**

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode. Failure to do this may result in the low power mode not being entered correctly.

### 32.4.1.1 SMC0 Memory map

SMC0 base address: 4002_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0200_00ABh |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0000h |
| 8h | Power Mode Protection register (PMPROT) | 32 | RW | See description. |
| 10h | Power Mode Control register (PMCTRL) | 32 | RW | See description. |
| 18h | Power Mode Status register (PMSTAT) | 32 | RW | See description. |
| 20h | System Reset Status (SRS) | 32 | RO | 0000_0006h |
| 24h | Reset Pin Control (RPC) | 32 | RW | 0000_0000h |
| 28h | Sticky System Reset Status (SSRS) | 32 | W1C | 0000_0006h |
| 2Ch | System Reset Interrupt Enable (SRIE) | 32 | RW | 0000_0000h |
| 30h | System Reset Interrupt Flag (SRIF) | 32 | W1C | 0000_0000h |
| 40h | Mode Register (MR) | 32 | W1C | 0000_0000h |
| 50h | Force Mode Register (FM) | 32 | RW | 0000_0000h |
| 60h | SRAM Low Power Register (SRAMLPR) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 64h | SRAM Deep Sleep Register (SRAMDSR) | 32 | RW | 0000_0000h |

## 32.4.1.2 Version ID Register (VERID)

### 32.4.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 32.4.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

### 32.4.1.2.3 Fields

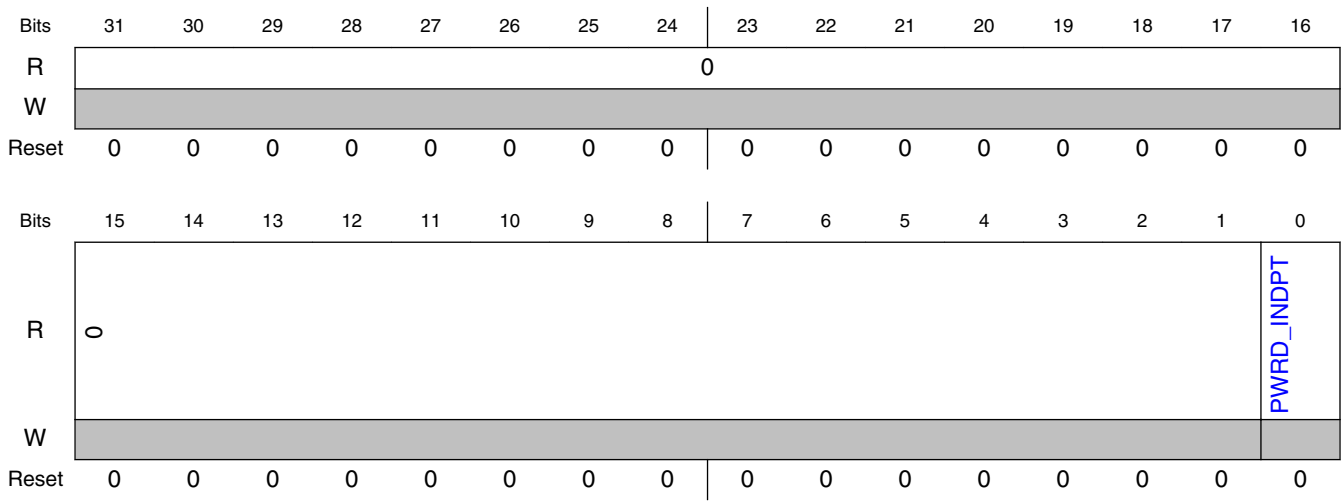
| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000010101011b - Default features supported |

# 32.4.1.3 Parameter Register (PARAM)

## 32.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

## 32.4.1.3.2 Diagram




## 32.4.1.3.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>PWRD_INDPT | Power Domains Independent<br>When set, means each core is able to enter power modes independently of the other. When clear, means certain dependancies are enforced by the MSMC before each core is able to enter some power modes. See "Power mode transitions" section for detailed requirements for each core to enter a power mode. |

# 32.4.1.4 Power Mode Protection register (PMPROT)

### 32.4.1.4.1 Offset

| Register | Offset |
|---|---|
| PMPROT | 8h |

### 32.4.1.4.2 Function

This register provides protection for entry into any run or stop power mode. The enabling of the run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

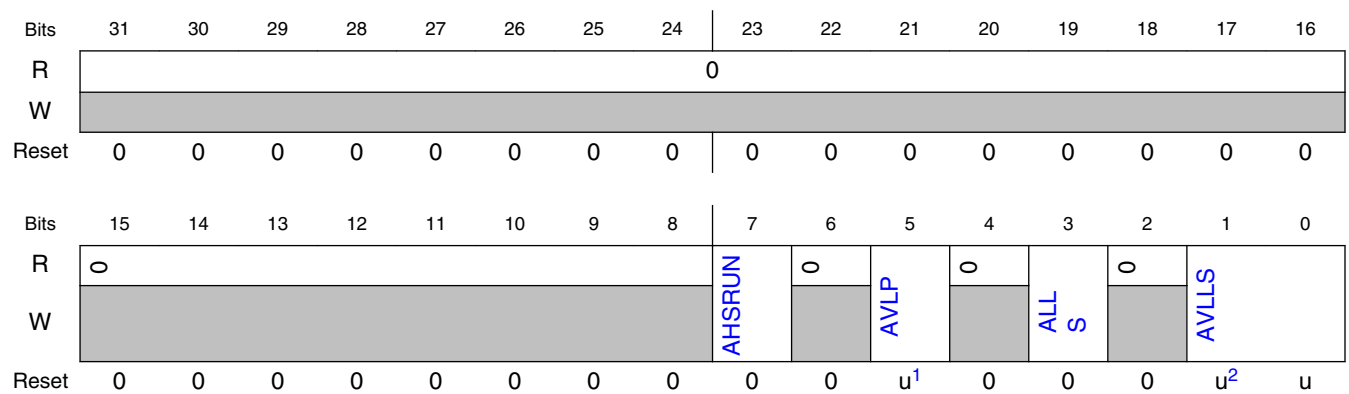
The PMPROT register can be written only once after any system reset.

If a core is configured for a disallowed or reserved power mode, the core remains in its current power mode. For example, if the core is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and PMCTRL[RUNM] remains 00b, indicating the core is still in Normal Run mode.

**NOTE**

This register is reset on Chip Reset and by reset sources that trigger Chip Reset (see the Reset section for more information).

### 32.4.1.4.3 Diagram




1. The reset value is 1 if booting in VLPR mode is supported and configured, otherwise it is 0.
2. The reset value is 10 on VLLS wakeup, otherwise it is 00.

### 32.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-8 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 7<br>AHSRUN | Allow High Speed Run mode<br>When set, allows the PMCTRL[RUNM] field to be written to HSRUN.<br>0b - HSRUN is not allowed<br>1b - HSRUN is allowed |
| 6<br>— | Reserved |
| 5<br>AVLP | Allow Very-Low-Power Modes<br>When set, allows the PMCTRL[RUNM] field to be written to VLPR and/or the PMCTRL[STOPM] field to be written with VLPS.<br>0b - VLPR, VLPW, and VLPS are not allowed.<br>1b - VLPR, VLPW, and VLPS are allowed. |
| 4<br>— | Reserved |
| 3<br>ALLS | Allow Low-Leakage Stop Mode<br>When set, allows the PMCTRL[STOPM] field to be written to LLS.<br>0b - LLS is not allowed<br>1b - LLS is allowed |
| 2<br>— | Reserved |
| 1-0<br>AVLLS | Allow Very-Low-Leakage Stop Mode<br>When set, allows the PMCTRL[STOPM] field to be written to VLLS.<br>00b - VLLS mode is not allowed<br>01b - VLLS0/1 mode is allowed<br>10b - VLLS2/3 mode is allowed<br>11b - VLLS0/1/2/3 mode is allowed |

## 32.4.1.5 Power Mode Control register (PMCTRL)

### 32.4.1.5.1 Offset

| Register | Offset |
|---|---|
| PMCTRL | 10h |

### 32.4.1.5.2 Function

The PMCTRL register controls entry into low-power Run and Stop modes, provided that the selected power mode is allowed via an appropriate setting of the protection (PMPROT) register.

**NOTE**

This register is reset on Chip POR and by reset types that trigger Chip POR. See the Reset section details for more information.

### 32.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | 0 | | | | | | PSTOPO | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | 0 | RUNM | | 0 | | | | | STOPM | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | 0 | 0 | 0 | 0 | 0 | u[2] | u | u |

1. The reset value is 10 if booting in VLPR mode is supported and configured on this device, otherwise it is 00.
2. The reset value is 100 on VLLS wakeup , otherwise it is 000.

### 32.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>PSTOPO | Partial Stop Option<br>These bits control whether a Partial Stop mode is entered when STOPM=STOP. When entering a Partial Stop mode from RUN mode, the PMC and SCG remain fully powered, allowing the device to wakeup almost instantaneously at the expense of higher power consumption. In PSTOP2, only system clocks are gated allowing peripherals running on bus clock to remain fully functional. In PSTOP1, both system and bus clocks are gated.<br>00b - STOP - Normal Stop mode<br>01b - PSTOP1 - Partial Stop with system and bus clock disabled<br>10b - PSTOP2 - Partial Stop with system clock disabled and bus clock enabled<br>11b - PSTOP3 - Partial Stop with system clock enabled and bus clock enabled |
| 15-11<br>— | Reserved |
| 10<br>— | Reserved |
| 9-8 | Run Mode Control |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RUNM | When written, causes entry into the selected run mode. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. Writes to this field that attempt to enter VLPR or HSRUN are also blocked unless PMSTAT=RUN.<br>**NOTE:** Stop mode entry should not be attempted while RUNM=HSRUN or PMSTAT=HSRUN.<br>00b - Normal Run mode (RUN)<br>01b - Reserved<br>10b - Very-Low-Power Run mode (VLPR)<br>11b - High Speed Run mode (HSRUN) |
| 7-3<br>— | Reserved |
| 2-0<br>STOPM | Stop Mode Control<br>Selects the desired stop mode when a core enters deepsleep. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. After any system reset, this field is cleared by hardware on any successful write to the PMPROT register.<br>**NOTE:** When set to STOP, the PSTOPO bits can be used to select a Partial Stop mode if desired.<br>000b - Normal Stop (STOP)<br>001b - Reserved<br>010b - Very-Low-Power Stop (VLPS)<br>011b - Low-Leakage Stop (LLS)<br>100b - Very-Low-Leakage Stop with SRAM retention(VLLS2/3)<br>101b - Reserved<br>110b - Very-Low-Leakage Stop without SRAM retention (VLLS0/1)<br>111b - Reserved |

## 32.4.1.6 Power Mode Status register (PMSTAT)

### 32.4.1.6.1 Offset

| Register | Offset |
|---|---|
| PMSTAT | 18h |

### 32.4.1.6.2 Function

PMSTAT is a read-only, one-hot register which indicates the current power mode of the system.

### 32.4.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | STOPSTAT | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PMSTAT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u | u | u | u | u |

1. The reset value is 0x04 if booting in VLPR mode is supported and configured on this device, otherwise it is 0x01.

### 32.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>STOPSTAT | Stop Entry Status<br>Returns the result of the previous stop mode entry, write 0x01 to clear this field (all other writes are ignored).<br>[0] - When set, indicates a Stop mode entry since this field was last cleared.<br>[1] - When set, indicates the system bus masters acknowledged the Stop mode entry.<br>[2] - When set, indicates the system clock peripherals acknowledged the Stop mode entry.<br>[3] - When set, indicates the bus clock peripherals acknowledged the Stop mode entry.<br>[4] - When set, indicates the slow clock peripherals acknowledged the Stop mode entry.<br>[5] - When set, indicates Stop mode entry completed.<br>[6] - Reserved.<br>[7] - Reserved. |
| 23-8<br>— | Reserved |
| 7-0<br>PMSTAT | Power Mode Status<br>**NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS<br>**NOTE:** When a Partial Stop Option is enabled, the PMSTAT will not update to STOP or VLPS<br>**NOTE:** If booting into VLPR mode is supported and configured on this device, the PMSTAT will update to VLPR shortly after the reset sequence is complete.<br>00000001b - Current power mode is RUN.<br>00000010b - Current power mode is any STOP mode.<br>00000100b - Current power mode is VLPR.<br>10000000b - Current power mode is HSRUN |

## 32.4.1.7 System Reset Status (SRS)

### 32.4.1.7.1 Offset

| Register | Offset |
|---|---|
| SRS | 20h |

### 32.4.1.7.2 Function

This register includes read-only status flags to indicate the source of the most recent reset.

**NOTE**

The reset state of this register depends on the reset source:

- POR = 0x0000_0006
- LVD = 0x0000_0004
- VLLS Wakeup = 0x0000_0001
- VLLS Reset Wakeup = 0x0000_0111

### 32.4.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | JTAG | 0 | | | 0 | 0 | | | | | | CORE1 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKUP | SW | WDOG | SCG | STOPACK | RSTACK | MDM | PIN | CORE | 0 | FATAL | WARM | HVD | LVD | POR | WAKEUP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

### 32.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>— | Reserved |
| 28<br>JTAG | JTAG System Reset<br>Indicates a reset has been caused by a JTAG system reset request. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated by JTAG system reset.<br>1b - Reset generated by JTAG system reset. |
| 27-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>CORE1 | Core1 System Reset<br>Indicates the reset source was generated by a Core1 specific reset.<br>0b - Reset not generated from Core1 system reset source.<br>1b - Reset generated from Core1 system reset source. |
| 16<br>— | Reserved |
| 15<br>LOCKUP | Lockup Reset<br>Indicates a reset has been caused by the ARM core indication of a LOCKUP event.<br>0b - Reset not generated by core lockup or exception.<br>1b - Reset generated by core lockup or exception. |
| 14<br>SW | Software Reset<br>Indicates a reset has been caused by a software reset request from the ARM core (SYSRESETREQ).<br>0b - Reset not generated by software request from core.<br>1b - Reset generated by software request from core. |
| 13<br>WDOG | Watchdog Reset<br>Indicates a reset has been caused by a WatchDog timeout.<br>0b - Reset is not generated from the WatchDog timeout.<br>1b - Reset is generated from the WatchDog timeout. |
| 12<br>SCG | SCG Reset<br>Indicates a reset has been caused by a loss-of-clock or loss-of-lock event in the SCG. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset is not generated from an SCG loss of lock or loss of clock.<br>1b - Reset is generated from an SCG loss of lock or loss of clock. |
| 11<br>STOPACK | Stop Timeout Reset<br>Indicates a reset has been caused by a timeout in the Stop mode entry logic. This timeout is generated if a peripheral does not acknowledge entry into any Stop mode within approximately one second.<br>0b - Reset not generated by Stop Controller Timeout.<br>1b - Reset generated by Stop Controller Timeout. |
| 10<br>RSTACK | Reset Timeout<br>Indicates a reset has been caused by a timeout in the system reset generaton logic. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated from Reset Controller Timeout.<br>1b - Reset generated from Reset Controller Timeout. |

*Table continues on the next page...*

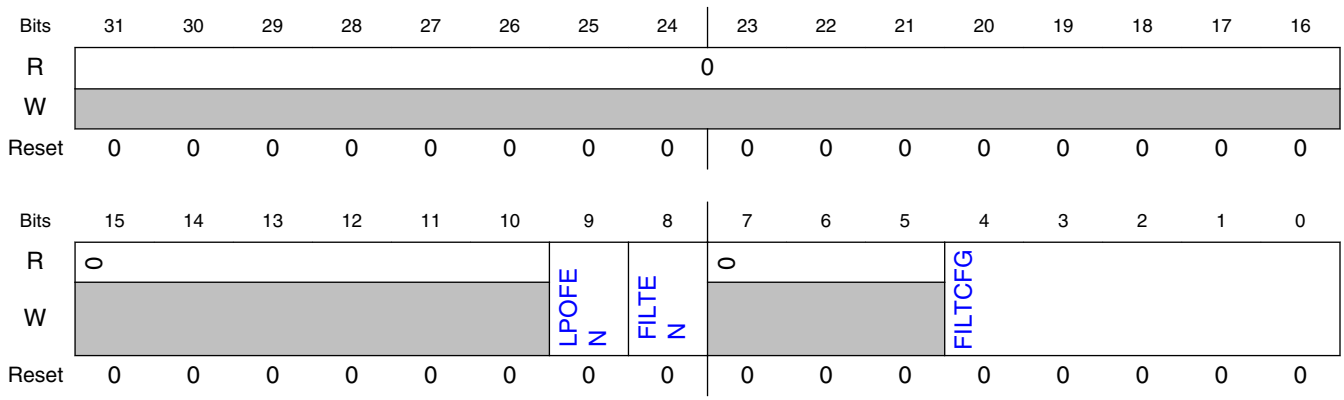
| Field | Function |
|---|---|
| 9<br>MDM | MDM Reset<br>Indicates a reset has been caused by a reset request from the Miscellaneous Debug Module (MDM).<br>0b - Reset was not generated from the MDM reset request.<br>1b - Reset was generated from the MDM reset request. |
| 8<br>PIN | Pin Reset<br>Indicates a reset has been caused by external assertion of the RESET_b pin.<br>0b - Reset was not generated from the assertion of RESET_B pin.<br>1b - Reset was generated from the assertion of RESET_B pin. |
| 7<br>CORE | Core Reset<br>Indicates the last reset was a reset that only reset the core.<br>0b - Reset source was not core only reset.<br>1b - Reset source was core reset and reset the core only. |
| 6<br>— | Reserved |
| 5<br>FATAL | Fatal Reset<br>Fatal Reset will assert if the system reset source was fatal. SRAM contents cannot be guaranteed following a fatal reset source.<br>0b - Reset was not generated by a fatal reset source.<br>1b - Reset was generated by a fatal reset source. |
| 4<br>WARM | Warm Reset<br>Warm Reset flag will assert whenever any of the system reset sources in this register assert (SRS[31:8]).<br>0b - Reset not generated by Warm Reset source.<br>1b - Reset generated by Warm Reset source. |
| 3<br>HVD | HVD Reset<br>Indicates a reset has been caused by a High Voltage Detect. SRAM contents cannot be guaranteed following a HVD reset source.<br>0b - Reset not generated by HVD.<br>1b - Reset generated by HVD. |
| 2<br>LVD | LVD Reset<br>Indicates a reset has been caused by a Low Voltage Detect. SRAM contents cannot be guaranteed following a LVD reset source.<br>0b - Reset not generated by LVD.<br>1b - Reset generated by LVD. |
| 1<br>POR | POR Reset<br>Indicates a reset has been caused by Power On Reset detection logic. SRAM contents cannot be guaranteed following a POR reset source.<br>0b - Reset not generated by POR.<br>1b - Reset generated by POR. |
| 0<br>WAKEUP | Wakeup Reset<br>Indicates a reset has been caused by a wakeup from VLLS mode.<br>0b - Reset not generated by wakeup from VLLS mode.<br>1b - Reset generated by wakeup from VLLS mode. |

# 32.4.1.8 Reset Pin Control (RPC)

## 32.4.1.8.1 Offset

| Register | Offset |
|---|---|
| RPC | 24h |

## 32.4.1.8.2 Diagram




## 32.4.1.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>LPOFEN | LPO Filter Enable<br>0b - LPO clock reset pin filter disabled.<br>1b - LPO clock reset pin filter enabled in all modes. |
| 8<br>FILTEN | Filter Enable<br>0b - Slow clock reset pin filter disabled.<br>1b - Slow clock reset pin filter enabled in Run modes. |
| 7-5<br>— | Reserved |
| 4-0<br>FILTCFG | Reset Filter Configuration<br>Configures the width of the reset pin filter from 1 to 32 slow clock cycles. |

# 32.4.1.9 Sticky System Reset Status (SSRS)

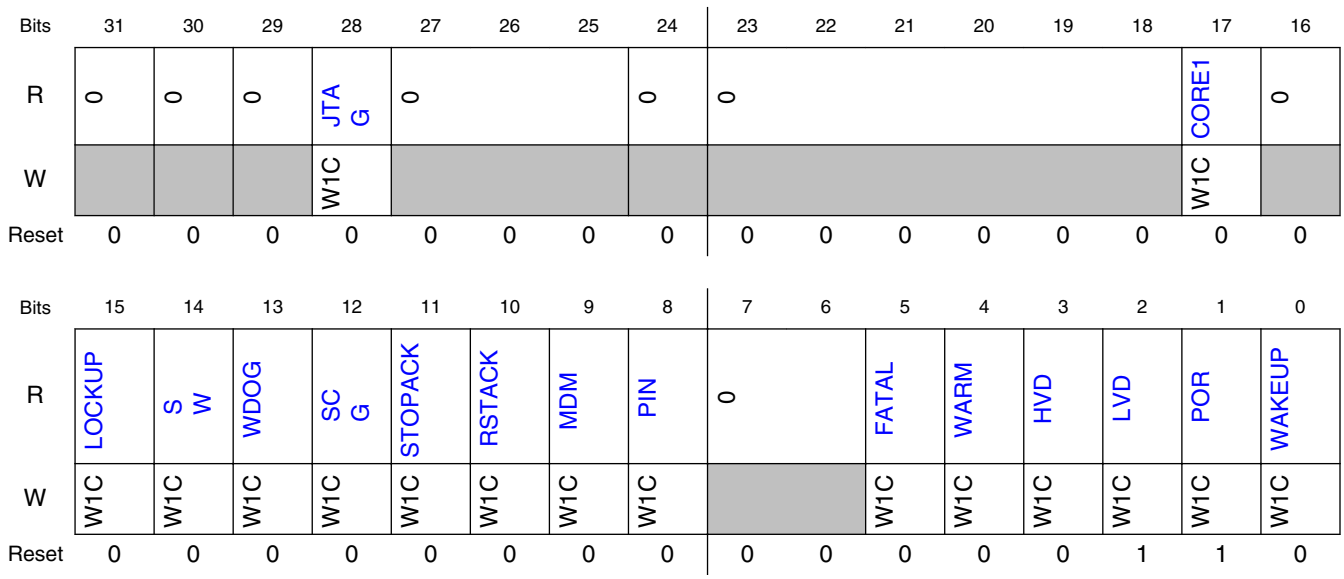
### 32.4.1.9.1 Offset

| Register | Offset |
|---|---|
| SSRS | 28h |

### 32.4.1.9.2 Function

The SSRS are only reset on POR/LVD/HVD/WAKEUP and remain set until cleared by software. The SSRS stores all system reset sources that have generated a system reset, since the last POR/LVD/HVD/WAKEUP, that have not been cleared by software. The SSRS does not update following a core software reset.

### 32.4.1.9.3 Diagram




### 32.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28 | JTAG System Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| JTAG | This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated by JTAG system reset.<br>1b - Reset generated by JTAG system reset. |
| 27-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>CORE1 | Core1 Reset<br>0b - Reset not generated from Core1 reset source.<br>1b - Reset generated from Core1 reset source. |
| 16<br>— | Reserved |
| 15<br>LOCKUP | Lockup Reset<br>0b - Reset not generated by core lockup.<br>1b - Reset generated by core lockup. |
| 14<br>SW | Software Reset<br>0b - Reset not generated by software request from core.<br>1b - Reset generated by software request from core. |
| 13<br>WDOG | Watchdog Reset<br>0b - Reset is not generated from the WatchDog timeout.<br>1b - Reset is generated from the WatchDog timeout. |
| 12<br>SCG | SCG Reset<br>This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset is not generated from an SCG loss of lock or loss of clock.<br>1b - Reset is generated from an SCG loss of lock or loss of clock. |
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Reset not generated by Stop Controller Timeout.<br>1b - Reset generated by Stop Controller Timeout. |
| 10<br>RSTACK | Reset Timeout<br>This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated from Reset Controller Timeout.<br>1b - Reset generated from Reset Controller Timeout. |
| 9<br>MDM | MDM Reset<br>0b - Reset was not generated from the MDM reset request.<br>1b - Reset was generated from the MDM reset request. |
| 8<br>PIN | Pin Reset<br>0b - Reset was not generated from the RESET_B pin.<br>1b - Reset was generated from the RESET_B pin. |
| 7-6<br>— | Reserved |
| 5<br>FATAL | Fatal Reset<br>0b - Reset was not generated by a fatal reset source.<br>1b - Reset was generated by a fatal reset source. |
| 4<br>WARM | Warm Reset<br>0b - Reset not generated by system reset source. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Reset generated by system reset source. |
| 3<br>HVD | HVD Reset<br>0b - Reset not generated by HVD.<br>1b - Reset generated by HVD. |
| 2<br>LVD | LVD Reset<br>0b - Reset not generated by LVD.<br>1b - Reset generated by LVD. |
| 1<br>POR | POR Reset<br>0b - Reset not generated by POR.<br>1b - Reset generated by POR. |
| 0<br>WAKEUP | Wakeup Reset<br>0b - Reset not generated by wakeup from VLLS mode.<br>1b - Reset generated by wakeup from VLLS mode. |

## 32.4.1.10 System Reset Interrupt Enable (SRIE)

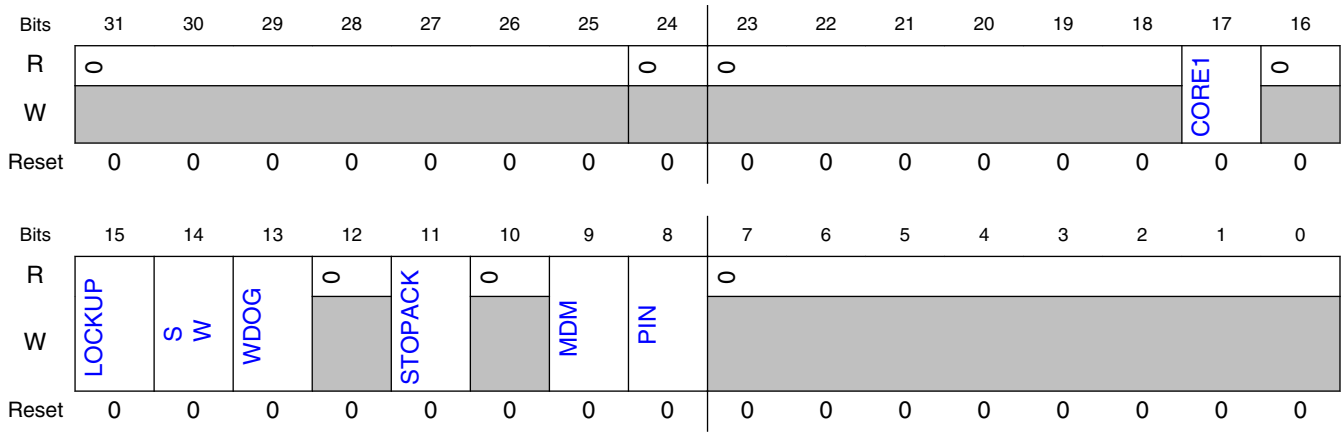
### 32.4.1.10.1 Offset

| Register | Offset |
|---|---|
| SRIE | 2Ch |

### 32.4.1.10.2 Function

This registers delays the assertion of a system reset for between 9 and 10 LPO clock cycles while an interrupt is generated. This allows software to perform a graceful shutdown or to abort the system reset provided the pending reset source is cleared by resetting the source of the reset and then clearing the pending flag. A Chip POR or fatal reset source cannot be delayed by this feature. The SRS will only update after the system reset occurs.

## 32.4.1.10.3 Diagram




## 32.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>CORE1 | Core1 Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 16<br>— | Reserved |
| 15<br>LOCKUP | Lockup Reset<br>Note that interrupts cannot be serviced by a core in the lockup state.<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 14<br>SW | Software Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 13<br>WDOG | Watchdog Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 12<br>— | Reserved |
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 10 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 9<br>MDM | MDM Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 8<br>PIN | Pin Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 7-0<br>— | Reserved |

## 32.4.1.11 System Reset Interrupt Flag (SRIF)

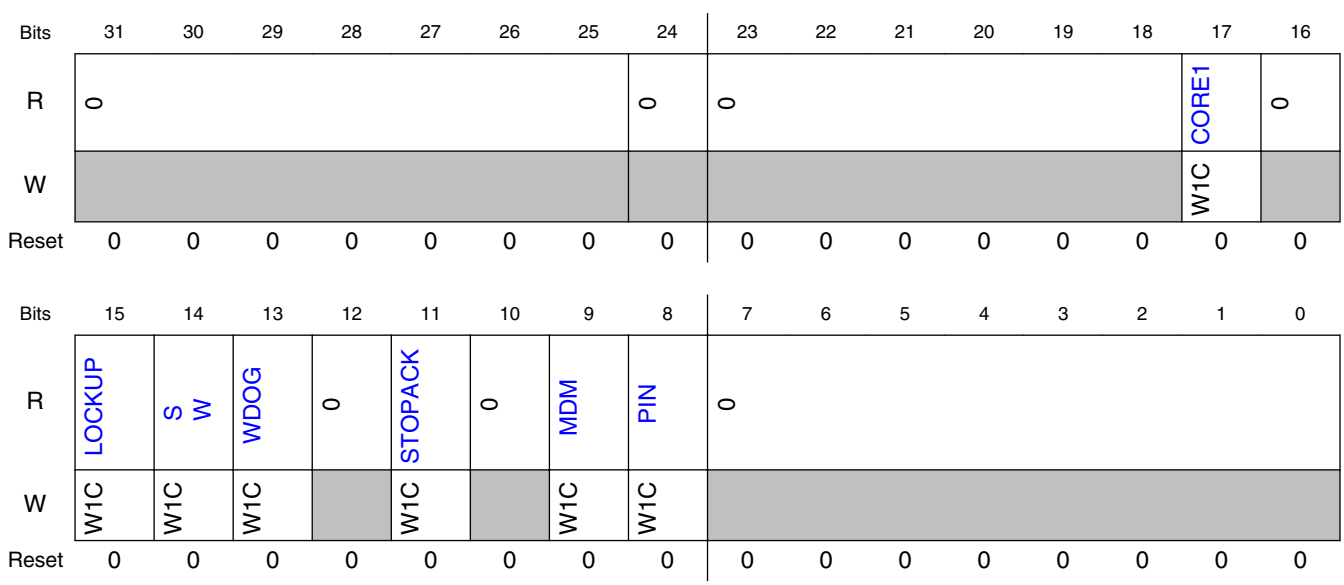
### 32.4.1.11.1 Offset

| Register | Offset |
|---|---|
| SRIF | 30h |

### 32.4.1.11.2 Function

This registers returns the source of the reset interrupt. The pending reset source can be cleared by resetting the source of the reset and then clearing the pending flag.

### 32.4.1.11.3 Diagram



### 32.4.1.11.4 Fields

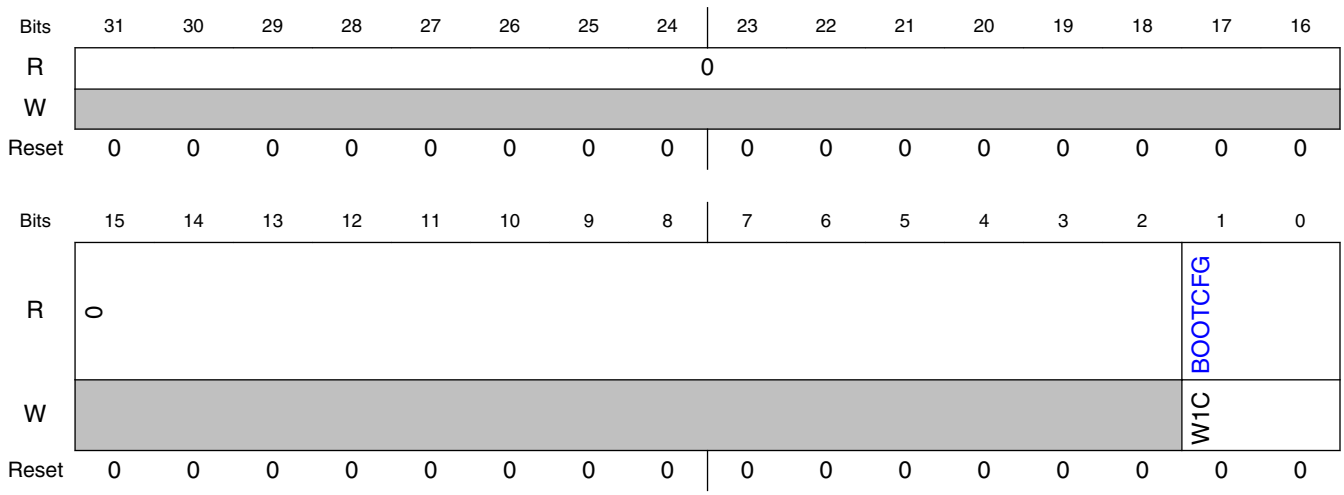
| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>CORE1 | Core1 Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 16<br>— | Reserved |
| 15<br>LOCKUP | Lockup Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 14<br>SW | Software Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 13<br>WDOG | Watchdog Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 12<br>— | Reserved |
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 10<br>— | Reserved |
| 9<br>MDM | MDM Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 8<br>PIN | Pin Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 7-0<br>— | Reserved |

## 32.4.1.12 Mode Register (MR)

### 32.4.1.12.1 Offset

| Register | Offset |
|---|---|
| MR | 40h |

### 32.4.1.12.2 Diagram




### 32.4.1.12.3 Fields

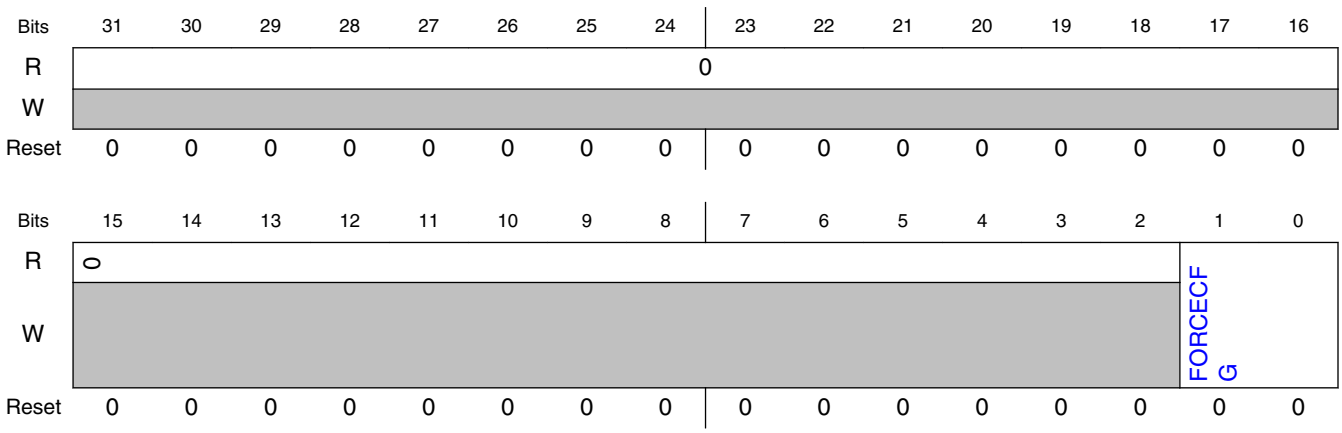
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>BOOTCFG | Boot Configuration<br>Indicates the boot source, the boot source remains set until the next System Reset or software can write logic one to clear the corresponding mode bit. The boot source is forced to Flash on a VLLS wakeup that does not include another system reset source.<br>While either bit is set, the vector take is relocated to the ROM base address. These bits should be cleared by writing logic one before executing any code from either Flash or SRAM.<br>00b - Boot from Flash.<br>01b - Boot from ROM due to BOOTCFG0 pin assertion.<br>10b - Boot from ROM due to FOPT configuration.<br>11b - Boot from ROM due to both BOOTCFG0 pin assertion and FOPT configuration. |

## 32.4.1.13 Force Mode Register (FM)

### 32.4.1.13.1 Offset

| Register | Offset |
|---|---|
| FM | 50h |

### 32.4.1.13.2 Diagram




### 32.4.1.13.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>FORCECFG | Boot Configuration<br>This register can force the corresponding bit in the Mode Register to assert on next system reset.<br>00b - No effect.<br>01b - Assert corresponding bit in Mode Register on next system reset. |

## 32.4.1.14 SRAM Low Power Register (SRAMLPR)

### 32.4.1.14.1 Offset

| Register | Offset |
|---|---|
| SRAMLPR | 60h |

#### 32.4.1.14.2 Function

This register places specific System SRAM arrays into a low power retention state whenever the corresponding register bit is set. The SRAM arrays should not be accessed while they are in a low power retention state.

#### 32.4.1.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 32.4.1.14.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LPE | Low Power Enable<br>0b - Corresponding SRAM array in Run mode.<br>1b - Corresponding SRAM array in low power retention state and should not be accessed. |

### 32.4.1.15 SRAM Deep Sleep Register (SRAMDSR)

#### 32.4.1.15.1 Offset

| Register | Offset |
|---|---|
| SRAMDSR | 64h |

#### 32.4.1.15.2 Function

This register configures if the System SRAM arrays are retained or powered off in Stop modes.

### 32.4.1.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DSE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DSE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 32.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DSE | Deep Sleep Enable<br>0b - Corresponding SRAM array is retained in Stop modes.<br>1b - Corresponding SRAM array is powered off in Stop modes. |

## 32.4.2 SMC1 register descriptions

Different MSMC registers reset on different reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

**NOTE**

The MSMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**NOTE**

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode. Failure to do this may result in the low power mode not being entered correctly.

### 32.4.2.1 SMC1 Memory map

SMC1 base address: 4102_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0200_00ABh |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0000h |
| 8h | Power Mode Protection register (PMPROT) | 32 | RW | See description. |
| 10h | Power Mode Control register (PMCTRL) | 32 | RW | See description. |
| 18h | Power Mode Status register (PMSTAT) | 32 | RW | See description. |
| 20h | System Reset Status (SRS) | 32 | RO | 0000_0006h |
| 28h | Sticky System Reset Status (SSRS) | 32 | W1C | 0000_0006h |
| 2Ch | System Reset Interrupt Enable (SRIE) | 32 | RW | 0000_0000h |
| 30h | System Reset Interrupt Flag (SRIF) | 32 | W1C | 0000_0000h |
| 40h | Mode Register (MR) | 32 | W1C | 0000_0000h |
| 50h | Force Mode Register (FM) | 32 | RW | 0000_0000h |
| 60h | SRAM Low Power Register (SRAMLPR) | 32 | RW | 0000_0000h |
| 64h | SRAM Deep Sleep Register (SRAMDSR) | 32 | RW | 0000_0000h |

### 32.4.2.2 Version ID Register (VERID)

#### 32.4.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 32.4.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

#### 32.4.2.2.3 Fields

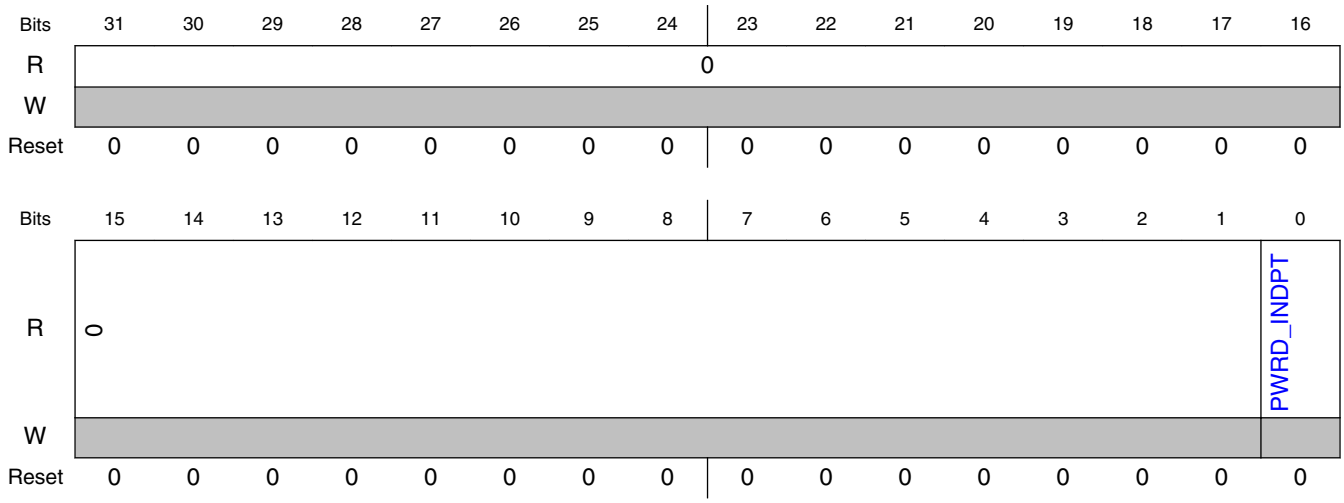
| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000010101011b - Default features supported |

### 32.4.2.3 Parameter Register (PARAM)

#### 32.4.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 32.4.2.3.2 Diagram




#### 32.4.2.3.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>PWRD_INDPT | Power Domains Independent<br>When set, means each core is able to enter power modes independently of the other. When clear, means certain dependancies are enforced by the MSMC before each core is able to enter some power modes. See "Power mode transitions" section for detailed requirements for each core to enter a power mode. |

### 32.4.2.4 Power Mode Protection register (PMPROT)

#### 32.4.2.4.1 Offset

| Register | Offset |
|---|---|
| PMPROT | 8h |

#### 32.4.2.4.2 Function

This register provides protection for entry into any run or stop power mode. The enabling of the run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

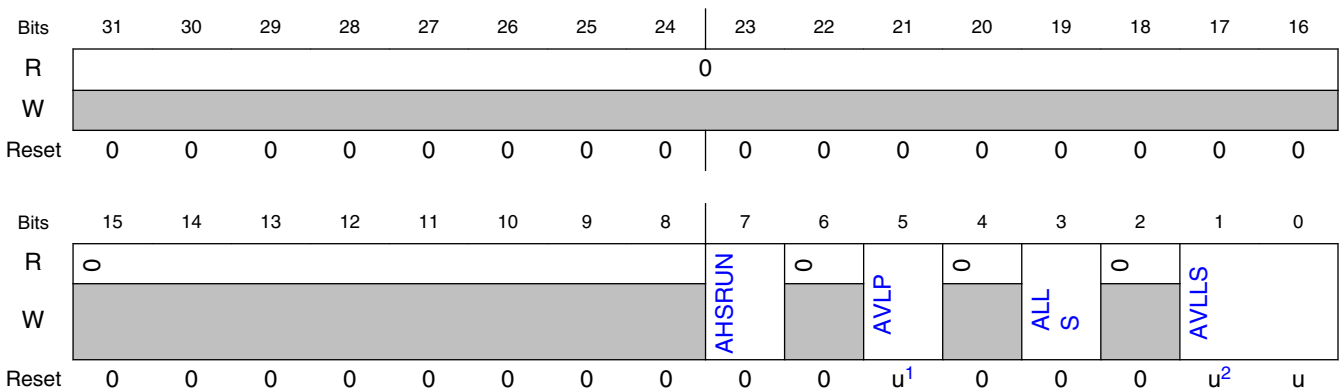
The PMPROT register can be written only once after any system reset.

If a core is configured for a disallowed or reserved power mode, the core remains in its current power mode. For example, if the core is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and PMCTRL[RUNM] remains 00b, indicating the core is still in Normal Run mode.

**NOTE**

This register is reset on Chip Reset and by reset sources that trigger Chip Reset (see the Reset section for more information).

### 32.4.2.4.3 Diagram




1. The reset value is 1 if booting in VLPR mode is supported and configured, otherwise it is 0.
2. The reset value is 10 on VLLS wakeup, otherwise it is 00.

### 32.4.2.4.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>AHSRUN | Allow High Speed Run mode<br>When set, allows the PMCTRL[RUNM] field to be written to HSRUN.<br>0b - HSRUN is not allowed<br>1b - HSRUN is allowed |
| 6<br>— | Reserved |
| 5<br>AVLP | Allow Very-Low-Power Modes<br>When set, allows the PMCTRL[RUNM] field to be written to VLPR and/or the PMCTRL[STOPM] field to be written with VLPS.<br>0b - VLPR, VLPW, and VLPS are not allowed.<br>1b - VLPR, VLPW, and VLPS are allowed. |
| 4 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 3<br>ALLS | Allow Low-Leakage Stop Mode<br>When set, allows the PMCTRL[STOPM] field to be written to LLS.<br>0b - LLS is not allowed<br>1b - LLS is allowed |
| 2<br>— | Reserved |
| 1-0<br>AVLLS | Allow Very-Low-Leakage Stop Mode<br>When set, allows the PMCTRL[STOPM] field to be written to VLLS.<br>00b - VLLS mode is not allowed<br>01b - VLLS0/1 mode is allowed<br>10b - VLLS2/3 mode is allowed<br>11b - VLLS0/1/2/3 mode is allowed |

## 32.4.2.5 Power Mode Control register (PMCTRL)

### 32.4.2.5.1 Offset

| Register | Offset |
|---|---|
| PMCTRL | 10h |

### 32.4.2.5.2 Function

The PMCTRL register controls entry into low-power Run and Stop modes, provided that the selected power mode is allowed via an appropriate setting of the protection (PMPROT) register.

**NOTE**

This register is reset on Chip POR and by reset types that trigger Chip POR. See the Reset section details for more information.

## 32.4.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | 0 | | | | | | PSTOPO | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | 0 | RUNM | | 0 | | | | | STOPM | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | 0 | 0 | 0 | 0 | 0 | u[2] | u | u |

1. The reset value is 10 if booting in VLPR mode is supported and configured on this device, otherwise it is 00.
2. The reset value is 100 on VLLS wakeup , otherwise it is 000.

## 32.4.2.5.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>PSTOPO | Partial Stop Option<br>These bits control whether a Partial Stop mode is entered when STOPM=STOP. When entering a Partial Stop mode from RUN mode, the PMC and SCG remain fully powered, allowing the device to wakeup almost instantaneously at the expense of higher power consumption. In PSTOP2, only system clocks are gated allowing peripherals running on bus clock to remain fully functional. In PSTOP1, both system and bus clocks are gated.<br>00b - STOP - Normal Stop mode<br>01b - PSTOP1 - Partial Stop with system and bus clock disabled<br>10b - PSTOP2 - Partial Stop with system clock disabled and bus clock enabled<br>11b - PSTOP3 - Partial Stop with system clock enabled and bus clock enabled |
| 15-11<br>— | Reserved |
| 10<br>— | Reserved |
| 9-8<br>RUNM | Run Mode Control<br>When written, causes entry into the selected run mode. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. Writes to this field that attempt to enter VLPR or HSRUN are also blocked unless PMSTAT=RUN.<br>**NOTE:** Stop mode entry should not be attempted while RUNM=HSRUN or PMSTAT=HSRUN.<br>00b - Normal Run mode (RUN)<br>01b - Reserved<br>10b - Very-Low-Power Run mode (VLPR) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 11b - High Speed Run mode (HSRUN) |
| 7-3<br>— | Reserved |
| 2-0<br>STOPM | Stop Mode Control<br>Selects the desired stop mode when a core enters deepsleep. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. After any system reset, this field is cleared by hardware on any successful write to the PMPROT register.<br>**NOTE:** When set to STOP, the PSTOPO bits can be used to select a Partial Stop mode if desired.<br>000b - Normal Stop (STOP)<br>001b - Reserved<br>010b - Very-Low-Power Stop (VLPS)<br>011b - Low-Leakage Stop (LLS)<br>100b - Very-Low-Leakage Stop with SRAM retention(VLLS2/3)<br>101b - Reserved<br>110b - Very-Low-Leakage Stop without SRAM retention (VLLS0/1)<br>111b - Reserved |

## 32.4.2.6 Power Mode Status register (PMSTAT)

### 32.4.2.6.1 Offset

| Register | Offset |
|---|---|
| PMSTAT | 18h |

### 32.4.2.6.2 Function

PMSTAT is a read-only, one-hot register which indicates the current power mode of the system.

### 32.4.2.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | STOPSTAT | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PMSTAT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u | u | u | u | u |

1. The reset value is 0x04 if booting in VLPR mode is supported and configured on this device, otherwise it is 0x01.

### 32.4.2.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>STOPSTAT | Stop Entry Status<br>Returns the result of the previous stop mode entry, write 0x01 to clear this field (all other writes are ignored).<br>[0] - When set, indicates a Stop mode entry since this field was last cleared.<br>[1] - When set, indicates the system bus masters acknowledged the Stop mode entry.<br>[2] - When set, indicates the system clock peripherals acknowledged the Stop mode entry.<br>[3] - When set, indicates the bus clock peripherals acknowledged the Stop mode entry.<br>[4] - When set, indicates the slow clock peripherals acknowledged the Stop mode entry.<br>[5] - When set, indicates Stop mode entry completed.<br>[6] - Reserved.<br>[7] - Reserved. |
| 23-8<br>— | Reserved |
| 7-0<br>PMSTAT | Power Mode Status<br>**NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS<br>**NOTE:** When a Partial Stop Option is enabled, the PMSTAT will not update to STOP or VLPS<br>**NOTE:** If booting into VLPR mode is supported and configured on this device, the PMSTAT will update to VLPR shortly after the reset sequence is complete.<br>00000001b - Current power mode is RUN.<br>00000010b - Current power mode is any STOP mode.<br>00000100b - Current power mode is VLPR.<br>10000000b - Current power mode is HSRUN |

## 32.4.2.7 System Reset Status (SRS)

### 32.4.2.7.1 Offset

| Register | Offset |
|---|---|
| SRS | 20h |

### 32.4.2.7.2 Function

This register includes read-only status flags to indicate the source of the most recent reset.

**NOTE**

The reset state of this register depends on the reset source:

- POR = 0x0000_0006

- LVD = 0x0000_0004
- VLLS Wakeup = 0x0000_0001
- VLLS Reset Wakeup = 0x0000_0111

### 32.4.2.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | JTAG | 0 | | | 0 | 0 | | | | | | 0 | CORE0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKUP | SW | WDOG | SCG | STOPACK | RSTACK | MDM | PIN | CORE | 0 | FATAL | WARM | HVD | LVD | POR | WAKEUP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

### 32.4.2.7.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>JTAG | JTAG System Reset<br>Indicates a reset has been caused by a JTAG system reset request. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated by JTAG system reset.<br>1b - Reset generated by JTAG system reset. |
| 27-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 16<br>CORE0 | Core0 System Reset<br>Indicates the reset source was generated by a Core0 specific reset.<br>0b - Reset not generated from Core0 system reset source.<br>1b - Reset generated from Core0 system reset source. |
| 15<br>LOCKUP | Lockup Reset<br>Indicates a reset has been caused by the ARM core indication of a LOCKUP event.<br>0b - Reset not generated by core lockup or exception.<br>1b - Reset generated by core lockup or exception. |
| 14<br>SW | Software Reset<br>Indicates a reset has been caused by a software reset request from the ARM core (SYSRESETREQ).<br>0b - Reset not generated by software request from core.<br>1b - Reset generated by software request from core. |
| 13<br>WDOG | Watchdog Reset<br>Indicates a reset has been caused by a WatchDog timeout.<br>0b - Reset is not generated from the WatchDog timeout.<br>1b - Reset is generated from the WatchDog timeout. |
| 12<br>SCG | SCG Reset<br>Indicates a reset has been caused by a loss-of-clock or loss-of-lock event in the SCG. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset is not generated from an SCG loss of lock or loss of clock.<br>1b - Reset is generated from an SCG loss of lock or loss of clock. |
| 11<br>STOPACK | Stop Timeout Reset<br>Indicates a reset has been caused by a timeout in the Stop mode entry logic. This timeout is generated if a peripheral does not acknowledge entry into any Stop mode within approximately one second.<br>0b - Reset not generated by Stop Controller Timeout.<br>1b - Reset generated by Stop Controller Timeout. |
| 10<br>RSTACK | Reset Timeout<br>Indicates a reset has been caused by a timeout in the system reset generaton logic. This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated from Reset Controller Timeout.<br>1b - Reset generated from Reset Controller Timeout. |
| 9<br>MDM | MDM Reset<br>Indicates a reset has been caused by a reset request from the Miscellaneous Debug Module (MDM).<br>0b - Reset was not generated from the MDM reset request.<br>1b - Reset was generated from the MDM reset request. |
| 8<br>PIN | Pin Reset<br>Indicates a reset has been caused by external assertion of the RESET_b pin.<br>0b - Reset was not generated from the assertion of RESET_B pin.<br>1b - Reset was generated from the assertion of RESET_B pin. |
| 7<br>CORE | Core Reset<br>Indicates the last reset was a reset that only reset the core.<br>0b - Reset source was not core only reset.<br>1b - Reset source was core reset and reset the core only. |
| 6<br>— | Reserved |
| 5 | Fatal Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FATAL | Fatal Reset will assert if the system reset source was fatal. SRAM contents cannot be guaranteed following a fatal reset source.<br>0b - Reset was not generated by a fatal reset source.<br>1b - Reset was generated by a fatal reset source. |
| 4<br>WARM | Warm Reset<br>Warm Reset flag will assert whenever any of the system reset sources in this register assert (SRS[31:8]).<br>0b - Reset not generated by Warm Reset source.<br>1b - Reset generated by Warm Reset source. |
| 3<br>HVD | HVD Reset<br>Indicates a reset has been caused by a High Voltage Detect. SRAM contents cannot be guaranteed following a HVD reset source.<br>0b - Reset not generated by HVD.<br>1b - Reset generated by HVD. |
| 2<br>LVD | LVD Reset<br>Indicates a reset has been caused by a Low Voltage Detect. SRAM contents cannot be guaranteed following a LVD reset source.<br>0b - Reset not generated by LVD.<br>1b - Reset generated by LVD. |
| 1<br>POR | POR Reset<br>Indicates a reset has been caused by Power On Reset detection logic. SRAM contents cannot be guaranteed following a POR reset source.<br>0b - Reset not generated by POR.<br>1b - Reset generated by POR. |
| 0<br>WAKEUP | Wakeup Reset<br>Indicates a reset has been caused by a wakeup from VLLS mode.<br>0b - Reset not generated by wakeup from VLLS mode.<br>1b - Reset generated by wakeup from VLLS mode. |

## 32.4.2.8 Sticky System Reset Status (SSRS)

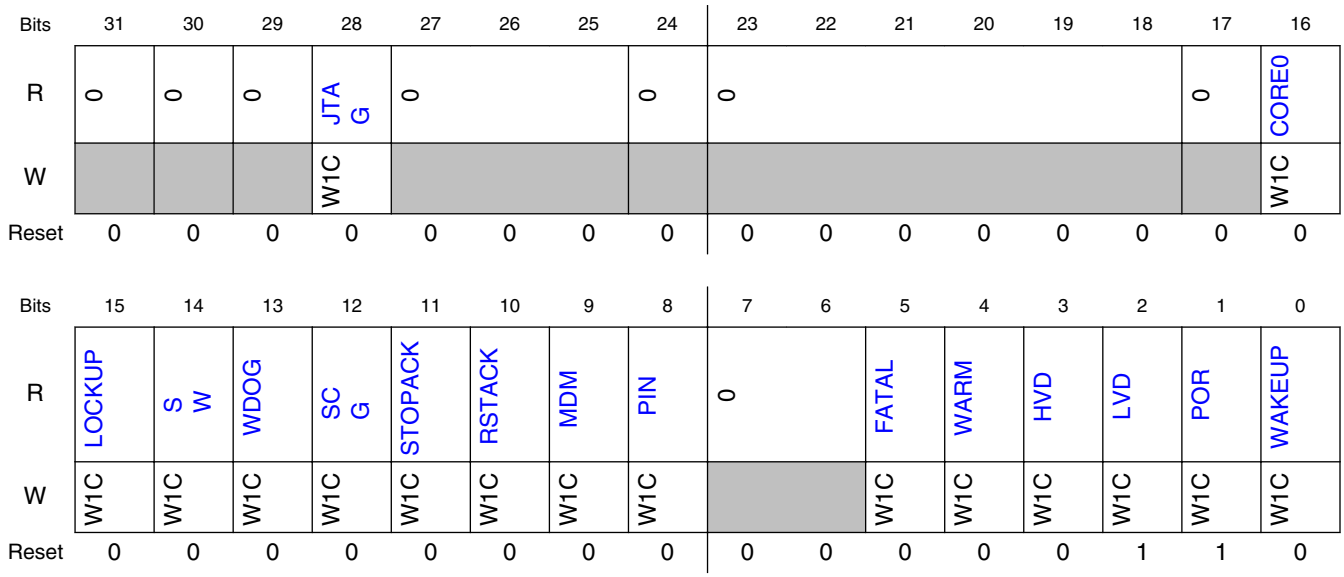
### 32.4.2.8.1 Offset

| Register | Offset |
|---|---|
| SSRS | 28h |

### 32.4.2.8.2 Function

The SSRS are only reset on POR/LVD/HVD/WAKEUP and remain set until cleared by software. The SSRS stores all system reset sources that have generated a system reset, since the last POR/LVD/HVD/WAKEUP, that have not been cleared by software. The SSRS does not update following a core software reset.

## 32.4.2.8.3 Diagram




## 32.4.2.8.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>JTAG | JTAG System Reset<br>This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated by JTAG system reset.<br>1b - Reset generated by JTAG system reset. |
| 27-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>CORE0 | Core0 Reset<br>0b - Reset not generated from Core0 reset source.<br>1b - Reset generated from Core0 reset source. |
| 15 | Lockup Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LOCKUP | 0b - Reset not generated by core lockup.<br>1b - Reset generated by core lockup. |
| 14<br>SW | Software Reset<br>0b - Reset not generated by software request from core.<br>1b - Reset generated by software request from core. |
| 13<br>WDOG | Watchdog Reset<br>0b - Reset is not generated from the WatchDog timeout.<br>1b - Reset is generated from the WatchDog timeout. |
| 12<br>SCG | SCG Reset<br>This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset is not generated from an SCG loss of lock or loss of clock.<br>1b - Reset is generated from an SCG loss of lock or loss of clock. |
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Reset not generated by Stop Controller Timeout.<br>1b - Reset generated by Stop Controller Timeout. |
| 10<br>RSTACK | Reset Timeout<br>This is a fatal reset source and SRAM contents are not guaranteed.<br>0b - Reset not generated from Reset Controller Timeout.<br>1b - Reset generated from Reset Controller Timeout. |
| 9<br>MDM | MDM Reset<br>0b - Reset was not generated from the MDM reset request.<br>1b - Reset was generated from the MDM reset request. |
| 8<br>PIN | Pin Reset<br>0b - Reset was not generated from the RESET_B pin.<br>1b - Reset was generated from the RESET_B pin. |
| 7-6<br>— | Reserved |
| 5<br>FATAL | Fatal Reset<br>0b - Reset was not generated by a fatal reset source.<br>1b - Reset was generated by a fatal reset source. |
| 4<br>WARM | Warm Reset<br>0b - Reset not generated by system reset source.<br>1b - Reset generated by system reset source. |
| 3<br>HVD | HVD Reset<br>0b - Reset not generated by HVD.<br>1b - Reset generated by HVD. |
| 2<br>LVD | LVD Reset<br>0b - Reset not generated by LVD.<br>1b - Reset generated by LVD. |
| 1<br>POR | POR Reset<br>0b - Reset not generated by POR.<br>1b - Reset generated by POR. |
| 0<br>WAKEUP | Wakeup Reset<br>0b - Reset not generated by wakeup from VLLS mode.<br>1b - Reset generated by wakeup from VLLS mode. |

## 32.4.2.9 System Reset Interrupt Enable (SRIE)

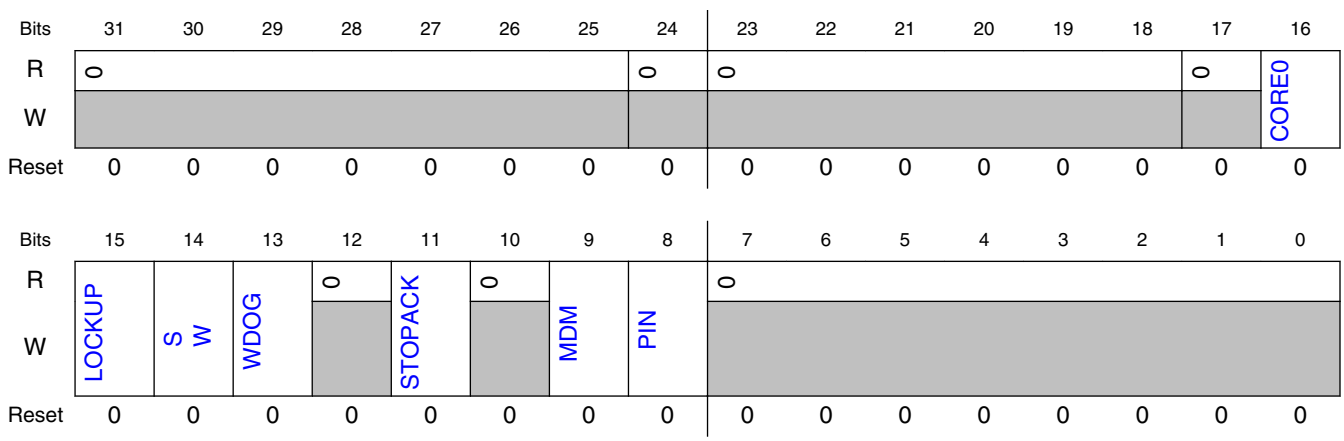
### 32.4.2.9.1 Offset

| Register | Offset |
|---|---|
| SRIE | 2Ch |

### 32.4.2.9.2 Function

This registers delays the assertion of a system reset for between 9 and 10 LPO clock cycles while an interrupt is generated. This allows software to perform a graceful shutdown or to abort the system reset provided the pending reset source is cleared by resetting the source of the reset and then clearing the pending flag. A Chip POR or fatal reset source cannot be delayed by this feature. The SRS will only update after the system reset occurs.

### 32.4.2.9.3 Diagram




### 32.4.2.9.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 17<br>— | Reserved |
| 16<br>CORE0 | Core0 Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 15<br>LOCKUP | Lockup Reset<br>Note that interrupts cannot be serviced by a core in the lockup state.<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 14<br>SW | Software Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 13<br>WDOG | Watchdog Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 12<br>— | Reserved |
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 10<br>— | Reserved |
| 9<br>MDM | MDM Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 8<br>PIN | Pin Reset<br>0b - Interrupt disabled.<br>1b - Interrupt enabled. |
| 7-0<br>— | Reserved |

## 32.4.2.10 System Reset Interrupt Flag (SRIF)

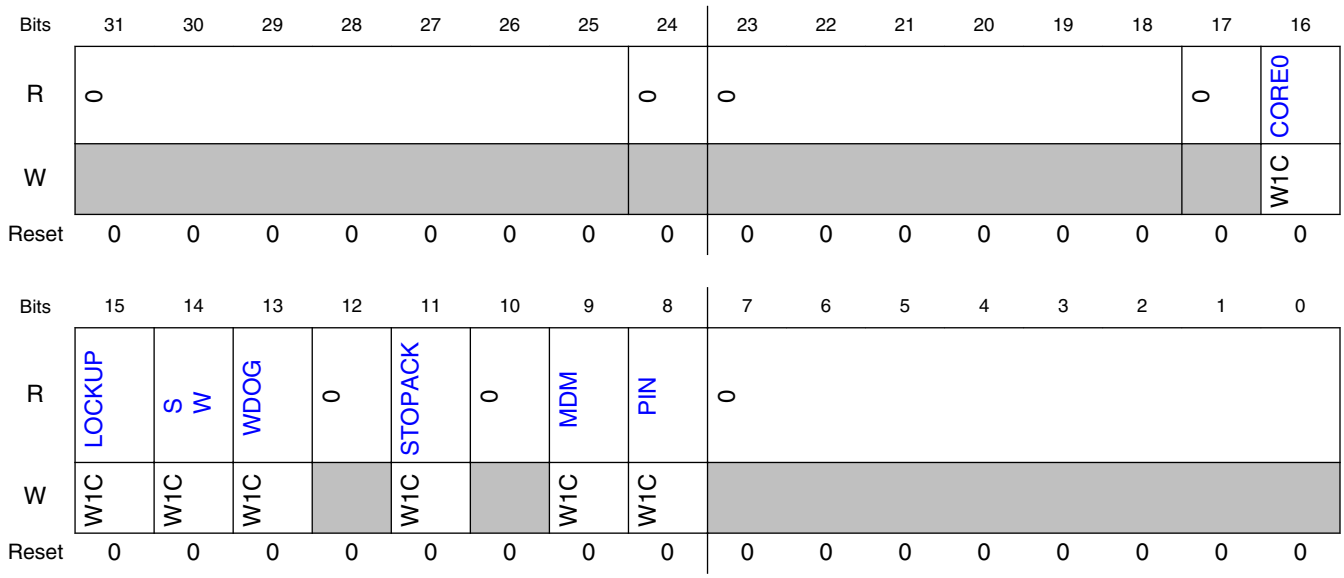
### 32.4.2.10.1 Offset

| Register | Offset |
|---|---|
| SRIF | 30h |

### 32.4.2.10.2 Function

This registers returns the source of the reset interrupt. The pending reset source can be cleared by resetting the source of the reset and then clearing the pending flag.

### 32.4.2.10.3 Diagram




### 32.4.2.10.4 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>CORE0 | Core0 Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 15<br>LOCKUP | Lockup Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 14<br>SW | Software Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 13<br>WDOG | Watchdog Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 12<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 11<br>STOPACK | Stop Timeout Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 10<br>— | Reserved |
| 9<br>MDM | MDM Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 8<br>PIN | Pin Reset<br>0b - Reset source not pending.<br>1b - Reset source pending. |
| 7-0<br>— | Reserved |

## 32.4.2.11 Mode Register (MR)

### 32.4.2.11.1 Offset

| Register | Offset |
|---|---|
| MR | 40h |

### 32.4.2.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | BOOTCFG | |
| W | | | | | | | | | | | | | | | W1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 32.4.2.11.3 Fields

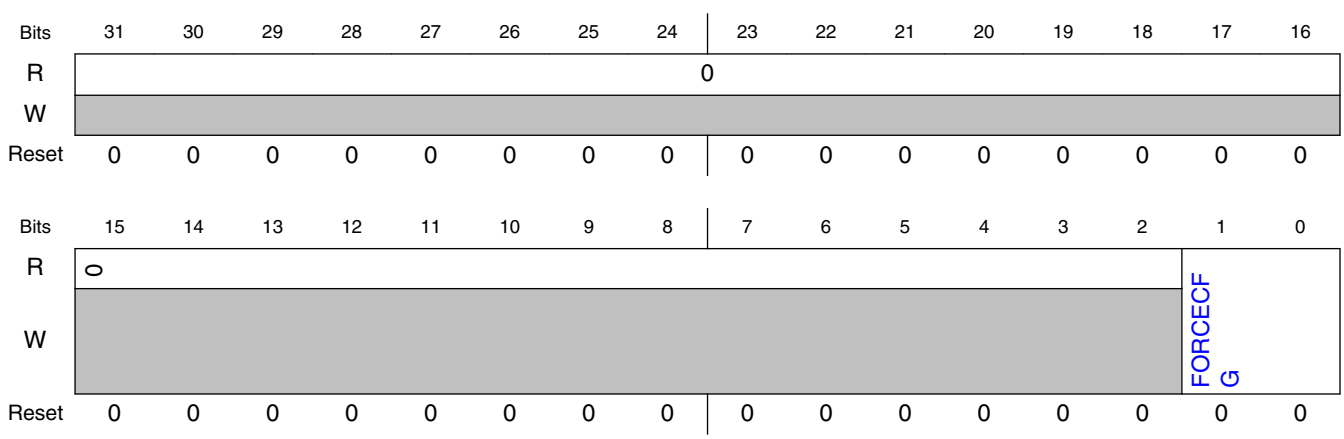
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>BOOTCFG | Boot Configuration<br>Indicates the boot source, the boot source remains set until the next System Reset or software can write logic one to clear the corresponding mode bit. The boot source is forced to Flash on a VLLS wakeup that does not include another system reset source.<br>While either bit is set, the vector take is relocated to the ROM base address. These bits should be cleared by writing logic one before executing any code from either Flash or SRAM.<br>00b - Boot from Flash.<br>01b - Boot from ROM due to BOOTCFG0 pin assertion.<br>10b - Boot from ROM due to FOPT configuration.<br>11b - Boot from ROM due to both BOOTCFG0 pin assertion and FOPT configuration. |

## 32.4.2.12 Force Mode Register (FM)

### 32.4.2.12.1 Offset

| Register | Offset |
|---|---|
| FM | 50h |

### 32.4.2.12.2 Diagram



#### 32.4.2.12.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>FORCECFG | Boot Configuration<br>This register can force the corresponding bit in the Mode Register to assert on next system reset.<br>00b - No effect.<br>01b - Assert corresponding bit in Mode Register on next system reset. |

### 32.4.2.13 SRAM Low Power Register (SRAMLPR)

#### 32.4.2.13.1 Offset

| Register | Offset |
|---|---|
| SRAMLPR | 60h |

#### 32.4.2.13.2 Function

This register places specific System SRAM arrays into a low power retention state whenever the corresponding register bit is set. The SRAM arrays should not be accessed while they are in a low power retention state.

#### 32.4.2.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 32.4.2.13.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LPE | Low Power Enable<br>0b - Corresponding SRAM array in Run mode.<br>1b - Corresponding SRAM array in low power retention state and should not be accessed. |

### 32.4.2.14 SRAM Deep Sleep Register (SRAMDSR)

#### 32.4.2.14.1 Offset

| Register | Offset |
|---|---|
| SRAMDSR | 64h |

#### 32.4.2.14.2 Function

This register configures if the System SRAM arrays are retained or powered off in Stop modes.

#### 32.4.2.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DSE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DSE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 32.4.2.14.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DSE | Deep Sleep Enable<br>0b - Corresponding SRAM array is retained in Stop modes.<br>1b - Corresponding SRAM array is powered off in Stop modes. |

## 32.5 Functional description

This section provides a complete functional description of the block.

### 32.5.1 Power mode transitions

A figure showing the power mode state transitions and a table defining triggers for the various state transitions can be found here.

The following figure shows the power mode state transitions available to each core. Any chip reset will initially bring both cores back to a RUN state. An exception is a POR event which could force core 1 to be disabled until re-enabled by core 0. In order to minimize peak power consumption, certain devices may also support reset to VLPR via nonvolatile configuration settings (e.g. Flash IFR or fuse), causing the MSMC to begin transitioning the MCU into VLPR mode during the reset recovery sequence.

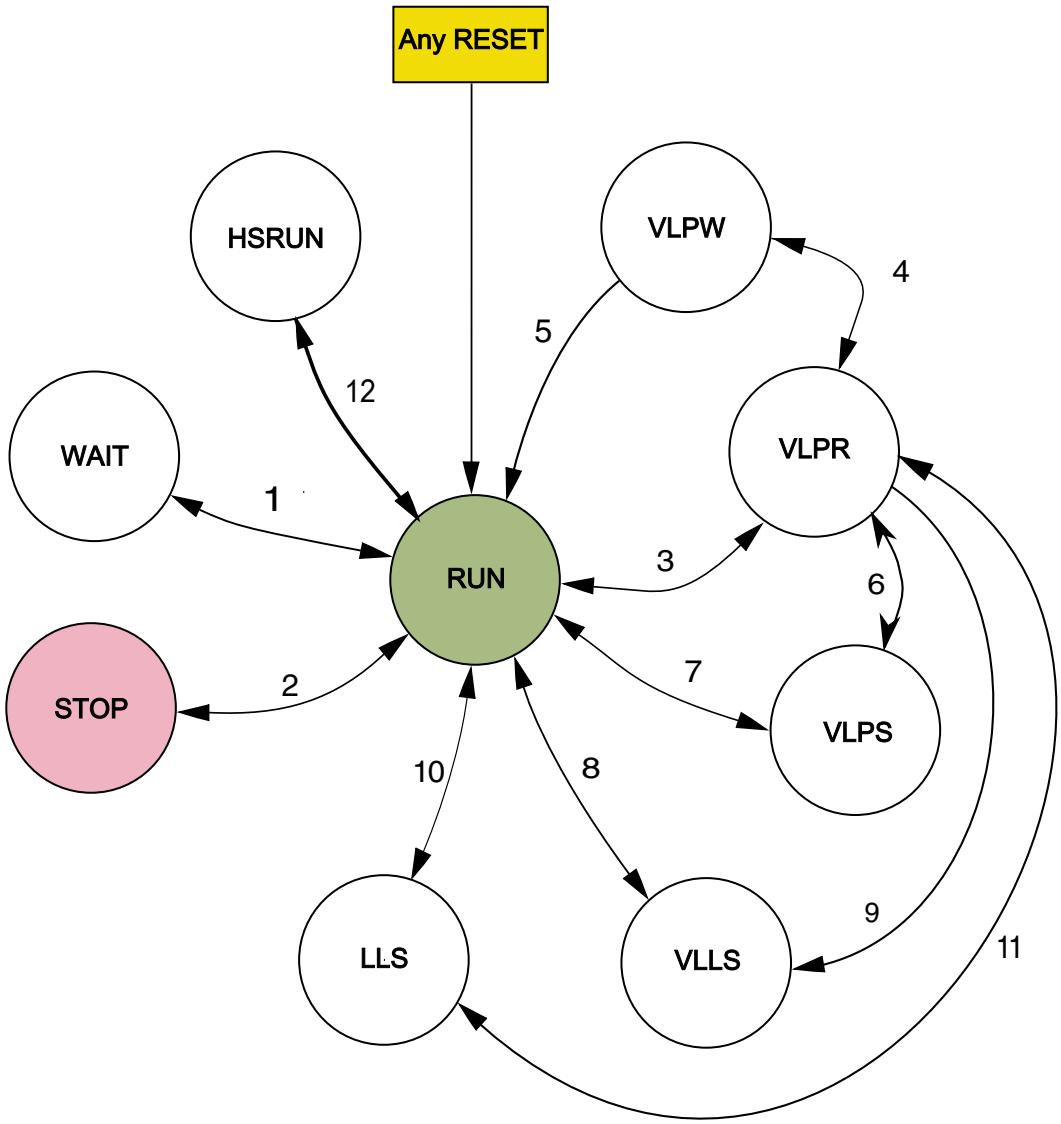



**Figure 32-2. Power mode state diagram**

The MSMC allows core 1 to enter/exit stop modes independently of core 0. For core 0 stop modes and the run mode of both cores, certain dependencies must be resolved before the MSMC will allow a power mode entry/exit to execute. As a general rule, the run mode for both cores will the highest power mode between the MSMC0 and MSMC1 RUNM settings. Likewise, the stop mode of core 0 will be the highest power mode between MSMC0 and MSMC1 STOPM settings, whereas the stop mode of core 1 will be

simply equal to the MSMC1 STOPM setting. The following tables describes in further detail the resulting power mode each core will receive for all combinations of STOPM and RUNM settings for both cores.

**Table 32-2. Core 0/1 RUNM result**

| MSMC0 PMCTRL[RUNM] setting | MSMC1 PMCTRL[RUNM] setting | Core 0 RUNM result[1] | Core 1 RUNM result |
|---|---|---|---|
| VLPR | VLPR | VLPR[2] | |
| | RUN | RUN | |
| | HSRUN | HSRUN | |
| RUN | VLPR/RUN | RUN | |
| | HSRUN | HSRUN | |
| HSRUN | VLPR/RUN/HSRUN | HSRUN | |

1. If Core 1 is disabled, the MSMC1 RUNM setting is ignored and Core 0 RUNM is simply equal to the MSMC0 RUNM setting
2. If a core has entered VLPS/LLS/VLLS mode with RUNM=RUN, VLPR mode can still be entered if the other core sets its RUNM=VLPR i.e. the RUNM field from a core in VLPS/LLS/VLLS mode is bypassed. Once the stopped core receives a wakeup event, the system will transition back to RUN mode.

**Table 32-3. Core0/1 STOPM result**

| MSMC0 STOPM setting | MSMC1 STOPM setting | Core 0 STOPM result[1] | Core 1 STOPM result |
|---|---|---|---|
| STOP/VLPS/LLS/VLLS | STOP | STOP | STOP |
| STOP | VLPS | STOP | VLPS |
| VLPS/LLS/VLLS | | VLPS | |
| STOP | LLS | STOP | LLS |
| VLPS | | VLPS | |
| LLS/VLLS | | LLS | |
| STOP | VLLS | STOP | VLLS |
| VLPS | | VLPS | |
| LLS | | LLS | |
| VLLS | | VLLS | |

1. If Core 1 is disabled, the MSMC1 STOPM setting is ignored and Core 0 STOPM is simply equal to the MSMC0 STOPM setting

The following table defines the trigger conditions required for each core to transition into or out of the power modes illustrated in the power mode state diagram above.

**Table 32-4. Power mode transition triggers for Core 0**

| Transition # | From | To | Trigger conditions[1] |
|---|---|---|---|
| 1 | RUN | WAIT | Sleep-now or sleep-on-exit executed with SLEEPDEEP=0 by Core 0.<br>See note.[2] |
| | WAIT | RUN | Core 0 Interrupt or Reset. |

*Table continues on the next page...*

## Table 32-4. Power mode transition triggers for Core 0 (continued)

| Transition # | From | To | Trigger conditions[1] |
|---|---|---|---|
| 2 | RUN | STOP | Core 0 STOPM=STOP[3] and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | STOP | RUN | Core 0 Interrupt or Reset |
| 3 | RUN | VLPR | Core 0 RUNM=VLPR.[4] |
| | VLPR | RUN | Core 0 RUNM=RUN or Reset. |
| 4 | VLPR | VLPW | Sleep-now or sleep-on-exit executed with SLEEPDEEP=0 by Core 0.<br>See note.[2] |
| | VLPW | VLPR | Core 0 Interrupt. |
| 5 | VLPW | RUN | Reset. |
| 6 | VLPR | VLPS | Core 0 STOPM=VLPS/STOP and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | VLPS | VLPR | Interrupt and VLPS mode was entered directly from VLPR. |
| 7 | RUN | VLPS | Core 0 STOPM=VLPS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | VLPS | RUN | Interrupt and VLPS mode was entered directly from RUN or<br>Reset. |
| 8 | RUN | VLLS | Core 0 STOPM=VLLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | VLLS | RUN | LLWU0 wakeup, NMI or RESET_B pin. |
| 9 | VLPR | VLLS | Core 0 STOPM=VLLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| 10 | RUN | LLS | Core 0 STOPM=LLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | LLS | RUN | LLWU0 wakeup, NMI pin (and LLS mode was entered directly from RUN) or<br>RESET_B pin. |
| 11 | VLPR | LLS | Core 0 STOPM=LLS and |

*Table continues on the next page...*

## Table 32-4. Power mode transition triggers for Core 0 (continued)

| Transition # | From | To | Trigger conditions[1] |
|---|---|---|---|
| | | | Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by both Core 0 and Core 1.<br>See note.[2] |
| | LLS | VLPR | LLWU0 wakeup or NMI pin (and LLS mode was entered directly from VLPR). |
| 12 | RUN | HSRUN | Core 0 RUNM=HSRUN |
| | HSRUN | RUN | Core 0 RUNM=RUN or Reset. |

1. "Core 0 STOPM" and "Core 0 RUNM" referenced in this table is the resulting setting after taking both MSMC0 and MSMC1 STOPM/RUNM settings into account (see previous tables).
2. If debug is enabled, the core clock remains to support debug.
3. If PMCTRL[STOPM]=STOP and PMCTRL[PSTOPO]=01 or 10 for either MSMC0 or MSMC1, then only a Partial Stop mode is entered instead of STOP
4. In order to limit peak current, certain devices allow PMPROT[AVLP] and PMCTRL[RUNM] bits to be set on any Reset via nonvolatile configuration settings (e.g. Flash IFR or fuse), causing the MSMC to transition the core from RUN->VLPR during the reset recovery sequence.

## Table 32-5. Power mode transition triggers for Core 1

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| 1 | RUN | WAIT | Sleep-now or sleep-on-exit executed with SLEEPDEEP=0 by Core 1.<br>See note.[1] |
| | WAIT | RUN | Core 1 Interrupt or Reset. |
| 2 | RUN | STOP | Core 1 STOPM=STOP[2] and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| | STOP | RUN | Core 1 Interrupt or Reset |
| 3 | RUN | VLPR | Core 1 RUNM=VLPR.[3] |
| | VLPR | RUN | Core 1 RUNM=RUN or Reset. |
| 4 | VLPR | VLPW | Sleep-now or sleep-on-exit executed with SLEEPDEEP=0 by Core 1.<br>See note.[1] |
| | VLPW | VLPR | Core 1 Interrupt. |
| 5 | VLPW | RUN | Reset. |
| 6 | VLPR | VLPS | Core 1 STOPM=VLPS/STOP and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| | VLPS | VLPR | Interrupt and VLPS mode was entered directly from VLPR. |
| 7 | RUN | VLPS | Core 1 STOPM=VLPS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1. |

*Table continues on the next page...*

**Table 32-5. Power mode transition triggers for Core 1 (continued)**

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| | | | See note.[1] |
| | VLPS | RUN | Interrupt and VLPS mode was entered directly from RUN or Reset. |
| 8 | RUN | VLLS | Core 1 STOPM=VLLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| | VLLS | RUN | LLWU1 wakeup, NMI or RESET_B pin. |
| 9 | VLPR | VLLS | Core 1 STOPM=VLLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| 10 | RUN | LLS | Core 1 STOPM=LLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| | LLS | RUN | LLWU1 wakeup, NMI pin (and LLS mode was entered directly from RUN) or<br>RESET_B pin. |
| 11 | VLPR | LLS | Core 1 STOPM=LLS and<br>Sleep-now or sleep-on-exit with SLEEPDEEP=1 executed by Core 1.<br>See note.[1] |
| | LLS | VLPR | LLWU1 wakeup or NMI pin (and LLS mode was entered directly from VLPR). |
| 12 | RUN | HSRUN | Core 1 RUNM=HSRUN |
| | HSRUN | RUN | Core 1 RUNM=RUN or Reset. |

1. If debug is enabled, the core clock remains to support debug.
2. If PMCTRL[STOPM]=STOP and PMCTRL[PSTOPO]=01 or 10 for MSMC1, then only a Partial Stop mode is entered instead of STOP
3. In order to limit peak current, certain devices allow PMPROT[AVLP] and PMCTRL[RUNM] bits to be set on any Reset via nonvolatile configuration settings (e.g. Flash IFR or fuse), causing the MSMC to transition the core from RUN->VLPR during the reset recovery sequence.

## 32.5.2 Run modes

The MSMC supports the following run modes:

- Normal Run (RUN)
- Very Low-Power Run (VLPR)
- High Speed Run (HSRUN)

#### 32.5.2.1 Run (RUN) mode

This is the normal operating mode for the device.

This mode is selected after any reset. In order to reduce peak power consumption, on certain devices, the MSMC can begin transitioning a core into VLPR mode during the reset recovery sequence if it has been configured for VLPR boot mode via nonvolatile configuration settings (e.g. Flash IFR or fuse). When the ARM processor exits reset, it sets up the stack, program counter (PC), and link register (LR):

- The processor reads the start SP (SP_main) from vector-table offset 0x000
- The processor reads the start PC from vector-table offset 0x004
- LR is set to 0xFFFF_FFFF.

To reduce power in this mode, clocks to unused modules can be disabled using their corresponding clock gating control bits in the Peripheral Clock Control modules (see PCC chapter) or compute operation can be enabled via the Miscellaneous Control Module (see MCM chapter).

#### 32.5.2.2 Very Low Power Run (VLPR) mode

VLPR mode is designed to provide the lowest power operating mode to the core at a reduced frequency. See the PMC chapter for additional power options available in this mode. Before entering this mode, the following conditions must be met:

- The maximum frequencies of the system, bus, flash, and core are restricted. See the Power Management details about which frequencies are supported.
- Mode protection must be set to allow VLP modes, that is, PMPROT[AVLP] is 1.
- PMCTRL[RUNM] must be set to VLPR in both SMC0 and SMC1 or, alternatively, one core is in VLPS/LLS/VLLS mode while the other core sets its PMCTRL[RUNM] to VLPR.
- Flash programming/erasing is not allowed.

**NOTE**

While in VLPR mode, do not modify the clock source in the SCG module or any clock divider registers. Module clock enables in the PCC can be set, however should only be cleared one at a time to limit load transitions.

To reenter Normal Run mode, clear PMCTRL[RUNM] in either SMC0 or SMC1. PMSTAT is a read-only status register that can be used to determine when the system has completed an exit to RUN mode. When PMSTAT=RUN, the system can run at full speed in any clock mode.

Any reset always causes an exit from VLPR and returns a core to RUN mode after the exiting its reset flow. Alternatively, certain devices can be configured for VLPR boot in which the MSMC will initiate VLPR entry during the reset flow.

To further reduce power in this mode, clocks to unused modules can be disabled using their corresponding clock gating control bits in the Peripheral Clock Control modules (see PCC chapter) or compute operation can be enabled via the Miscellaneous Control Module (see MCM chapter).

#### 32.5.2.3 High Speed Run (HSRUN) mode

HSRUN mode is designed to provide the highest performance mode to the core at the fastest supported frequency. See the PMC chapter for additional power options available in this mode. Before entering this mode, the following conditions must be met:

- The maximum allowable change in frequency of the system, bus, flash or core clocks is restricted to x2.
- Mode protection must be set to allow HSRUN modes, that is, PMPROT[AHSRUN] is 1.
- PMCTRL[RUNM] must be set to HSRUN in either SMC0 or SMC1.
- Flash programming/erasing is not allowed. (Not all devices contain on-chip flash.)
- Stop mode entry is not allowed.

To reenter normal RUN mode, clear the RUNM register in both the SMC0 and SMC1. PMSTAT is a read-only status register that can be used to determine when the system has completed an exit to RUN mode. Any reset will also clear RUNM and cause the core to exit to normal RUN mode after exiting its reset flow.

### 32.5.3 Wait modes

The MSMC supports the following wait modes.

- Wait (WAIT)
- Very-Low Power Wait (VLPW)

#### 32.5.3.1 Wait (WAIT) mode

WAIT mode is designed to allow a core to enter a static, low power state with instant wakeup times, while still allowing peripherals to operate with full functionality. WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit mode while SLEEDEEP is cleared. The core enters a low-power state in which it is not clocked, but peripherals continue to be clocked provided they are enabled. Clock gating to the peripheral is enabled via the PCC module.

When an interrupt request occurs, the core exits WAIT mode and resumes processing in RUN mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset will also cause an exit from WAIT mode, returning the core to normal RUN mode.

#### 32.5.3.2 Very Low Power Wait (VLPW) mode

VLPW mode is designed to allow a core to enter a static, very low power state, while still allowing peripherals to operate with full functionality at a reduced frequency. VLPW mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit mode while SLEEPDEEP is cleared and the core is in VLPR mode. The ARM CPU enters a low-power state in which it is not clocked, but peripherals continue to be clocked provided they are enabled. Clock gating to the peripheral is enabled via the PCC module.

VLPR mode restrictions also apply to VLPW.

When an interrupt from VLPW occurs, the device returns to VLPR mode to execute the interrupt service routine.

When an interrupt request occurs, the CPU exits VLPW mode and resumes processing in VLPR mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from VLPW mode, returning the device to normal RUN mode.

### 32.5.4 Stop modes

The MSMC supports a wide variety of stop modes for each core ranging from:

- a stopped core, with all I/O, logic, and memory states retained, and certain asynchronous mode peripherals operating

to:

- a powered down core, with only I/O and a small register file retained, very few asynchronous mode peripherals operating

The choice of stop mode depends upon the user's application, and how power usage and state retention versus functional needs and recovery time may be traded off.

The stop mode for a core can be selected by setting the appropriate fields in PMPROT and PMCTRL register fields in SMC0 and SMC1 (see "Power mode transitions" section for details).

The following stop modes are supported:

- Normal Stop (STOP)
- Very Low Power Stop (VLPS)
- Low Leakage Stop (LLS)
- Very Low Leakage Stop (VLLS)

#### 32.5.4.1 Stop (STOP) mode

STOP mode is designed to allow a core and its system to enter a static, low power state with relatively fast wakeup times, while still allowing peripherals to operate with full asynchronous functionality. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in STOP mode and see the PMC chapter for additional power options available in this mode.

A core can enter STOP mode by executing Sleep-Now or Sleep-On-Exit with SLEEPDEEP=1 while the core's STOPM equates to STOP (see "Power mode transitions" section for details). Note that a core's PSTOPO field can be set to enable a partial stop mode, allowing instantaneous wakeup at the expense of power consumption.

A module capable of providing an asynchronous interrupt can trigger an exit from STOP mode and return the device to normal RUN mode. When an interrupt request occurs, the core exits STOP mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will also cause an exit from STOP mode, returning the device to normal RUN mode.

### 32.5.4.2 Very Low Power Stop (VLPS) mode

VLPS mode is designed to allow a core and its system to enter a static, very low power state with relatively fast wakeup times, while still allowing peripherals to operate with limited asynchronous functionality. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in VLPS mode and see the PMC chapter for additional power options available in this mode.

A core can enter VLPS mode by executing Sleep-Now or Sleep-On-Exit with SLEEPDEEP=1 while the core's STOPM either equates to VLPS or equates to STOP with RUNM=VLPR (see "Power mode transitions" section for details).

A module capable of providing an asynchronous interrupt can trigger an exit from VLPS mode and return the device to the run mode from which VLPS was entered (i.e. RUN or VLPR mode). When an interrupt request occurs, the core exits VLPS mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will also cause an exit from VLPS mode, returning the device to normal RUN mode.

### 32.5.4.3 Low Leakage Stop (LLS) mode

LLS mode is designed to allow a core and its system to enter a low leakage, power gated state with relatively fast wakeup times, while still retaining the state of all logic. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in LLS mode and see the PMC chapter for additional power options available in this mode.

A core can enter LLS mode by executing Sleep-Now or Sleep-On-Exit with SLEEPDEEP=1 while the core's STOPM equates to LLS (see "Power mode transitions" section for details).

Before entering LLS mode, the user should configure the Low Leakage Wake-up unit (LLWU) to enable the desired pins and/or peripherals as wake-up sources . The available wake-up sources to the LLWU are detailed in the chip configuration details for this device.

Once configured, the LLWU can trigger an exit from LLS mode and return the device to the run mode from which LLS was entered (i.e. RUN or VLPR mode) with a pending LLWU interrupt. When an LLWU wakeup request occurs, the core exits LLS mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine. In the LLWU interrupt service routine (ISR), the user can read the LLWU module wake-up flags to determine the source of the wakeup.

A pin reset will also cause an exit from LLS mode, returning the device to normal RUN mode. When exiting LLS via the reset pin, the PIN and WAKEUP bits within the SRS register will be set.

**NOTE**

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully recover from LLS mode after an LLWU wakeup event.

### 32.5.4.4 Very Low Leakage Stop (VLLS) modes

VLLS mode is designed to allow a core and its system to enter a very low leakage, power gated state. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in VLLS mode and see the PMC chapter for additional power options available in this mode.

A core can enter VLLS mode by executing Sleep-Now or Sleep-On-Exit with SLEEPDEEP=1 while the core's STOPM equates to VLLS (see "Power mode transitions" section for details).

Before entering VLLS mode, the user should configure the Low Leakage Wake-up unit (LLWU) to enable the desired pins and/or peripherals as wake-up sources . The available wake-up sources to the LLWU are detailed in the chip configuration details for this device.

Once configured, the LLWU can trigger an exit from VLLS mode and return the device to RUN mode with a pending LLWU interrupt. When an LLWU wakeup request occurs, the core exits VLLS mode and begins executing the reset flow. In the LLWU interrupt service routine (ISR), the user can read the LLWU module wake-up flags to determine the source of the wakeup.

A pin reset will also cause an exit from VLLS mode, returning the device to normal RUN mode. When exiting VLLS via the reset pin, the PIN and WAKEUP bits within the SRS register will be set.

When entering VLLS, each I/O pin in the core's power domain is statically latched as configured before executing VLLS. Because all logic is powered off, all port and peripheral data is lost during VLLS. This data should be restored before writing the ACKISO bit in the power management module, after which I/O states are released to be driven by the restored logic.

### 32.5.5 Debug in low power modes

When the MCU is secure, the device disables/limits debugger operation. When the MCU is unsecure, the ARM debugger can assert two power-up request signals:

- System power up, via SYSPWR in the Debug Port Control/Stat register
- Debug power up, via CDBGPWRUPREQ in the Debug Port Control/Stat register

When asserted while in RUN, WAIT, VLPR, or VLPW, the MSMC drives a corresponding acknowledge for each signal i.e. CDBGPWRUPACK and CSYSPWRUPACK. When both requests are asserted, the mode controller handles attempts to enter STOP and VLPS by entering an emulated stop state. In this emulated stop state:

- all system clocks, except the core clock, are disabled,
- the debug module has access to core registers, and
- access to the on-chip peripherals is blocked.

No debug is available while the MCU is in LLS or VLLS modes. Since logic state is retained in LLS mode, all debug operation can continue after waking from LLS, even in cases where wakeup is due to a pin reset event.

Entering into a VLLS mode causes all of the debug controls and settings to be powered off. To give time to the debugger to sync with the MCU, the MDM AP Control Register includes a Very-Low-Leakage Debug Request (VLLDBGREQ) bit that is set to configure the Reset Controller Module (RCM) to hold the system in reset after the next recovery from a VLLS mode. This bit allows the debugger time to reinitialize the debug module before the debug session continues.

The MDM AP Control Register also includes a Very Low Leakage Debug Acknowledge (VLLDBGACK) bit that is set to release the ARM core being held in reset following a VLLS recovery. The debugger reinitializes all debug IP, and then asserts the VLLDBGACK control bit to allow the RCM to release the ARM core from reset and allow CPU operation to begin.

The VLLDBGACK bit is cleared by the debugger (or can be left set as is) or clears automatically due to the reset generated as part of the next VLLS recovery.

# Chapter 33
# System Power Manager (SPM)

## 33.1 Chip-specific SPM information

### 33.1.1 VDDIO voltage monitor

In this device, the VDDIO voltage monitor is connected to the VDDIO2 supply.

## 33.2 Introduction

The System Power Manager (SPM) contains and controls all three regulators including the CORE LDO, RF LDO, and DCDC. It manages switching between these regulators based on power mode change requests from MSMC or RADIO. In addition SPM module contains the POR (Power-on Reset), low voltage detect (LVD), and the high voltage detect (HVD) circuit which monitor and assure correct voltage levels.

## 33.3 Features

A list of included SPM features can be found here.

- Selectable Internal voltage regulator (CORE LDO, RF LDO, and DCDC)
- Active POR providing brown-out detect
- Low-voltage detect supporting two low-voltage trip points with four warning levels per trip point
- High-voltage detect supporting two high-voltage trip points
- Core Power Gate Control
- Radio Power Gate Control

The next figure shows the SPM module block diagram.

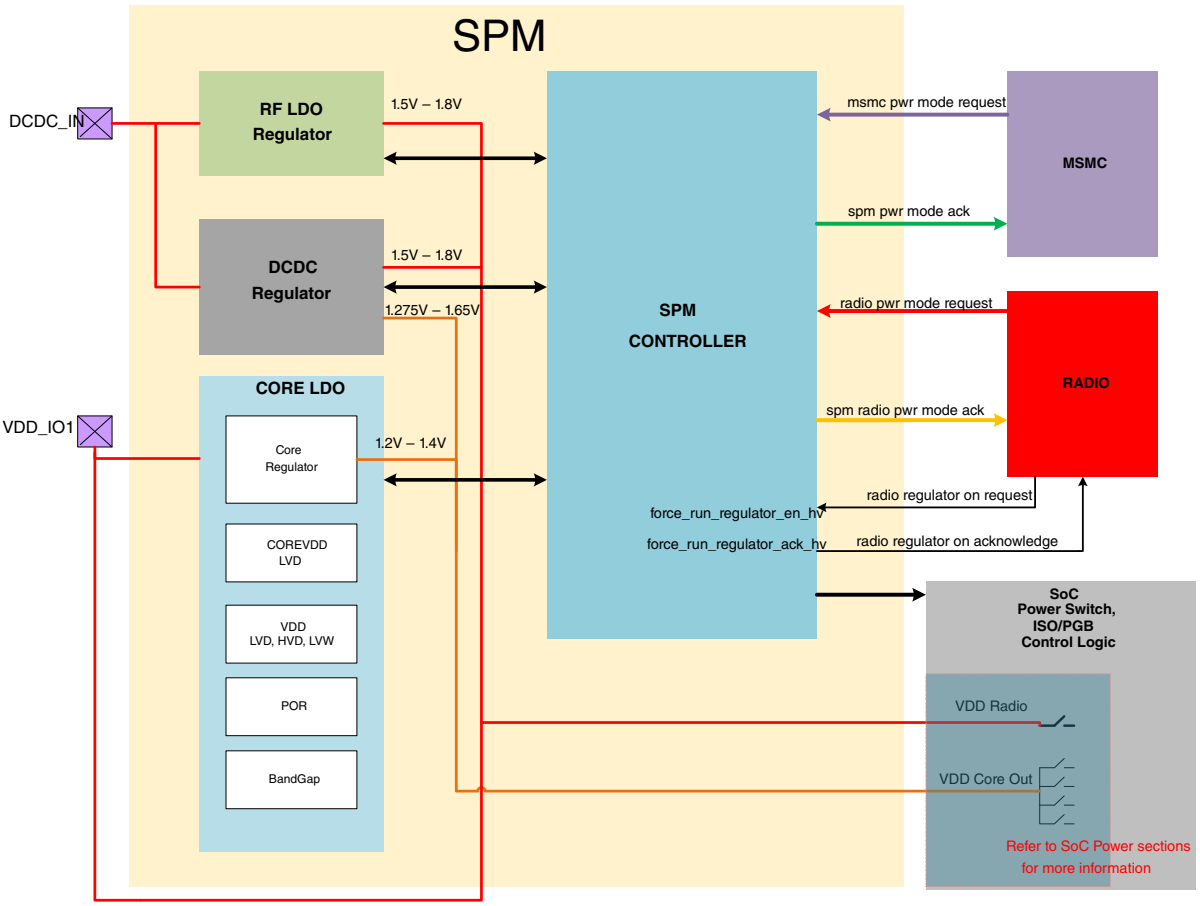



**Figure 33-1. System Power Manager (SPM) block diagram**

The next table shows description of SPM block diagram signals.

**Table 33-1. SPM Block Diagram Signals**

| | TYPE | DESCRIPTION |
|---|---|---|
| spm_msmc_lp_stop_ack_hv | out | spm msmc lp stop request ack |
| spm_msmc_lp_run_ack_hv | out | spm msmc lp run req ack |
| spm_msmc_hs_run_ack_hv | out | spm msmc high speed run request ack |
| msmc_lp_stop_req_hv | in | msmc lp stop request |
| msmc_lp_stop_mode_hv[2:0] | in | msmc lp stop mode bus (tells spm stop type) |
| msmc_lp_run_req_hv | in | msmc lp run request |
| msmc_hs_run_req_hv | in | msmc high speed run request |
| spm_radio_lp_stop_ack_hv | out | spm radio stop request ack |
| radio_lp_stop_req_hv | in | radio lp stop request |
| radio_lp_stop_mode_hv[2:0] | in | radio lp stop mode bus (tells spm stop type) |

*Table continues on the next page...*

**Table 33-1. SPM Block Diagram Signals (continued)**

| | TYPE | DESCRIPTION |
|---|---|---|
| force_run_regulator_req_hv | in | Backdoor request to enable regulator selected in the RCTRL[REGSEL] |
| force_run_regulator_ack_hv | out | Acknowledge to "force_run_regulator_req_hv" request |
| spm_swvdd_on_hv | out | Core VDD switch control |
| spm_swvddc_on_hv | out | Core VDDC switch control |
| spm_wbias_vdd_en_hv | out | Core VDD well bias enable |
| spm_wbias_vddc_en_hv | out | Core VDDC well bias enable |
| spm_iso_hv | out | Core ISO |
| spm_pgb_hv | out | Core SRPG gating control signal |
| spm_radio_swvdd_on_hv | out | Radio VDD switch control |
| spm_radio_swvddc_on_hv | out | Radio VDDC switch control |
| spm_radio_wbias_vdd_en_hv | out | Radio VDD well bias enable |
| spm_radio_wbias_vddc_en_hv | out | Radio VDDC well bias enable |
| spm_radio_iso_hv | out | Radio ISO |
| spm_radio_pgb_hv | out | Radio SRPG gating control signal |

## 33.4 Memory map and register descriptions

Details about the SPM registers can be found here.

### 33.4.1 SPM register descriptions

**NOTE**

Different portions of SPM registers are reset only by particular reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

The SPM registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

**Do not write to any SPM address space that is not defined by registers below.**

### 33.4.1.1 SPM Memory map

SPM base address: 4002_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0000_0000h |
| 8h | Regulator Status Register (RSR) | 32 | RO | 0400_0003h |
| 10h | Run Control Register (RCTRL) | 32 | RW | 0000_0003h |
| 14h | Low Power Control Register (LPCTRL) | 32 | RW | 0000_0003h |
| 100h | CORE LDO RUN Configuration Register (CORERCNFG) | 32 | RW | 0007_0000h |
| 104h | CORE LDO Low Power Configuration register (CORELPCNFG) | 32 | RW | 0007_4000h |
| 108h | Core LDO Status And Control register (CORESC) | 32 | RW | See description. |
| 10Ch | Low Voltage Detect Status and Control 1 register (LVDSC1) | 32 | RW | 0010_0010h |
| 110h | Low Voltage Detect Status and Control 2 register (LVDSC2) | 32 | RW | 0000_0000h |
| 114h | High Voltage Detect Status And Control 1 register (HVDSC1) | 32 | RW | 0001_0000h |
| 200h | RF LDO Low Power Configuration register (RFLDOLPCNFG) | 32 | RW | 0000_0000h |
| 204h | RF LDO Status And Control register (RFLDOSC) | 32 | RW | See description. |
| 304h | DCDC Status Control Register (DCDCSC) | 32 | RW | 0418_0000h |
| 30Ch | DCDC Control Register 1 (DCDCC1) | 32 | RW | 005F_001Ch |
| 310h | DCDC Control Register 2 (DCDCC2) | 32 | RW | 0000_4009h |
| 314h | DCDC Control Register 3 (DCDCC3) | 32 | RW | 0000_0001h |
| 318h | DCDC Control Register 4 (DCDCC4) | 32 | RW | 0000_0000h |
| 320h | DCDC Control Register 6 (DCDCC6) | 32 | RW | 0D00_0608h |
| 40Ch | LP Request Pin Control Register (LPREQPINCNTRL) | 32 | RW | 0000_0000h |

### 33.4.1.2 Version ID Register (VERID)

#### 33.4.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 33.4.1.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

#### 33.4.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 33.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | MAJOR<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | MINOR<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000000b - Standard features implemented. |

### 33.4.1.3 Regulator Status Register (RSR)

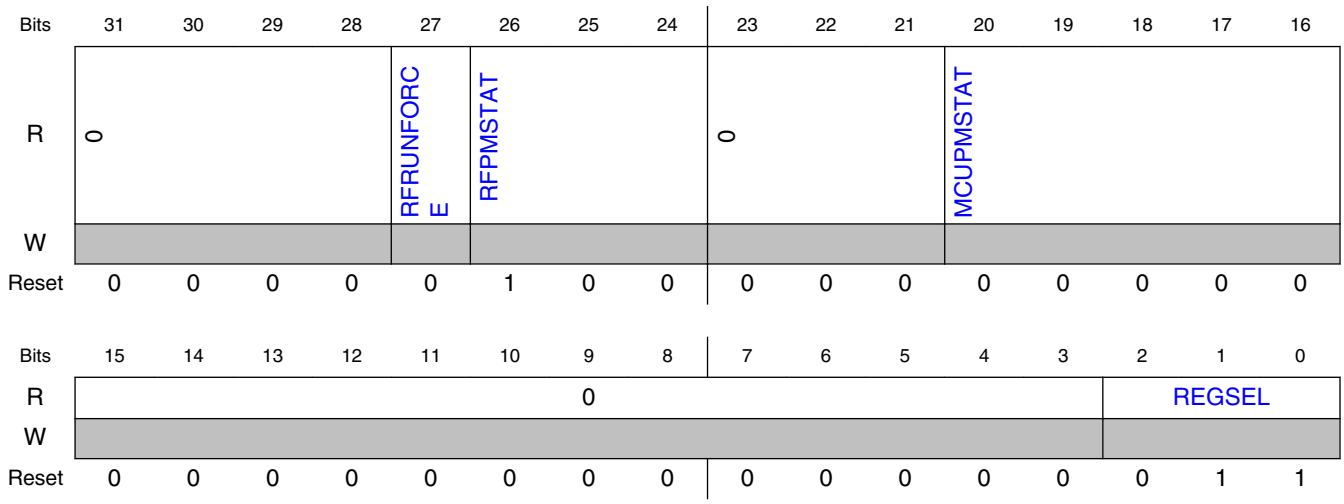
#### 33.4.1.3.1 Offset

| Register | Offset |
|---|---|
| RSR | 8h |

### 33.4.1.3.2 Function

This register returns the currently configured system regulator source. The RSR reflects the configuration set by one of the two control registers RCTRL or LPCTRL.

### 33.4.1.3.3 Diagram




### 33.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>RFRUNFORCE | RADIO Run Force Power Mode Status<br>Status of Radio Run Force configuration indicating if the RADIO block is enabling the regulator.<br>0b - Radio Run Force Regulator Off<br>1b - Radio Run Force Regulator On. |
| 26-24<br>RFPMSTAT | RADIO Power Mode Status<br>Status of last RADIO Power Mode Configuration.<br>000b - Reserved<br>001b - Current Power mode is VLPS.<br>010b - Current Power mode is LLS.<br>100b - Current Power mode is VLLS. |
| 23-21<br>— | Reserved |
| 20-16<br>MCUPMSTAT | MCU Power Mode Status<br>Status of last MCU STOP Mode Power Configuration.<br>00000b - Reserved<br>00001b - Last Low Power mode is STOP.<br>00010b - Last Low Power mode is VLPS.<br>00100b - Last Low Power mode is LLS. |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>01000b - Last Low Power mode is VLLS23.<br>10000b - Last Low Power mode is VLLS01.</td></tr>
<tr><td>15-3<br>—</td><td>Reserved</td></tr>
<tr><td>2-0<br>REGSEL</td><td>REGSEL<br>Indicates which regulators are regulators are enabled or disabled. REGSEL[0] identifies the CORE LDO regulator, REGSEL[1] identifies the RF LDO regulator, REGSEL[2] identifies the DCDC regulator.<br>Valid configurations are tabled below:<br>Table 33-2. SPM Regulator Status
<table>
<tr><th>DCDC</th><th>RF LDO</th><th>CORE LDO</th><th>Notes</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>All regulators disabled. Extenal PMIC selected</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Invalid</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Invalid</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>CORE AND RF LDO regulator enabled</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>DCDC regulator enabled</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Invalid</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Invalid</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>CORE LDO, RF LDO, and DCDC regulator enabled.</td></tr>
</table>
</td></tr>
</table>

## 33.4.1.4 Run Control Register (RCTRL)

### 33.4.1.4.1 Offset

| Register | Offset |
|---|---|
| RCTRL | 10h |

### 33.4.1.4.2 Function

This register controls which regulator (CORE LDO, RF LDO, and DCDC) are enabled in RUN and HSRUN modes.

**NOTE**

The RCTRL bits are reset solely on a POR/LVD only event.

### 33.4.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | REGSEL | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 33.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>REGSEL | REGSEL<br>Selects which regulators are regulators are used in RUN mode. REGSEL[0] controls the CORE LDO regulator, REGSEL[1] controls the RF LDO regulator, REGSEL[2] controls the DCDC regulator.<br>Valid configurations are tabled below: |

**Table 33-3. SPM Regulator Select**

| DCDC | RF LDO | CORE LDO | Notes |
|---|---|---|---|
| 0 | 0 | 0 | All regulators disabled. Select extenal PMIC |
| 0 | 0 | 1 | Invalid |
| 0 | 1 | 0 | Invalid |
| 0 | 1 | 1 | CORE AND RF LDO regulator enabled |
| 1 | 0 | 0 | DCDC regulator enabled |
| 1 | 0 | 1 | Invalid |
| 1 | 1 | 0 | Invalid |
| 1 | 1 | 1 | CORE LDO, RF LDO, and DCDC regulator enabled. The CORE and RF LDO regulators are automatically set at a voltage lower than the DCDC output. |

## 33.4.1.5 Low Power Control Register (LPCTRL)

### 33.4.1.5.1 Offset

| Register | Offset |
|---|---|
| LPCTRL | 14h |

### 33.4.1.5.2 Function

This register controls which regulator (CORE LDO, RF LDO, and DCDC) are enabled in Low Power modes.

This register can be written only in RUN mode.

**NOTE**

The LPCTRL bits are reset solely on a POR/LVD only event.

### 33.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | REGSEL | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 33.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>REGSEL | REGSEL<br>Selects which regulators are enabled in low power mode.. REGSEL[0] controls the CORE LDO regulator, REGSEL[1] controls the RF LDO regulator, REGSEL[2] controls the DCDC regulator. |

| Field | Function |
|---|---|
| | Valid configurations are tabled below:<br>**Table 33-4. SPM Regulator Select**<br>DCDC / RF LDO / CORE LDO / Notes<br>0 / 0 / 0 / All regulators disabled. Select extenal PMIC<br>0 / 0 / 1 / Invalid<br>0 / 1 / 0 / Invalid<br>0 / 1 / 1 / CORE AND RF LDO regulator enabled<br>1 / 0 / 0 / Invalid<br>1 / 0 / 1 / Invalid<br>1 / 1 / 0 / Invalid<br>1 / 1 / 1 / Invalid |

## 33.4.1.6 CORE LDO RUN Configuration Register (CORERCNFG)

### 33.4.1.6.1 Offset

| Register | Offset |
|---|---|
| CORERCNFG | 100h |

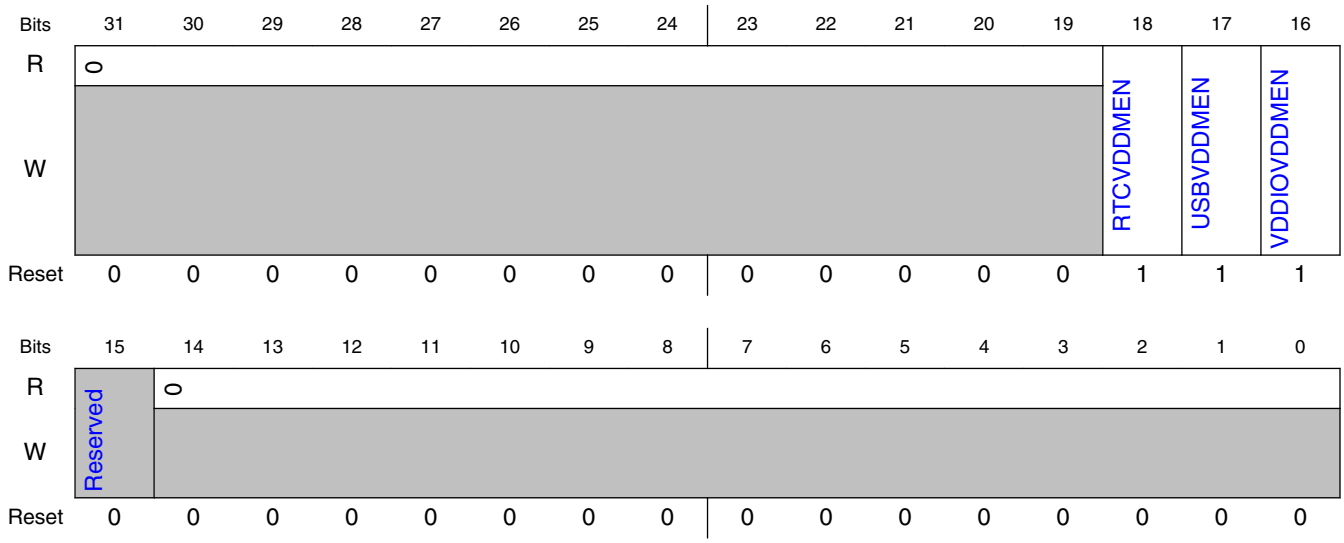
### 33.4.1.6.2 Function

This register configures core Low Drop-out regulator in run modes.

**NOTE**

The CORERCNFG bits are reset solely on a POR/LVD only event.

### 33.4.1.6.3 Diagram




### 33.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>RTCVDDMEN | RTCVDDMEN<br>RTC Voltage Monitor Enable register<br>0b - RTC voltage monitor disabled in run modes.<br>1b - RTC voltage monitor enabled in run modes. |
| 17<br>USBVDDMEN | USBVDDMEN<br>USB Voltage Monitor Enable register<br>0b - USB voltage monitor disabled in run modes.<br>1b - USB voltage monitor enabled in run modes. |
| 16<br>VDDIOVDDMEN | VDDIOVDDMEN<br>VDDIO Voltage Monitor Enable register.<br>0b - VDDIO voltage monitor disabled in run modes.<br>1b - VDDIO voltage monitor enabled in run modes. |
| 15<br>— | Reserved |
| 14-0<br>— | Reserved |

# 33.4.1.7 CORE LDO Low Power Configuration register (CORELPCNFG)

## 33.4.1.7.1 Offset

| Register | Offset |
|---|---|
| CORELPCNFG | 104h |

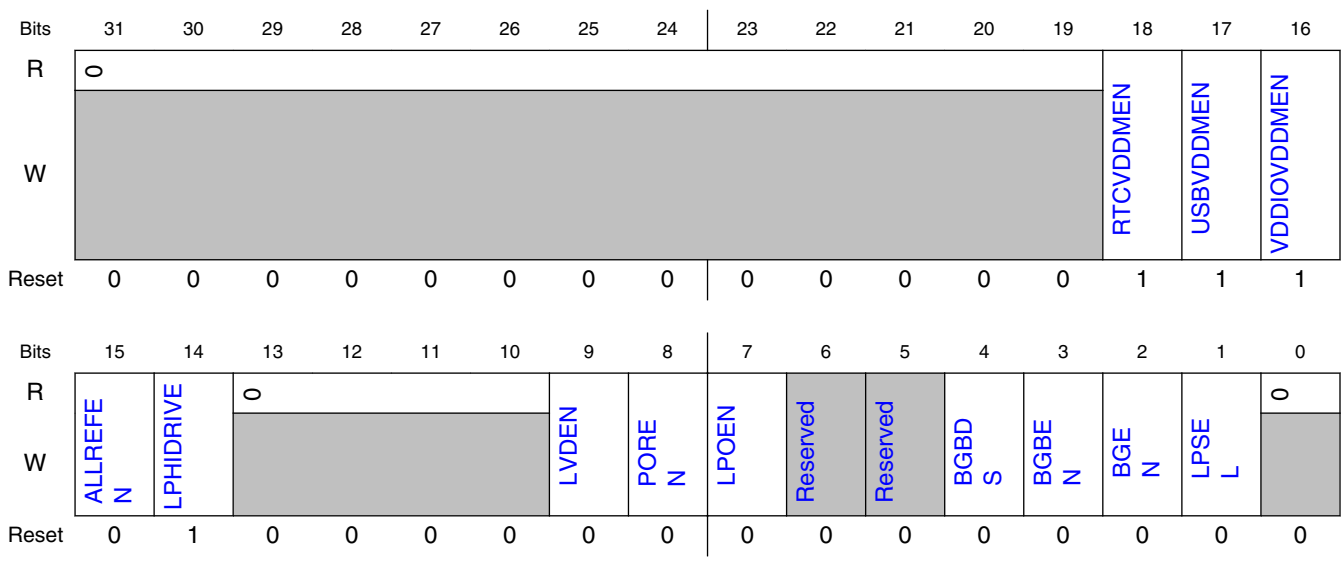
## 33.4.1.7.2 Function

This register configures core LDO in low power modes.

**NOTE**

The CORELPCNFG bits are reset solely on a POR/LVD only event.

## 33.4.1.7.3 Diagram




## 33.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>RTCVDDMEN | RTCVDDMEN<br>RTC Voltage Monitor Enable register |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - RTC voltage monitor disabled in lp modes.<br>1b - RTC voltage monitor enabled in lp modes. Note: voltage monitor is always disabled in VLLS0/1 modes. |
| 17<br>USBVDDMEN | USBVDDMEN<br>USB Voltage Monitor Enable register<br>0b - USB voltage monitor disabled in lp modes.<br>1b - USB voltage monitor enabled in lp modes. Note: voltage monitor is always disabled in VLLS0/1 modes. |
| 16<br>VDDIOVDDMEN | VDDIOVDDMEN<br>VDDIO Voltage Monitor Enable register.<br>0b - VDDIO voltage monitor disabled in lp modes.<br>1b - VDDIO voltage monitor enabled in lp modes. Note: voltage monitor is always disabled in VLLS0/1 modes. |
| 15<br>ALLREFEN | All Reference Enable. This bit only has an affect in VLLS0/1.<br>Turns off all the low power references. When this bit is clear the RFLDO, LPCMP and LPO are not available.<br>POREN must bet set if ALLREFEN is to be set.<br>0b - All references are disabled in VLLS.<br>1b - All references are enabled in VLLS0/1. |
| 14<br>LPHIDRIVE | LPHIDRIVE<br>Low Power High Drive register. It is recommended that this bit remain set to 1.<br>0b - High Drive disabled.<br>1b - High Drive enabled. |
| 13-10<br>— | Reserved |
| 9<br>LVDEN | LVD Enabled<br>Enables Low Voltage Detect (LVD). The BGEN must be set when this bit is set if the LVD is to remain enabled in low power modes.<br>0b - LVD/HVD is disabled in low power modes.<br>1b - LVD/HVD remains enabled in low power modes. BGEN must be set when this bit is also set. |
| 8<br>POREN | POR Enabled<br>Enables the POR. POR brownout is always enabled when COREREG is enabled (NOT VLLS0/1 stop modes). This bit can only be written to zero if ALLREFEN and LPOEN are also written to zero.<br>0b - POR brownout is disabled in VLLS0/1 mode.<br>1b - POR brownout remains enabled in VLLS0/1 mode. |
| 7<br>LPOEN | LPO Enabled<br>Enables the LPO. LPO is always enabled when COREREG is enabled (NOT VLLS0/1 stop modes). If LPOEN is to be set to 1, ALLREFEN must also be set to 1.<br>0b - LPO is disabled in VLLS modes.<br>1b - LPO remains enabled in VLLS modes. |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>BGBDS | Bandgap Buffer Drive Select<br>Selects the bandgap buffer drive. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE:** If the bandgap buffer is disabled (BGBE is clear), clear BGBDS.<br>0b - Low Drive<br>1b - High Drive |
| 3<br>BGBEN | Bandgap Buffer Enable<br>Enables the bandgap buffer.<br>0b - Bandgap buffer not enabled<br>1b - Bandgap buffer enabled BGEN must be set when this bit is also set. |
| 2<br>BGEN | Bandgap Enable In Low Power Mode Operation<br>BGEN controls whether the bandgap is enabled in lower power modes of operation (VLPx, LLS, and VLLSx). When on-chip peripherals require the bandgap voltage reference in low power modes of operation, set BGEN to continue to enable the bandgap operation. If it is required to keep the regulators in high power mode when operating the system in a low power mode (LPSEL = 1), BGEN must also be set to 1.<br>**NOTE:** When the bandgap voltage reference is not needed in low power modes, clear BGEN to avoid excess power consumption.<br>0b - Bandgap is disabled in STOP/VLP/LLS and VLLS modes.<br>1b - Bandgap remains enabled in STOP/VLP/LLS and VLLS modes. |
| 1<br>LPSEL | LPSEL<br>Core LDO Regulator Low Power Select register.<br>0b - Core LDO enters low power state in VLP/Stop modes.<br>1b - Core LDO remains in high power state in VLP/Stop modes. If LPSEL = 1 in a low power mode then BGEN must also be set to 1. |
| 0<br>— | Reserved |

## 33.4.1.8 Core LDO Status And Control register (CORESC)

### 33.4.1.8.1 Offset

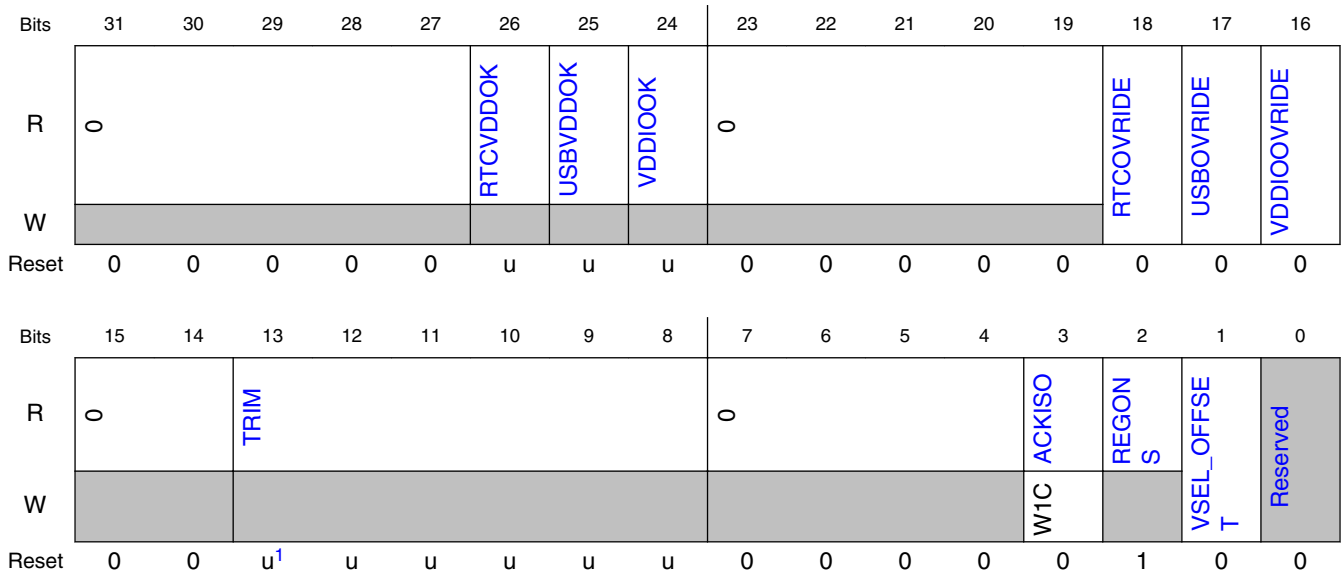
| Register | Offset |
|---|---|
| CORESC | 108h |

### 33.4.1.8.2 Function

**NOTE**

CORESC bits only reset on POR/LVD with the exception of ACKISO bit. ACKISO reset by any system reset.

### 33.4.1.8.3 Diagram




1. Loaded from IFR

### 33.4.1.8.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>RTCVDDOK | RTCVDDOK<br>Read only bit "Observe signal for VBAT". RTCVDDMEN must be set in CORERCNFG |
| 25<br>USBVDDOK | USBVDDOK<br>Read only bit "Observe signal for USB VREGIN". USBVDDMEN must be set in CORERCNFG |
| 24<br>VDDIOOK | VDDIOOK<br>Read only bit "Observe signal for VDDIO". VDDIOVDDMEN must be set in CORERCNFG. |
| 23-19<br>— | Reserved |
| 18<br>RTCOVRIDE | RTCOVRIDE<br>RTC Voltage Monitor override register. Sets the default state of RTC when its monitor is disabled.<br>0b - RTCVDDOK status set to 1'b0.<br>1b - RTCVDDOK status set to 1'b1. |
| 17<br>USBOVRIDE | USBOVRIDE<br>USB Voltage Monitor override register. Sets the default state of USB when its monitor is disabled.<br>0b - USBVDDOK status set to 1'b0.<br>1b - USBVDDOK status set to 1'b1. |
| 16<br>VDDIOOVRIDE | VDDIOOVRIDE<br>VDDIO Voltage Monitor override register. Sets the default state of VDDIO when its monitor is disabled.<br>0b - VDDIOOK status set to 1'b0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - VDDIOOK status set to 1'b1. |
| 15-14<br>— | Reserved |
| 13-8<br>TRIM | Core LDO Regulator TRIM value<br>Returns currently configured trim value. |
| 7-4<br>— | Reserved |
| 3<br>ACKISO | Acknowledge Isolation<br>Reading this field indicates whether certain peripherals and the I/O pads are in a latched state as a result of having been in VLLS mode. Writing 1 to this field when it is set releases the I/O pads and certain peripherals to their normal run mode state. ((Write 1 to Clear = W1C)<br>**NOTE:** After recovering from a VLLS mode, user should restore chip configuration before clearing ACKISO. In particular, pin configuration for enabled LLWU wakeup pins should be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.<br>0b - Peripherals and I/O pads are in normal run state.<br>1b - Certain peripherals and I/O pads are in a isolated and latched state. |
| 2<br>REGONS | CORE LDO Regulator in Run Regulation Status<br>This read-only field provides the current status of the CORE LDO voltage regulator.<br>0b - Regulator is in low power state or in transition to/from it.<br>1b - Regulator is in high power state. |
| 1<br>VSEL_OFFSET | Voltage Offset Select<br>Forces the Core LDO to an offset voltage level. This bit should only be set when in DCDC mode (RCTRL = 0x7).<br>0b - Core LDO offset not applied.<br>1b - Core LDO offset is applied. |
| 0<br>— | Reserved |

## 33.4.1.9 Low Voltage Detect Status and Control 1 register (LVDSC1)

### 33.4.1.9.1 Offset

| Register | Offset |
|---|---|
| LVDSC1 | 10Ch |

### 33.4.1.9.2 Function

This register contains status and control bits to support the low voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

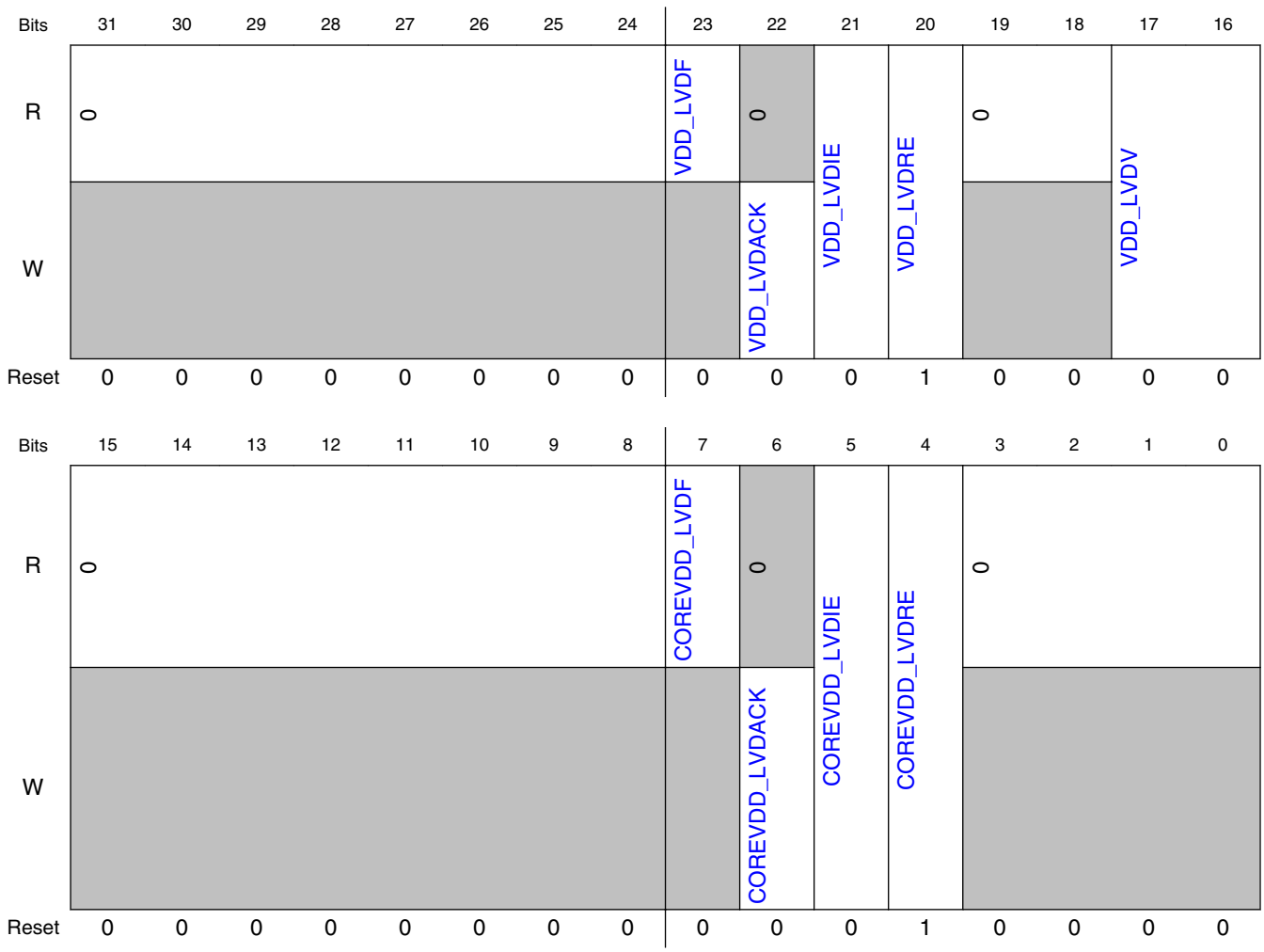
Whether or not LVD is on in low power mode depends on MCU and RF mode arbitration and related configuration in CORELPCNFG. To protect systems that must have LVD always on, configure the Power Mode Protection (PMPROT) register of the SMC module (SMC_PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact LVD trip voltages.

**NOTE**

The LVDV bits are reset solely on a POR Only event. The register's other bits are reset on POR or LVD event. For more information about these reset types, refer to the Reset section details.

### 33.4.1.9.3 Diagram



### 33.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>VDD_LVDF | VDD Low-Voltage Detect Flag<br>This read-only status field indicates a VDD low-voltage detect event.<br>0b - Low-voltage event not detected<br>1b - Low-voltage event detected |
| 22<br>VDD_LVDACK | VDD Low-Voltage Detect Acknowledge<br>This write-only field is used to acknowledge low voltage detection errors. Write 1 to clear VDD_LVDF. Reads always return 0. |
| 21<br>VDD_LVDIE | VDD Low-Voltage Detect Interrupt Enable<br>Enables hardware interrupt requests for VDD LVDF.<br>0b - Hardware interrupt disabled (use polling)<br>1b - Request a hardware interrupt when VDD_LVDF = 1 |
| 20<br>VDD_LVDRE | VDD Low-Voltage Detect Reset Enable<br>This write-once bit enables VDD_LVDF events to generate a hardware reset. Additional writes are ignored.<br>0b - VDD_LVDF does not generate hardware resets<br>1b - Force an MCU reset when VDD_LVDF = 1 |
| 19-18<br>— | Reserved |
| 17-16<br>VDD_LVDV | VDD Low-Voltage Detect Voltage Select<br>Selects the VDD LVD trip point voltage ($V_{LVD}$).<br>00b - Low trip point selected ($V_{LVD} = V_{LVDL}$)<br>01b - High trip point selected ($V_{LVD} = V_{LVDH}$)<br>10b - Reserved<br>11b - Reserved |
| 15-8<br>— | Reserved |
| 7<br>COREVDD_LVDF | Low-Voltage Detect Flag<br>This read-only status field indicates a low-voltage detect event.<br>0b - Low-voltage event not detected<br>1b - Low-voltage event detected |
| 6<br>COREVDD_LVDACK | Low-Voltage Detect Acknowledge<br>This write-only field is used to acknowledge low voltage detection errors. Write 1 to clear LVDF. Reads always return 0. |
| 5<br>COREVDD_LVDIE | Low-Voltage Detect Interrupt Enable<br>Enables hardware interrupt requests for LVDF.<br>0b - Hardware interrupt disabled (use polling)<br>1b - Request a hardware interrupt when LVDF = 1 |
| 4<br>COREVDD_LVDRE | Core Low-Voltage Detect Reset Enable<br>This write-once bit enables CORE_LVDF events to generate a hardware reset. Additional writes are ignored.<br>0b - COREVDD_LVDF does not generate hardware resets<br>1b - Force an MCU reset when CORE_LVDF = 1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>— | Reserved |

## 33.4.1.10 Low Voltage Detect Status and Control 2 register (LVDSC2)

### 33.4.1.10.1 Offset

| Register | Offset |
|---|---|
| LVDSC2 | 110h |

### 33.4.1.10.2 Function

This register contains status and control bits to support the low voltage warning function.

Whether or not LVD system is on in low power mode depends on MCU and RF mode arbitration and related configurations in CORELPCNFG.

See the device's data sheet for the exact LVD trip voltages.

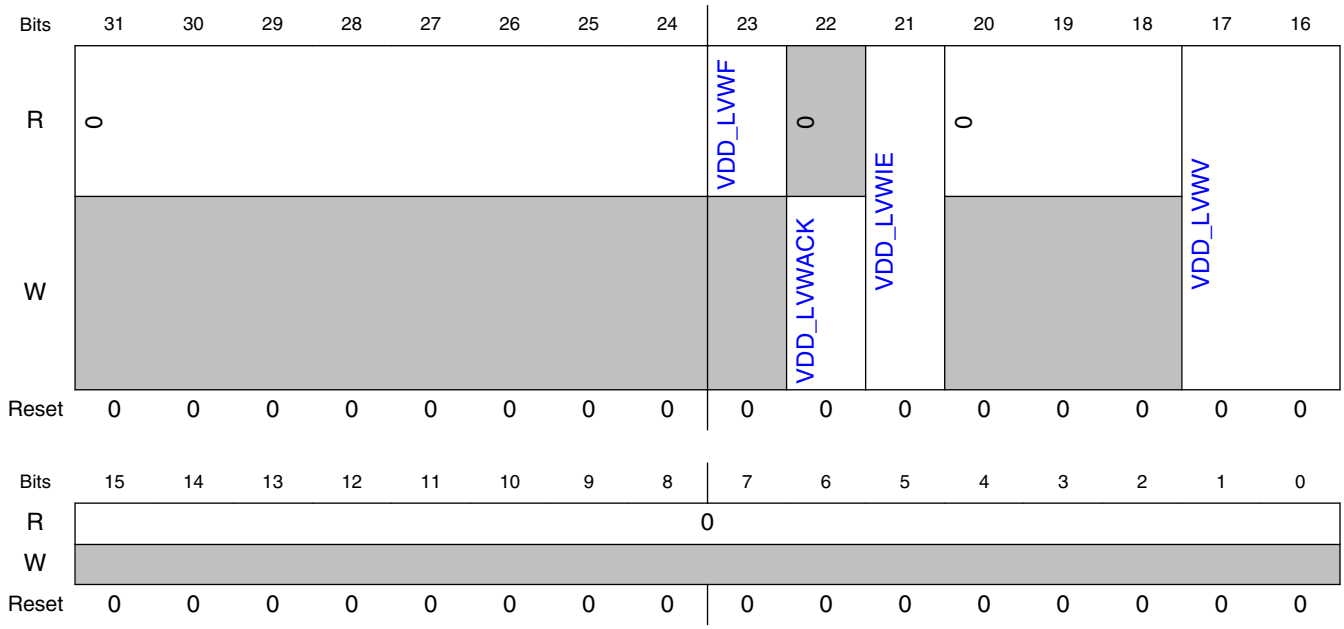
**NOTE**

The LVW trip voltages depend on LVWV and LVDV.

**NOTE**

LVWV is reset solely on a POR Only event. The other fields of the register are reset on POR or LVD only event. For more information about these reset types, refer to the Reset section details.

## 33.4.1.10.3 Diagram




## 33.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>VDD_LVWF | VDD Low-Voltage Warning Flag<br>This read-only status field indicates a VDD low-voltage warning event.<br>0b - Low-voltage warning event not detected<br>1b - Low-voltage warning event detected |
| 22<br>VDD_LVWACK | VDD Low-Voltage Warning Acknowledge<br>This write-only field is used to acknowledge low voltage warning errors. Write 1 to clear VDD_LVWF. Reads always return 0. |
| 21<br>VDD_LVWIE | VDD Low-Voltage Warning Interrupt Enable<br>Enables hardware interrupt requests for VDD LVWF.<br>0b - Hardware interrupt disabled (use polling)<br>1b - Request a hardware interrupt when VDD_LVWF = 1 |
| 20-18<br>— | Reserved |
| 17-16<br>VDD_LVWV | VDD Low-Voltage Warning Voltage Select<br>Selects the LVW trip point voltage ($V_{LVW}$). The actual voltage for the warning depends on LVDSC1[VDD_LVDV].<br>00b - Low trip point selected ($V_{LVW} = V_{LVW1}$)<br>01b - Mid 1 trip point selected ($V_{LVW} = V_{LVW2}$)<br>10b - Mid 2 trip point selected ($V_{LVW} = V_{LVW3}$)<br>11b - High trip point selected ($V_{LVW} = V_{LVW4}$) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-0<br>— | Reserved |

### 33.4.1.11 High Voltage Detect Status And Control 1 register (HVDSC1)

#### 33.4.1.11.1 Offset

| Register | Offset |
|---|---|
| HVDSC1 | 114h |

#### 33.4.1.11.2 Function

This register contains status and control bits to support the high voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

Whether or not HVD is on in low power mode depends on MCU and RF mode arbitration and related configuration in CORELPCNFG. To protect systems that must have HVD always on, configure the Power Mode Protection (PMPROT) register of the SMC module (SMC_PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact HVD trip voltages.

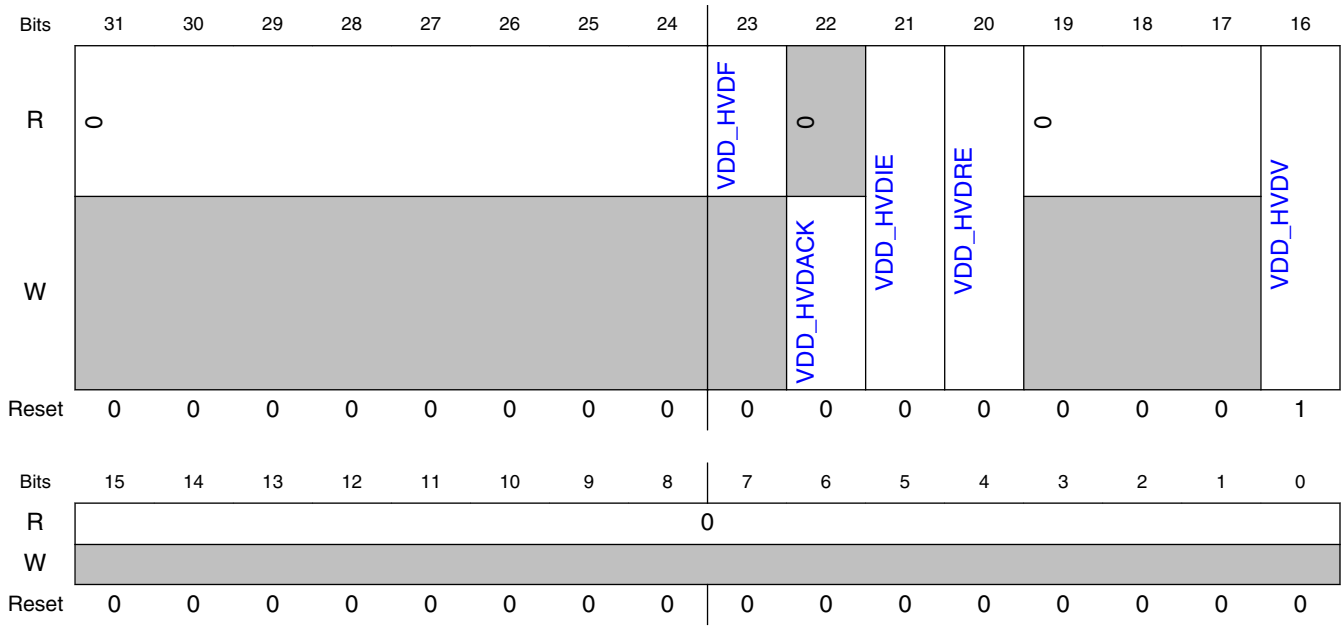
**NOTE**

This register is reset solely on a POR Only event. For more information about these reset types, refer to the Reset section details.

**NOTE**

The HVDV and HVDRE field is reset solely on a POR only event. The register's other fields are reset on POR or LVD only event. For more information about these reset types, refer to the Reset section details.

## 33.4.1.11.3 Diagram




## 33.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>VDD_HVDF | VDD High-Voltage Detect Flag<br>This read-only status field indicates a VDD high-voltage detect event.<br>0b - Vdd High-voltage event not detected<br>1b - Vdd High-voltage event detected |
| 22<br>VDD_HVDACK | VDD High-Voltage Detect Acknowledge<br>This write-only field is used to acknowledge VDD high voltage detection errors. Write 1 to clear VDD_HVDF. Reads always return 0. |
| 21<br>VDD_HVDIE | VDD High-Voltage Detect Interrupt Enable<br>Enables hardware interrupt requests for VDD_HVDF.<br>0b - Hardware interrupt disabled (use polling)<br>1b - Request a hardware interrupt when HVDF = 1 |
| 20<br>VDD_HVDRE | VDD High-Voltage Detect Reset Enable<br>This write-once bit enables VDD HVDF events to generate a hardware reset. Additional writes are ignored until the next chip reset.<br>0b - VDD HVDF does not generate hardware resets<br>1b - Force an MCU reset when VDD_HVDF = 1 |
| 19-17<br>— | Reserved |
| 16<br>VDD_HVDV | VDD High-Voltage Detect Voltage Select<br>Selects the VDD HVD trip point voltage ($V_{VDD_HVD}$). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Low trip point selected ($V_{VDD} = V_{VDD_HVDL}$)<br>1b - High trip point selected ($V_{VDD} = V_{VDD_HVDH}$) |
| 15-0<br>— | Reserved |

## 33.4.1.12 RF LDO Low Power Configuration register (RFLDOLPCNFG)

### 33.4.1.12.1 Offset

| Register | Offset |
|---|---|
| RFLDOLPCNFG | 200h |

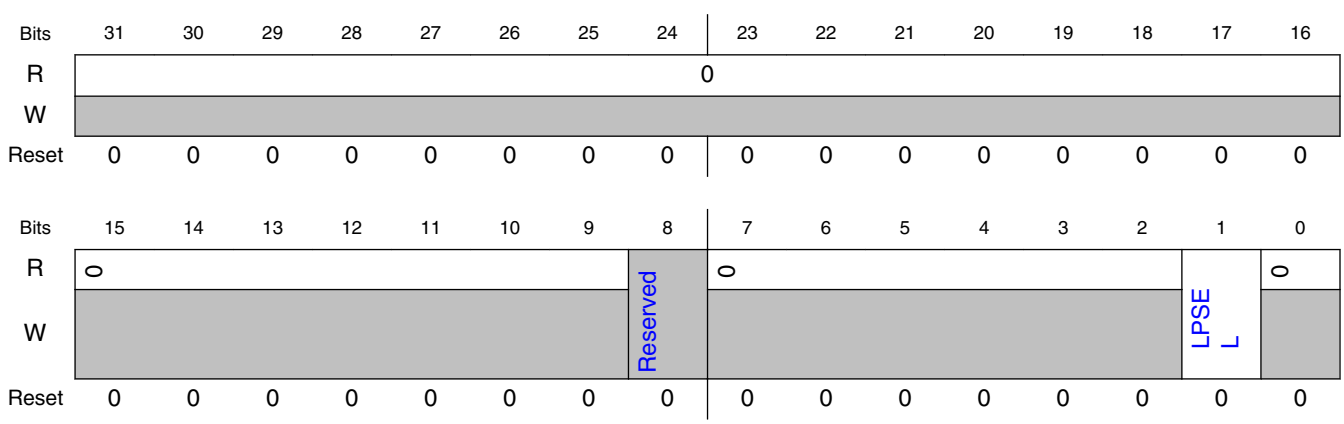
### 33.4.1.12.2 Function

This register controls RF LDO regulator in low power modes.

**NOTE**

The bits are reset solely on a POR/LVD only event.

### 33.4.1.12.3 Diagram




### 33.4.1.12.4 Fields

| Field | Function |
|---|---|
| 31-9 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1<br>LPSEL | LPSEL<br>RF LDO Low Power Select register.<br>0b - RF LDO regulator enters low power state in VLP/Stop modes.<br>1b - RF LDO regulator remains in high power state in VLP/Stop modes. |
| 0<br>— | Reserved |

## 33.4.1.13 RF LDO Status And Control register (RFLDOSC)

### 33.4.1.13.1 Offset

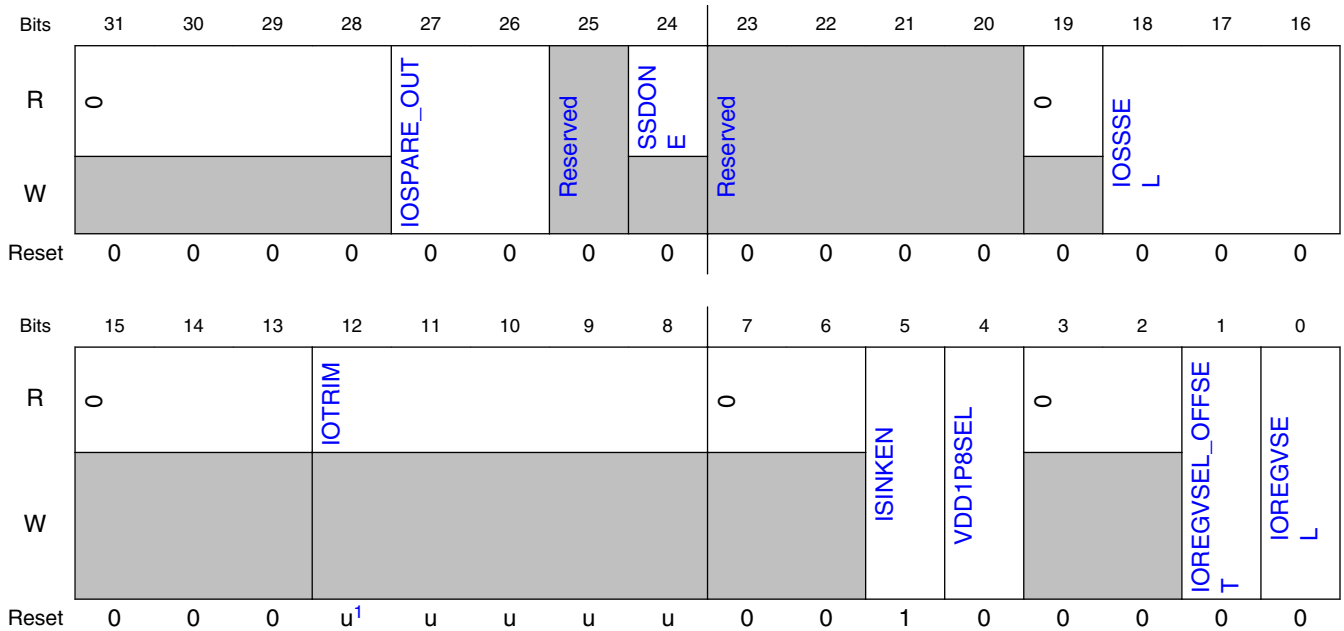
| Register | Offset |
|---|---|
| RFLDOSC | 204h |

### 33.4.1.13.2 Function

**NOTE**

The bits are reset solely on a POR/LVD only event.

### 33.4.1.13.3 Diagram




1. Loaded from IFR

### 33.4.1.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-26<br>IOSPARE_OUT | IO Spare Outputs |
| 25<br>— | Reserved |
| 24<br>SSDONE | IO Soft Start Done Status Registers |
| 23-20<br>— | Reserved |
| 19<br>— | Reserved |
| 18-16<br>IOSSSEL | IO 1.8 Reg Soft Start Select<br>Selects the soft start duration delay for the IO 1.8 full power regulator.<br>000b - Soft Start duration set to 110us.<br>001b - Soft Start duration set to 95us.<br>010b - Soft Start duration set to 60us.<br>011b - Soft Start duration set to 48us.<br>100b - Soft Start duration set to 38us.<br>101b - Soft Start duration set to 30us. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 110b - Soft Start duration set to 24us.<br>111b - Soft Start duration set to 17us. |
| 15-13<br>— | Reserved |
| 12-8<br>IOTRIM | IO Regulator TRIM value<br>Returns currently configured trim value for IO regulator. |
| 7-6<br>— | Reserved |
| 5<br>ISINKEN | ISINKEN<br>Enables current sink feature of low power regulator.<br>0b - Disable current sink feature of low power regulator.<br>1b - Enable current sink feature of low power regulator. |
| 4<br>VDD1P8SEL | VDD 1p8 SNS Pin Select<br>Selects which sns 1p8 vdd pin is used.<br>0b - VDD1p8_SNS0 selected.<br>1b - VDD1p8_SNS1 selected. |
| 3-2<br>— | Reserved |
| 1<br>IOREGVSEL_OFFSET | IO Regulator Offset Voltage Select<br>Forces IO regulator voltage to an offset level. This bit should only be set when the DCDC is enabled (RCTRL = 0x7).<br>0b - The IOREG offset is not applied.<br>1b - The IOREG offset is applied. |
| 0<br>IOREGVSEL | IO Regulator Voltage Select<br>Sets IO regulator voltage level.<br>0b - Regulate to 1.8V.<br>1b - Regulate to 1.5V. |

## 33.4.1.14 DCDC Status Control Register (DCDCSC)

### 33.4.1.14.1 Offset

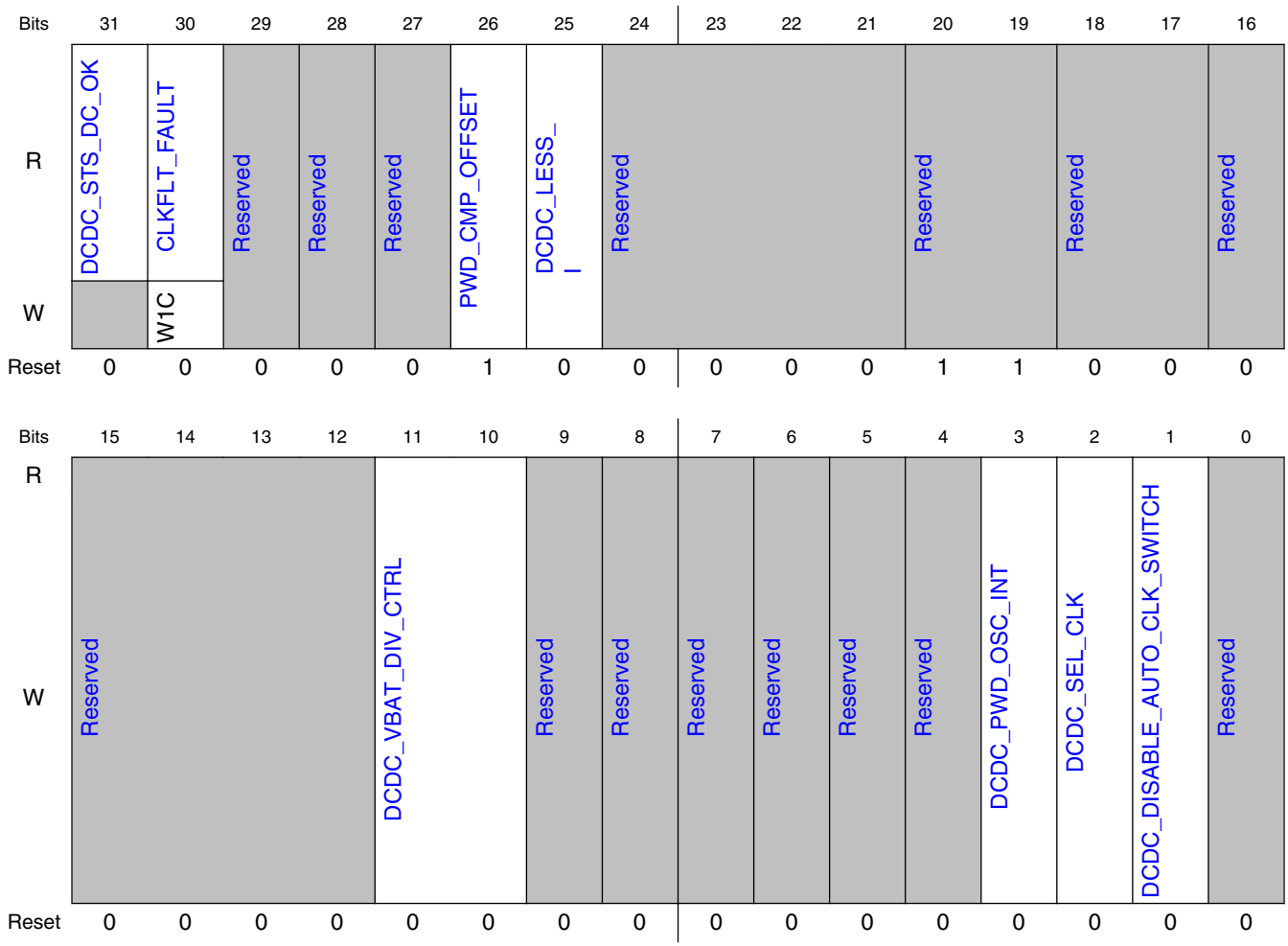
| Register | Offset |
|---|---|
| DCDCSC | 304h |

### 33.4.1.14.2 Function

**NOTE**

The bits are reset solely on a POR/LVD only event.

### 33.4.1.14.3 Diagram




### 33.4.1.14.4 Fields

| Field | Function |
|---|---|
| 31<br>DCDC_STS_DC_OK | DCDC_STS_DC_OK<br>Status register to indicate DCDC lock. The lock time depends on the loading and the DCDC mode. When changing output voltage target, it will take approximately (0.5ms* target change steps). In pulsed mode (if supported), it will take approximately 5ms. In startup, it will take approximately 50ms. |
| 30<br>CLKFLT_FAULT | DCDC CLKFLT Fault Status Flag<br>Asserts if DCDC detect a clk fault. Will cause a system lvd reset to assert.<br>**NOTE:** This read only bit is reset only by POR. |
| 29<br>— | Reserved |
| 28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27<br>— | Reserved |
| 26<br>PWD_CMP_OFFSET | PWD_CMP_OFFSET<br>Power down output range comparator |
| 25<br>DCDC_LESS_I | DCDC_LESS_I<br>Reduce DCDC current. It will save approximately 20 µA in RUN. |
| 24-21<br>— | Reserved |
| 20-19<br>— | Reserved |
| 18-17<br>— | Reserved |
| 16<br>— | Reserved<br>Reserved |
| 15-12<br>— | Reserved<br>Reserved |
| 11-10<br>DCDC_VBAT_DIV_CTRL | DCDC_VBAT_DIV_CTRL<br>Controls VBAT voltage divider. The divided VBAT output is input to an ADC channel which allows the battery voltage to be measured.<br>00b - OFF<br>01b - VBAT<br>10b - VBAT / 2<br>11b - VBAT / 4 |
| 9<br>— | Reserved |
| 8<br>— | Reserved<br>Reserved |
| 7<br>— | Reserved<br>Reserved |
| 6<br>— | Reserved<br>Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>DCDC_PWD_OSC_INT | DCDC_PWD_OSC_INT<br>Power down internal oscillator. Only set this bit when 32M crystal oscillator is available. |
| 2 | DCDC_SEL_CLK<br>Select external clock for DCDC when DCDC_DISABLE_AUTO_CLK_SWITCH is set. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DCDC_SEL_CLK | |
| 1<br>DCDC_DISABLE_AUTO_CLK_SWITCH | DCDC_DISABLE_AUTO_CLK_SWITCH<br>Disable automatic clock switch from internal oscillator to external clock. |
| 0<br>— | Reserved<br>Reserved |

## 33.4.1.15 DCDC Control Register 1 (DCDCC1)

### 33.4.1.15.1 Offset

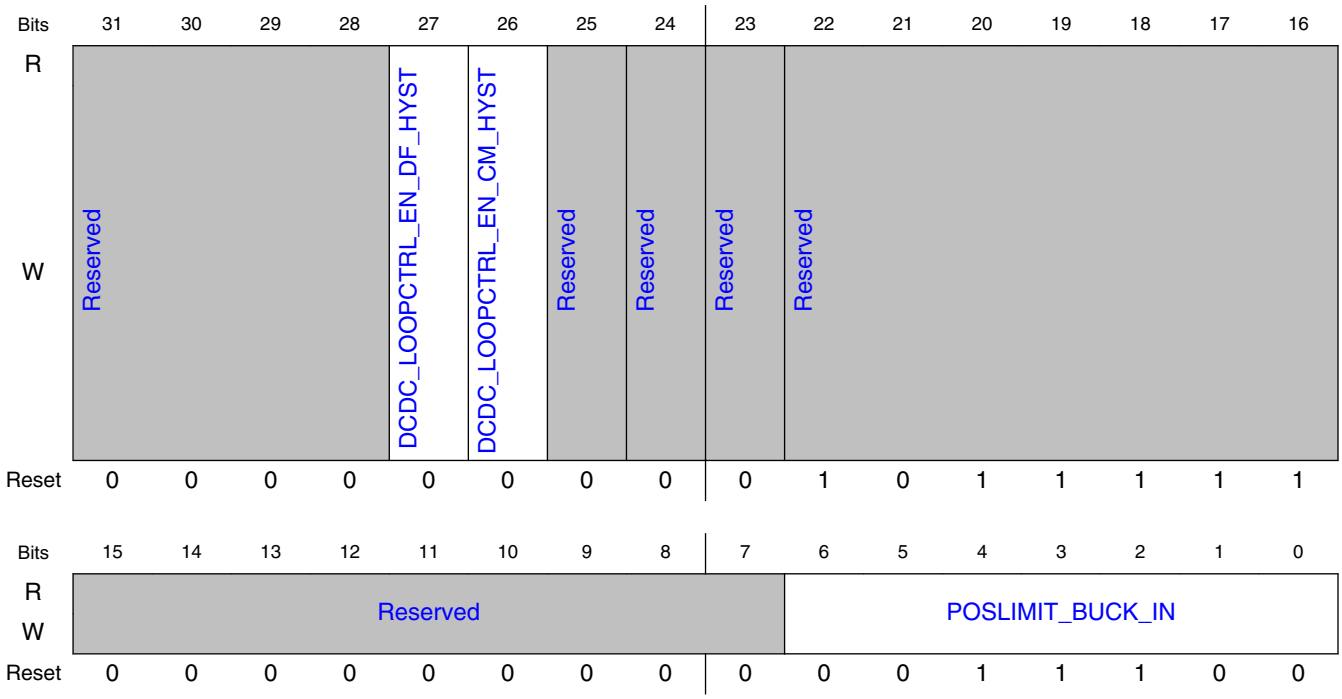
| Register | Offset |
|---|---|
| DCDCC1 | 30Ch |

### 33.4.1.15.2 Function

**NOTE**

The bits are reset solely on a POR/LVD only event.

## 33.4.1.15.3 Diagram




## 33.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved<br>Reserved |
| 27<br>DCDC_LOOPCTRL_EN_DF_HYST | DCDC_LOOPCTRL_EN_DF_HYST<br>Enable hysteresis in switching converter differential mode analog comparators. This feature improves transient supply ripple and efficiency. |
| 26<br>DCDC_LOOPCTRL_EN_CM_HYST | DCDC_LOOPCTRL_EN_CM_HYST<br>Enable hysteresis in switching converter common mode analog comparators. This feature improves transient supply ripple and efficiency. |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved<br>Reserved. |
| 22-16<br>— | Reserved<br>Reserved. |
| 15-7 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | Reserved. |
| 6-0<br>POSLIMIT_BUCK_IN | POSLIMIT_BUCK_IN<br>Upper limit duty cycle limit in DC-DC converter. This field limits the maximum VDDIO achievable for a given battery voltage, and its value may be increased if very low battery operation is met. |

## 33.4.1.16 DCDC Control Register 2 (DCDCC2)

### 33.4.1.16.1 Offset

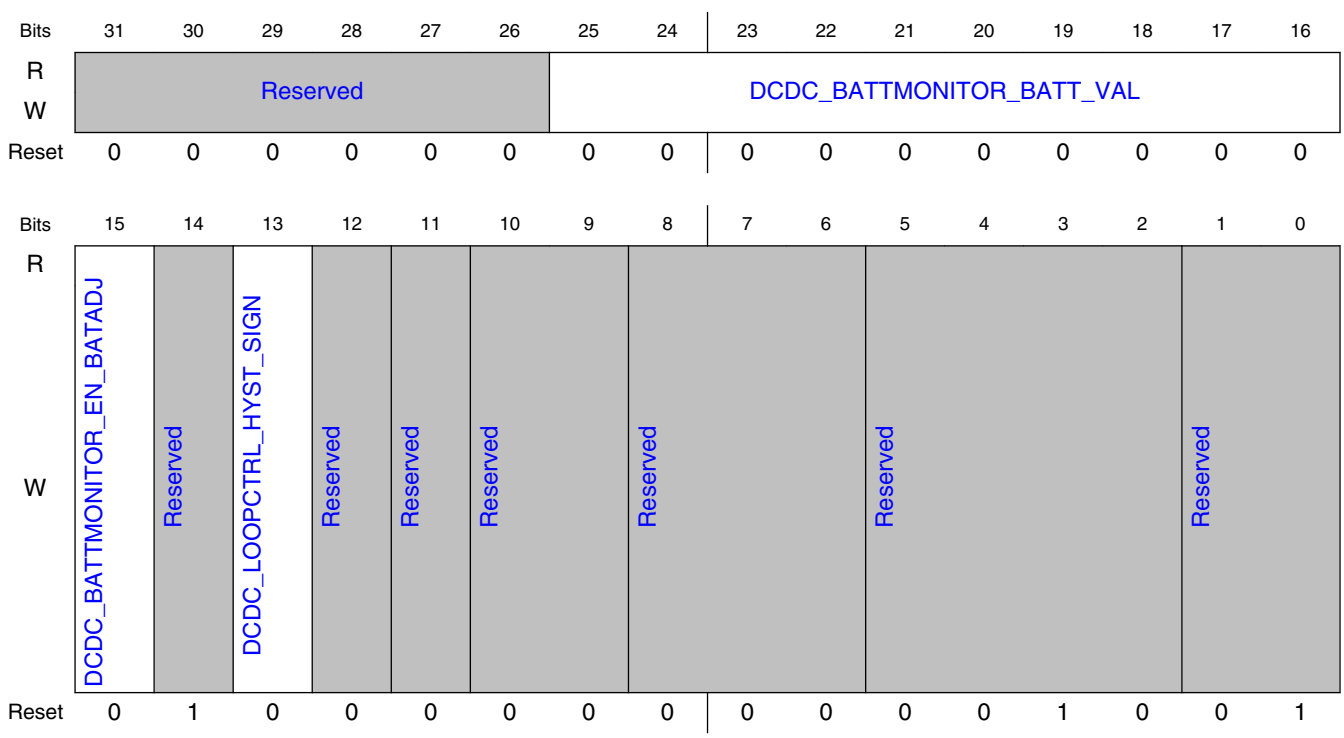
| Register | Offset |
|---|---|
| DCDCC2 | 310h |

### 33.4.1.16.2 Function

**NOTE**

The bits are reset solely on a POR/LVD only event.

### 33.4.1.16.3 Diagram



### 33.4.1.16.4 Fields

| Field | Function |
|---|---|
| 31-26 | Reserved |
| — | Reserved |
| 25-16 | DCDC_BATTMONITOR_BATT_VAL |
| DCDC_BATTMONITOR_BATT_VAL | Software should be configured to place the battery voltage in this register measured with an 8 mV LSB resolution through the ADC. This value is used by the DC-DC converter and must be proper configured before setting EN_BATADJ. |
| 15 | DCDC_BATTMONITOR_EN_BATADJ |
| DCDC_BATTMONITOR_EN_BATADJ | This bit enables the DC-DC to improve efficiency and minimize ripple using the information from the BATT_VAL field. The BATT_VAL contains accurate information before setting EN_BATADJ. |
| 14 | Reserved |
| — | Reserved |
| 13 | DCDC_LOOPCTRL_HYST_SIGN |
| DCDC_LOOPCTRL_HYST_SIGN | Invert the sign of the hysteresis in DC-DC analog comparators. This bit is set when in Pulsed mode. |
| 12 | Reserved |
| — | Reserved |
| 11 | Reserved |
| — | Reserved |
| 10-9 | Reserved |
| — | Reserved |
| 8-6 | Reserved |
| — | Reserved |
| 5-2 | Reserved |
| — | Reserved |
| 1-0 | Reserved |
| — | Reserved |

## 33.4.1.17 DCDC Control Register 3 (DCDCC3)

### 33.4.1.17.1 Offset

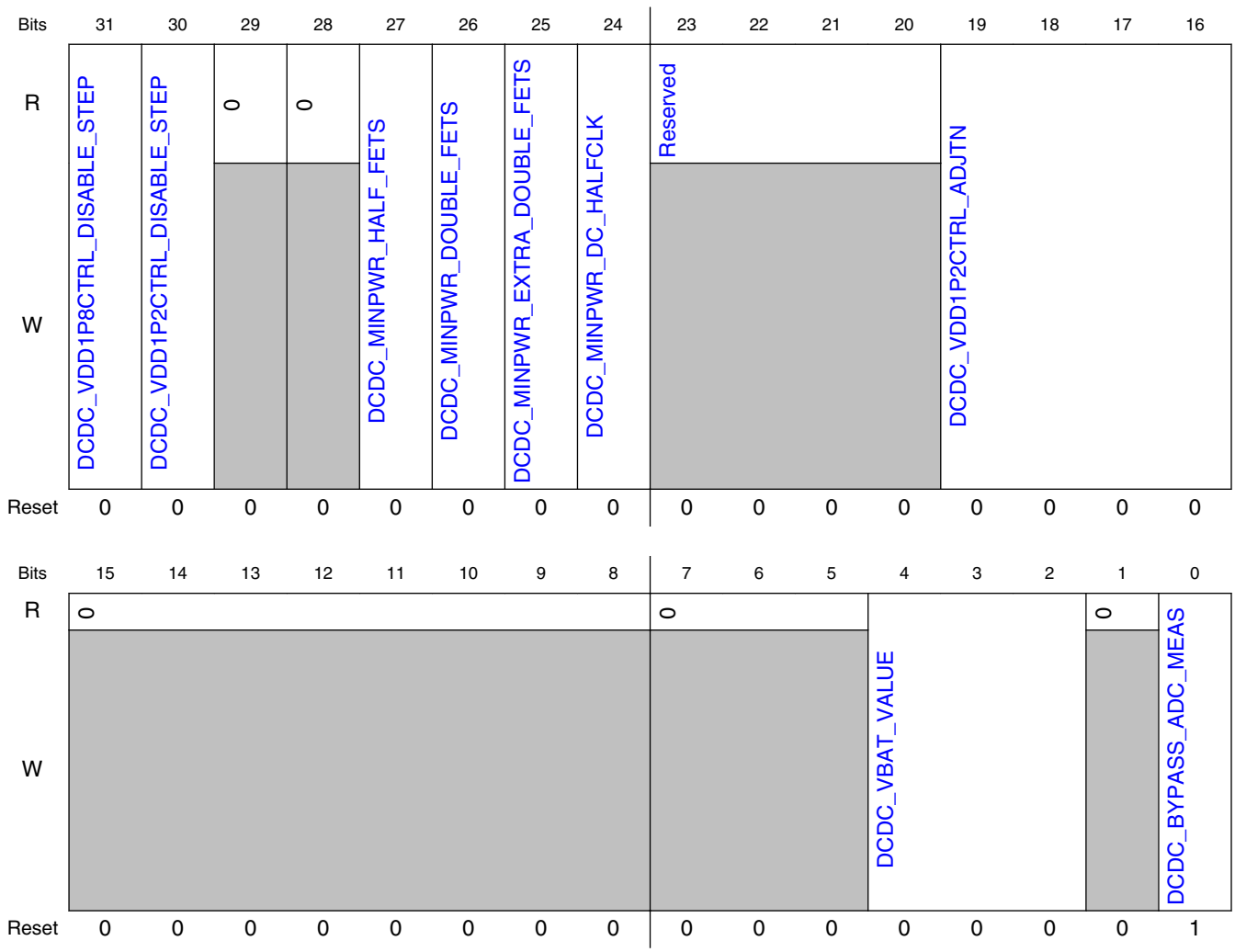
| Register | Offset |
|---|---|
| DCDCC3 | 314h |

### 33.4.1.17.2 Function

**NOTE**

The bits are reset on a POR/LVD event.

### 33.4.1.17.3 Diagram




### 33.4.1.17.4 Fields

| Field | Function |
|---|---|
| 31<br>DCDC_VDD1P8CTRL_DISABLE_STEP | DCDC_VDD1P8CTRL_DISABLE_STEP<br>Disable stepping for VDD1P8. Must set this bit before entering low power modes. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 30<br>DCDC_VDD1P2CTRL_DISABLE_STEP | DCDC_VDD1P2CTRL_DISABLE_STEP<br>Disable stepping for VDD_CORE. Must set this bit before entering low power modes. |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>DCDC_MINPWR_HALF_FETS | DCDC_MINPWR_HALF_FETS<br>Use half switch FET for the continuous mode. |
| 26<br>DCDC_MINPWR_DOUBLE_FETS | DCDC_MINPWR_DOUBLE_FETS<br>Use double switch FET for the continuous mode. |
| 25<br>DCDC_MINPWR_EXTRA_DOUBLE_FETS | DCDC_MINPWR_EXTRA_DOUBLE_FETS<br>Use extra double switch FET for the continuous mode. |
| 24<br>DCDC_MINPWR_DC_HALFCLK | DCDC_MINPWR_DC_HALFCLK<br>Set DCDC clock to half frequency for the continuous mode. |
| 23-20<br>— | Reserved |
| 19-16<br>DCDC_VDD1P2CTRL_ADJTN | DCDC_VDD1P2CTRL_ADJTN<br>Adjust value of duty cycle when switching between VDD_CORE and VDD1P8.<br>To calculate the adjusted value use following formula:<br>[ (DCDC_1p8 Target Value/DCDC_IN) – (DCDC_1p2 Target Value/DCDC_IN) ] *32. |
| 15-8<br>— | Reserved |
| 7-5<br>— | Reserved |
| 4-2<br>DCDC_VBAT_VALUE | DCDC_VBAT_VALUE<br>DCDC VBAT values that are used by DCDC when DCDC_BYPASS_ADC_MEAS is set. |
| 1<br>— | Reserved |
| 0<br>DCDC_BYPASS_ADC_MEAS | DCDC_BYPASS_ADC_MEAS<br>Forces DCDC to bypass the ADC measuring state and loads the DCDCC4[INTEGRATOR_VALUE] value if DCDCC4[INTEGRATOR_VALUE_SELECT] is set. |

## 33.4.1.18 DCDC Control Register 4 (DCDCC4)

### 33.4.1.18.1 Offset

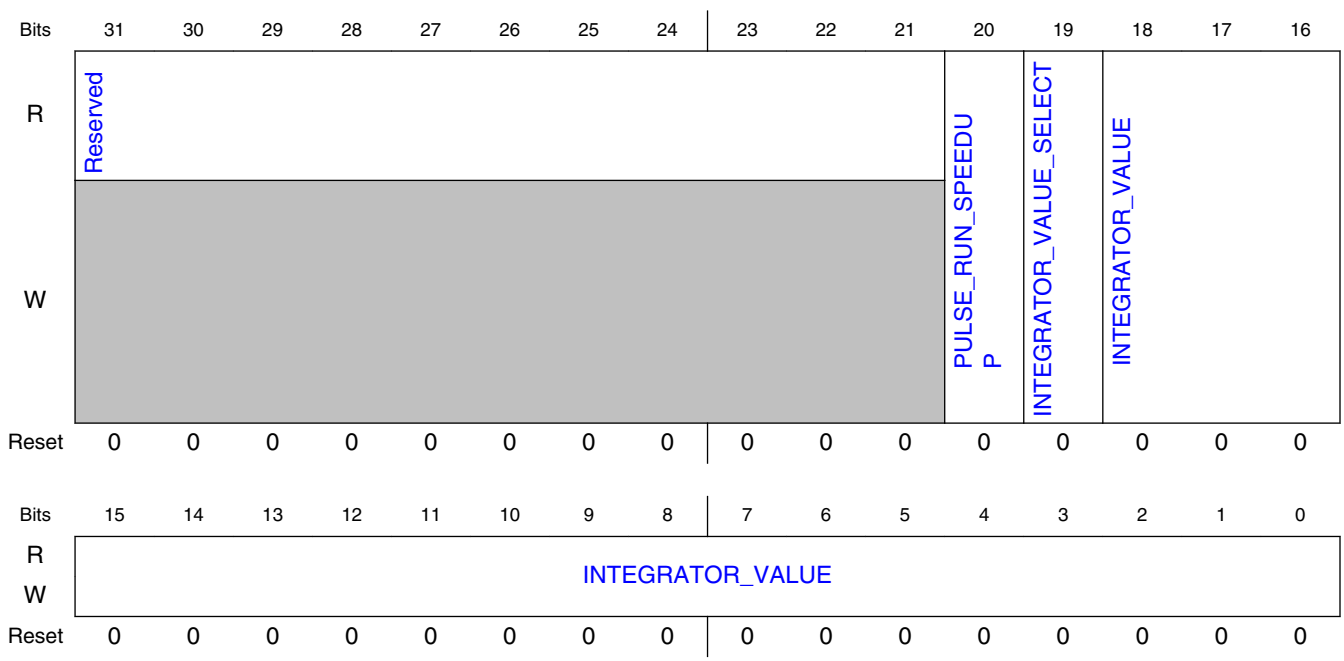
| Register | Offset |
|---|---|
| DCDCC4 | 318h |

### 33.4.1.18.2 Function

**NOTE**

The bits are reset on a POR/LVD event.

### 33.4.1.18.3 Diagram




### 33.4.1.18.4 Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 20 | PULSE RUN SPEEDUP |

*Table continues on the next page...*

| Field | Function |
|---|---|
| PULSE_RUN_SPEEDUP | Enable pulse run speedup. Before setting this bit, INTEGRATOR_VALUE_SEL must be set to 1'b1 and integrator value must be programmed. |
| 19<br>INTEGRATOR_VALUE_SELECT | INTEGRATOR VALUE SELECT<br>Selects the integrator value from INTEGRATOR_VALUE register or saved hardware value.<br>0b - Select the saved value in hardware<br>1b - Select the integrator value in this register |
| 18-0<br>INTEGRATOR_VALUE | INTEGRATOR VALUE<br>Integrator value which can be loaded in pulsed mode. Software can program this value according to battery voltage and VDD_CORE output target value before goes to the pulsed mode. It is signed number. The register value = (Dutycycle * 32 - 16) * 2 ^ 13<br>dutycycle = VDD_CORE/Vbat |

## 33.4.1.19 DCDC Control Register 6 (DCDCC6)

### 33.4.1.19.1 Offset

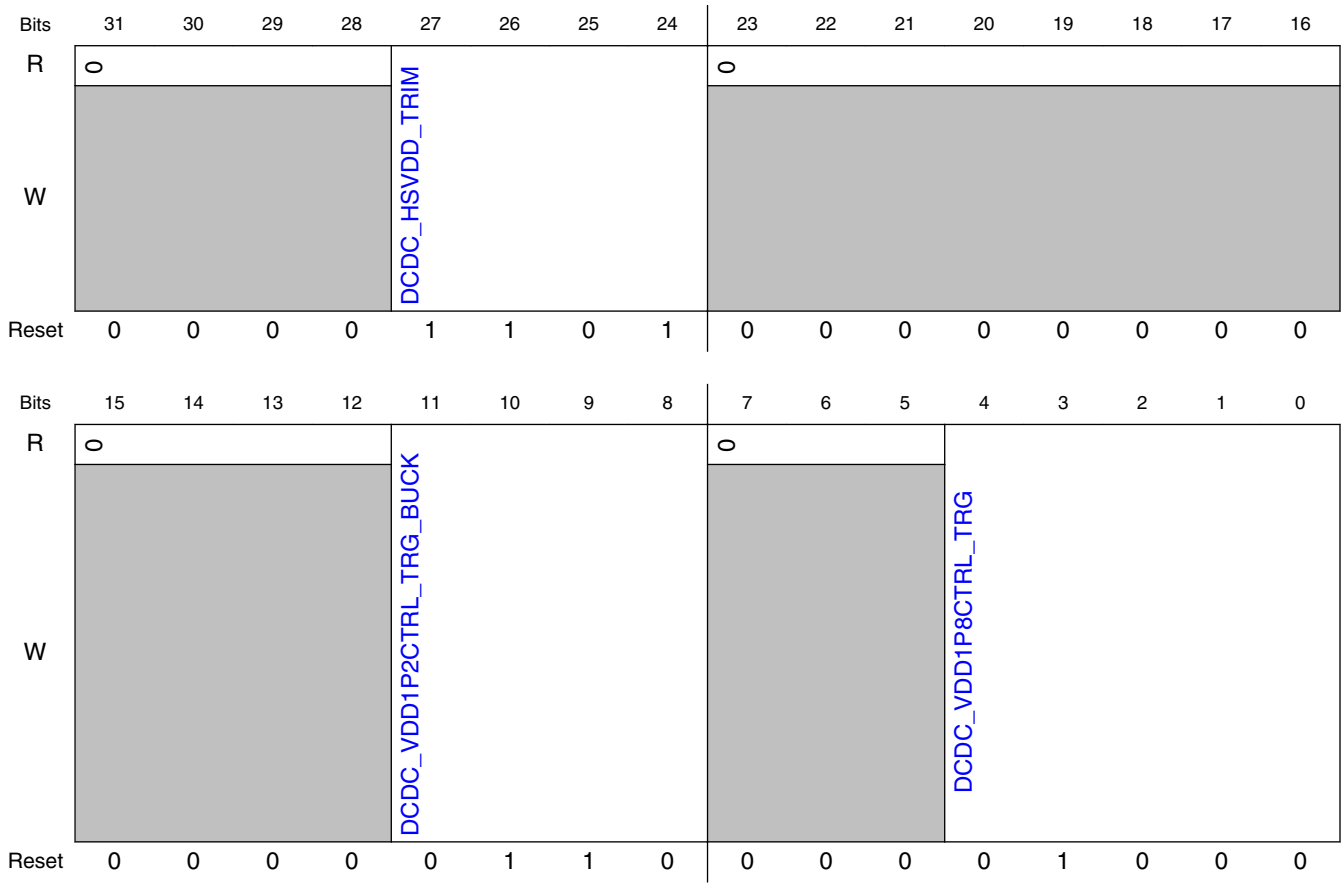
| Register | Offset |
|---|---|
| DCDCC6 | 320h |

### 33.4.1.19.2 Function

**NOTE**

The bits are reset on a POR/LVD event.

### 33.4.1.19.3 Diagram




### 33.4.1.19.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 27-24<br>DCDC_HSVDD_TRIM | DCDC_HSVDD_TRIM<br>Target value of VDD_CORE in buck HRUN mode, 25 mV each step from 0x0D to 0x0F. This value is automatically selected on entry into HSRUN. On exit from HSRUN, DCDC VDD_CORE trim values will default back to values set by DCDC_VDD1P2CTRL_TRG_BUCK register.<br>Valid configurations are tabled below: |

**Table 33-5. DCDC Trim Values**

| Trim Code (Decimal) | Trim Target Value | Step Size |
|---|---|---|
| 13 | 1.400 | 0.025 |
| 14 | 1.425 | 0.025 |
| 15 | 1.450 | 0.025 |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>23-12<br>—</td><td>Reserved</td></tr>
<tr><td>11-8<br>DCDC_VDD1P2CTRL_TRG_BUCK</td><td>DCDC_VDD1P2CTRL_TRG_BUCK<br>Target value of VDD_CORE in buck mode, 25 mV each step from 0x06 to 0x0A<br>Valid configurations are tabled below:<br>Table 33-6. DCDC Trim Values<table><tr><th>Trim Code (Decimal)</th><th>Trim Target Value</th><th>Step Size</th></tr><tr><td>6</td><td>1.225</td><td>0.025</td></tr><tr><td>7</td><td>1.250</td><td>0.025</td></tr><tr><td>8</td><td>1.275</td><td>0.025</td></tr><tr><td>9</td><td>1.300</td><td>0.025</td></tr><tr><td>10</td><td>1.325</td><td>0.025</td></tr></table></td></tr>
<tr><td>7-5<br>—</td><td>Reserved</td></tr>
<tr><td>4-0<br>DCDC_VDD1P8CTRL_TRG</td><td>DCDC_VDD1P8CTRL_TRG<br>Target value of VDD1P8.<br>The default target voltage of the VDD1P8 DCDC output is 1.5V. When switching from the LDO to the DCDC, the LDO voltage should match that of the DCDC voltage. The RFLDOSC[IOREGVSEL] target voltage should match that of the DCDC_VDD1P8CTRL_TRG before enabling the DCDC regulator.<br>• If RFLDOSC[IOREGVSEL] = 0 then minimum DCDC_VDD1P8CTRL_TRG = 8<br>• If RFLDOSC[IOREGVSEL] = 1 then minimum DCDC_VDD1P8CTRL_TRG = 20<br>NOTE: The bits are reset solely on a POR/LVD only event.<br>Valid configurations are tabled below:<br>Table 33-7. DCDC Trim Values<table><tr><th>Trim Code (Decimal)</th><th>Trim Target Value</th><th>Step Size</th></tr><tr><td>8</td><td>1.500</td><td>0.025</td></tr><tr><td>9</td><td>1.525</td><td>0.025</td></tr><tr><td>10</td><td>1.550</td><td>0.025</td></tr><tr><td>11</td><td>1.575</td><td>0.025</td></tr><tr><td>12</td><td>1.600</td><td>0.025</td></tr><tr><td>13</td><td>1.625</td><td>0.025</td></tr><tr><td>14</td><td>1.650</td><td>0.025</td></tr><tr><td>15</td><td>1.675</td><td>0.025</td></tr><tr><td>16</td><td>1.700</td><td>0.025</td></tr><tr><td>17</td><td>1.725</td><td>0.025</td></tr><tr><td>18</td><td>1.750</td><td>0.025</td></tr><tr><td>19</td><td>1.775</td><td>0.025</td></tr><tr><td>20</td><td>1.800</td><td>0.025</td></tr></table></td></tr>
</table>

| Field | Function |
|---|---|
| | **Table 33-7. DCDC Trim Values (continued)** |

| | | |
|---|---|---|
| 21 | 1.825 | 0.025 |
| 22 | 1.850 | 0.025 |
| 23 | 1.875 | 0.025 |
| 24 | 1.900 | 0.025 |
| 25 | 1.925 | 0.025 |
| 26 | 1.950 | 0.025 |
| 27 | 1.975 | 0.025 |
| 28 | 2.000 | 0.025 |
| 29 | 2.025 | 0.025 |
| 30 | 2.050 | 0.025 |
| 31 | 2.075 | 0.025 |

## 33.4.1.20 LP Request Pin Control Register (LPREQPINCNTRL)

### 33.4.1.20.1 Offset

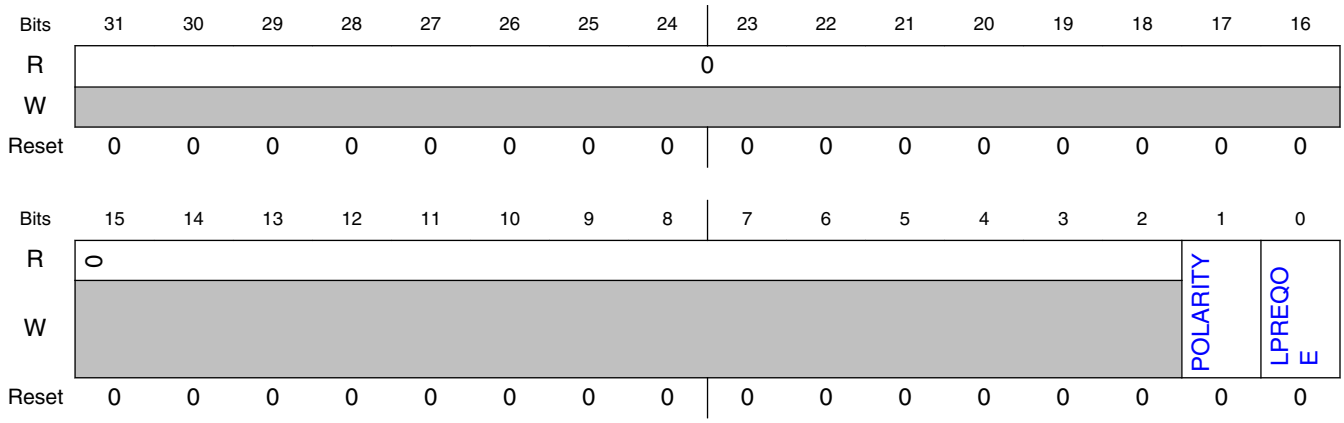
| Register | Offset |
|---|---|
| LPREQPINCNTRL | 40Ch |

### 33.4.1.20.2 Function

**NOTE**

The bits are reset solely on a POR/LVD only event.

### 33.4.1.20.3 Diagram




### 33.4.1.20.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>POLARITY | Low Power Request Output Pin Polarity Control Register<br>Controls the polarity of the Low Power Request output pin.<br>0b - High true polarity.<br>1b - Low true polarity. |
| 0<br>LPREQOE | Low Power Request Output Enable Register<br>0b - Low Power request output pin not enabled.<br>1b - Low Power request output pin enabled. |

## 33.5 Functional Description

### 33.5.1 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions. This protects memory contents and controls MCU system states during supply voltage variations.

The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage: high ($V_{LVDH}$) or low ($V_{LVDL}$). The trip voltage is selected by LVDSC1[LVDV].Whether or not the LVD system is on in low power modes depends on MCU and RF arbitration and configurations of CORELPCNFG.

Two flags are available to indicate the status of the low-voltage detect system:

- The Low Voltage Detect Flag in the Low Voltage Status and Control 1 Register (LVDSC1[LVDF]) operates in a level sensitive manner. LVDSC1[LVDF] is set when the supply voltage falls below the selected trip point (VLVD). LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC1[LVDF] remains set.
- The Low Voltage Warning Flag (LVWF) in the Low Voltage Status and Control 2 Register (LVDSC2[LVWF]) operates in a level sensitive manner. LVDSC2[LVWF] is set when the supply voltage falls below the selected monitor trip point (VLVW). LVDSC2[LVWF] is cleared by writing one to LVDSC2[LVWACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC2[LVWF] remains set.

#### 33.5.1.1 LVD reset operation

By setting LVDSC1[LVDRE], the LVD generates a reset upon detection of a low-voltage condition. The low-voltage detection threshold is determined by LVDSC1[LVDV]. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. See also LVD reset settings in either the Reset Control Module or the Core Mode Control chapter for more information.

#### 33.5.1.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDSC1[LVDIE] set and LVDSC1[LVDRE] clear), LVDSC1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK].

#### 33.5.1.3 Low-voltage warning (LVW) interrupt operation

The LVD system contains a Low-Voltage Warning Flag (LVWF) in the Low Voltage Detect Status and Control 2 Register to indicate that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt, which is enabled by setting LVDSC2[LVWIE]. If enabled, an LVW interrupt request occurs when LVDSC2[LVWF] is set. LVDSC2[LVWF] is cleared by writing 1 to LVDSC2[LVWACK].

LVDSC2[LVWV] selects one of the following trip voltages:

- Highest: $V_{LVWH}$
- Lowest: $V_{LVWL}$

## 33.5.2 High-voltage detect (HVD) system

This device includes a system to guard against high-voltage conditions.

The system is comprised of a HVD circuit with a user-selectable trip voltage: high ($V_{HVDH}$) or low ($V_{HVDL}$). The trip voltage is selected by HVDSC1[HVDV]. Whether or not the HVD system is on in low power modes depends on MCU and RF arbitration and configurations of CORELPCNFG.

A flag is available to indicate the status of the high-voltage detect system:

- The High Voltage Detect Flag in the High Voltage Status and Control 1 Register (HVDSC1[HVDF]) operates in a level sensitive manner. HVDSC1[HVDF] is set when the supply voltage rises above the selected trip point (VHVD). HVDSC1[HVDF] is cleared by writing 1 to HVDSC1[HVDACK], but only if the internal supply has returned below the trip point; otherwise, HVDSC1[LVDF] remains set.

### 33.5.2.1 HVD reset operation

By setting HVDSC1[HVDRE], the HVD generates a reset upon detection of a high-voltage condition. The high-voltage detection threshold is determined by HVDSC1[HVDV]. After an HVD reset occurs, the HVD system holds the MCU in reset until the supply voltage falls below this threshold. See also HVD reset settings in either the Reset Control Module or the Core Mode Control chapter for more information.

### 33.5.2.2 HVD interrupt operation

By configuring the HVD circuit for interrupt operation (HVDSC1[HVDIE] set and HVDSC1[HVDRE] clear), HVDSC1[HVDF] is set and an HVD interrupt request occurs upon detection of a high voltage condition. HVDSC1[HVDF] is cleared by writing 1 to HVDSC1[HVDACK].

### 33.5.3 Stop and VLPx Modes

The CORELPCNFG[LPSEL] and CORELPCNFG[BGEN] bits provide a means to optimize the device current consumption depending on the specific device use case when STOP or VLPx modes are being used.

The CORELPCNFG[LPSEL] bit controls the power state of the core LDO when in low power modes. The current that the core LDO can provide in the low power state is reduced but is still capable of providing significantly more current than most applications would require. Refer to the device data sheet for the maximum low power state current. It is recommended that the CORELPCNFG[LPSEL] bit remain cleared in both STOP and VLPx modes unless it is expected the core current consumption will exceed the maximum allowed low power state current. If CORELPCNFG[LPSEL] bit is to be set to one the CORELPCNFG[BGEN] bit must also be set to one.

If the core LDO is configured to be in low power state in STOP and VLPx modes, the CORELPCNFG[BGEN] bit can be cleared to zero. If the bandgap is required when in these modes, then the CORELPCNFG[BGEN] should be set to one. This will increase the current consumption so this bit should only be set when the bandgap is required to be on.

If it is required to keep the low voltage detect operating in STOP or VLPx modes, then the CORELPCNFG[BGEN] bit must also be set along with the CORELPCNFG[LVDEN] bit.

### 33.5.4 I/O retention

When in LLS mode, the I/O pins are held in their input or output state.

Upon wakeup, regulators selected by RCTRL[REGSEL] are re-enabled, performs a power-up sequence to full regulation, and releases the logic from state retention mode. The I/O are released immediately after a wake-up or reset event. In the case of LLS exit via a RESET pin, the I/O default to their reset state.

When in VLLS modes, the I/O states are held on a wake-up event (with the exception of wake-up by reset event) until the wake-up has been acknowledged via a write to REGSC[ACKISO]. In the case of VLLS exit via a RESET pin, the I/O are released and default to their reset state. In this case, no write to REGSC[ACKISO] is needed.

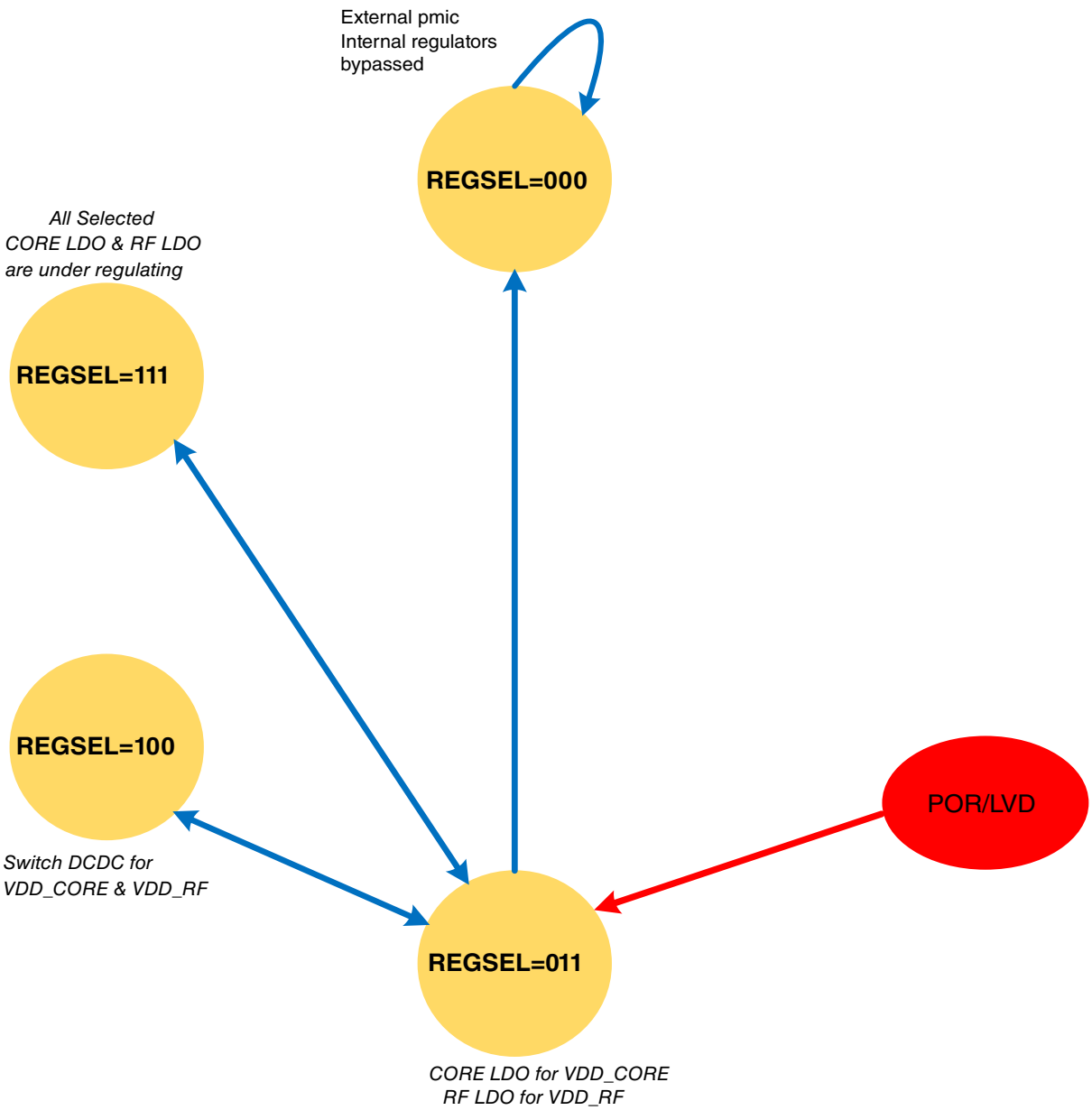
### 33.5.5 Regulators

The SPM has three regulators (CORE LDO Regulator, RF LDO Regulator, and DCDC) that supply power to the MCU including Cores, Memory Arrays, and On-Chip Radio.

Each of these regulators can be configured in High Power (HP) mode, Low Power (LP) mode, or Off mode. There is one exception - DCDC does not support Low Power mode. On Power-up, MCU will be come up on the Core LDO and RF LDO regulators with DCDC Off and power gated. External PMIC will also be supported that will allow bypass of the internal regulators and power-up on the PMIC. In RUN and HSRUN modes the RCTRL[REGSEL] selects the regulator or regulators that are enabled. In LP entry modes, the SPM will switch to the regulators selected by the LPCNTRL[REGSEL] register.

Below diagram illustrates valid regulator configurations:

# VALID REGULATOR CONFIGURATIONS




LEGEND:

REGSEL[0] = CORE LDO Select
REGSEL[1] = RF LDO Select
REGSEL[2] = DCDC Select

**Figure 33-2. VALID REGULATOR CONFIGURATIONS**

Transitions between the regulators are initiated either by the User programming a different regulator configuration in the RCTRL[REGSEL] when MCU is in RUN mode or by power mode request initiated by the Multi System Mode Controller (MSMC) or by the Radio. SPM will arbitrate between these two power mode request sources and service the first request it detects. Pending power mode request will get serviced after first is complete. When MSMC and SPM are requesting different power modes, SPM will transition to the higher power regulation selected between the MSMC/MCU and the Radio. (ex. MSMC sequences the SPM into VLPS with CORE LDO and RF LDO low power regulators selected. If Radio asserts a stop wake up request, SPM will sequence out of low power mode and switch to the regulator selected by the RCTRL[REGSEL] register. Note: In these cases the SPM will retain power gating to the CORE even though the Radio has requested and moved to RUN mode). Power mode transitions are not aborted but allowed to complete before SPM services the pending power mode request. So if MSMC starts a low power mode transition, a low power wake up request from Radio will not be serviced until SPM has fully completed and acknowledged the MSMC power mode request.

#### 33.5.5.1 DCDC

The DCDC converter module is a switching mode DC-DC converter supporting Buck and Bypass mode. It requires a single inductor to produce multiple switching outputs. The converter can be operated in continuous. This mode of operation is software configurable. An internal oscillator is included for use case where external oscillator is not available. The DCDC converter produces two switching outputs in Buck mode. They are VDD_CORE and VDD1p8, which both can produce up to 100mA continuous output current respectively. Typical switching frequency is 2MHz. DCDC conversion efficiency is typically 90%. In Buck mode, it supports operation with battery voltage from 2.1V to 4.25V. In Bypass mode, the DCDC converter is disabled. Individual supply signals need to be supplied with regulated supply accordingly.

DCDC by default is configured in off (Bypass) mode and not selected as the system regulator. To enable DCDC, the RCTRL[REGSEL[2]] needs to be set. When enabling the DCDC, it must be done while MCU is configured in RUN mode.

To switch to DCDC the following procedures are recommended:

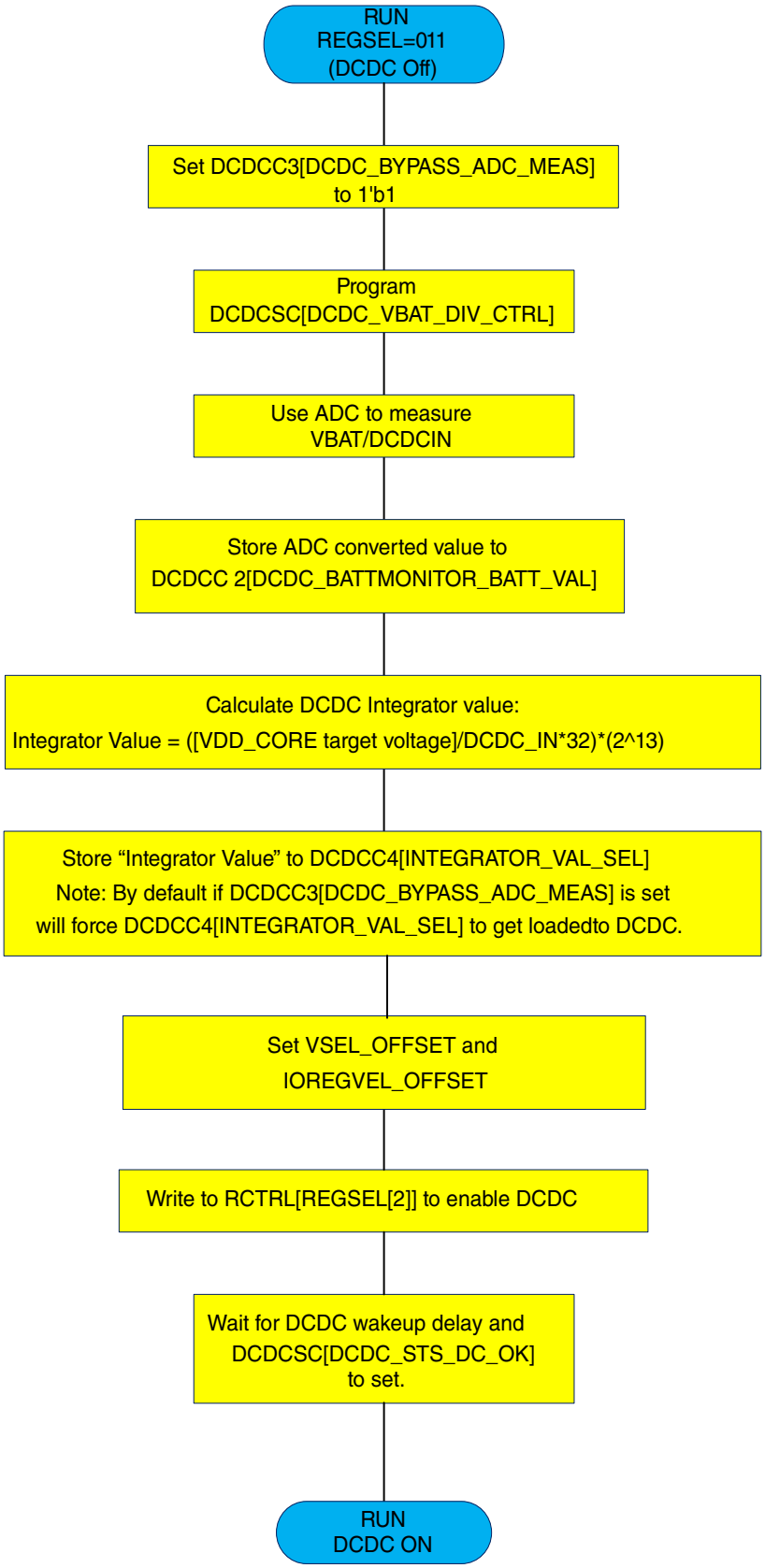



**Figure 33-3. RUN TO DCDC PROCEDURE**

#### 33.5.5.1.1 HSRUN

When entering HSRUN or exiting HSRUN with DCDC enabled the SPM automatically switched the VDD_CORE target value. On HSRUN entry SPM will set the VDD_CORE target value to one set in the DCDCC6[DCDC_HSVDD_TRIM] register. On HSRUN exit the SPM will set the VDD_CORE target value to one set in the DCDCC6[DCDC_VDD1P2CTRL_TRG_BUCK] register.

#### 33.5.5.1.2 Low Power Mode

When entering any of the low power modes, DCDC will get disabled at the end of low power entry. On exit from low power modes, SPM will automatically force DCDC disable step and enable DCDC if RCTRL[REGSEL[2]] was set prior to entering the low power mode. Hardware will clear DCDC disable step at end of DCDC low power wakeup delay. If in VLPR the RCTRL[REGSEL] are changed to switch off DCDC, on exit from VLPR, SPM will switch first to RUN with DCDC selected first and once in RUN mode will switch to the new RCTRL[REGSEL] settings as SPM will not support changing the selected RUN regulator while in a low power mode.

# Chapter 34
# Low-Leakage Wakeup Unit (LLWU)

## 34.1 Chip-specific LLWU information

### 34.1.1 Wake-up sources for Cortex-M4 core (CPU0)

The device uses the following inputs as wakeup sources to the LLWU0 module. The LLWU0_Px inputs are connections to external physical pins. The LLWU0_MxIF inputs are connections to the internal peripheral interrupt flags from different modules that can operate in low-power modes. The LLWU0_MxDR inputs are connections to the internal peripheral Asynchronous DMA or module Triggers from different modules that can operate in low-power modes.

**NOTE**

In addition to the LLWU0 wakeup sources, the device also wakes from low power modes when NMI or RESET pins are enabled and the respective pin is asserted.

**Table 34-1. Wakeup Sources for LLWU0 inputs**

| LLWU0 pin | Module source or pin name |
|---|---|
| LLWU_P0 | PTA1 |
| LLWU_P1 | PTA2 |
| LLWU_P2 | PTA22 |
| LLWU_P3 | PTA30 |
| LLWU_P4 | PTB1 |
| LLWU_P5 | PTB2 |
| LLWU_P6 | PTB4 |
| LLWU_P7 | PTB6 |
| LLWU_P8 | PTB7 |
| LLWU_P9 | PTB8 |

*Table continues on the next page...*

## Table 34-1. Wakeup Sources for LLWU0 inputs (continued)

| LLWU0 pin | Module source or pin name |
|---|---|
| LLWU_P10 | PTB16 |
| LLWU_P11 | PTB20 |
| LLWU_P12 | PTB22 |
| LLWU_P13 | PTB25 |
| LLWU_P14 | PTB28 |
| LLWU_P15 | PTC7 |
| LLWU_P16 | PTC9 |
| LLWU_P17 | PTC11 |
| LLWU_P18 | PTC12 |
| LLWU_P19 | PTD8 |
| LLWU_P20 | PTD10 |
| LLWU_P21 | PTE1[1] |
| LLWU_P22 | PTE3[1] |
| LLWU_P23 | PTE8[1] |
| LLWU_P24 | PTE9[1] |
| LLWU_P25 | PTE10[1] |
| LLWU_P26 | PTE12[1] |
| LLWU_P27 | Reserved[2] |
| LLWU_P28 | Reserved[3] |
| LLWU_P29 | USB0 VREGIN |
| LLWU_P30 | USB0_DP[4] |
| LLWU_P31 | USB0_DM[4] |
| LLWU_M0IF | LPTMR0, LPTMR1, LPTMR2 (sharing M0IF)[5] |
| LLWU_M1IF | LPCMP0[6] |
| LLWU_M2IF | LPCMP1[7] |
| LLWU_M3IF | Reserved[8] |
| LLWU_M4IF | Reserved[9] |
| LLWU_M5IF | Tamper Detect[10] |
| LLWU_M6IF | RTC Alarm[10] |
| LLWU_M7IF | RTC Seconds[10] |
| LLWU_M0DR | LPTMR0 Asynchronous DMA |
| LLWU_M1DR | LPTMR1 Asynchronous DMA |
| LLWU_M2DR | LPTMR2 Asynchronous DMA |
| LLWU_M3DR | Reserved[11] |
| LLWU_M4DR | LPTMR0 Trigger |
| LLWU_M5DR | LPTMR1 Trigger |
| LLWU_M6DR | LPTMR2 Trigger |
| LLWU_M7DR | Reserved[12] |

1. Set SPM_CORESC[VDDIOOVRIDE] for all PTE pins to wakeup from VLLS0/1
2. The corresponding LLWU0_PE[23: 22], LLWU0_PF[27], LLWU0_PDC2[27], and LLWU0_PMC[27] are reserved.

3. The corresponding LLWU0_PE[25: 24], LLWU0_PF[28], LLWU0_PDC2[28], and LLWU0_PMC[28] are reserved.
4. Set SPM_CORESC[USBOVRIDE] to wakeup from VLLS0 or VLLS1.
5. Requires the peripheral and the peripheral interrupt to be enabled. The LLWU's WUME bit enables the internal module flag as a wakeup input. After wakeup, the flags are cleared based on the peripheral clearing mechanism.
6. Set SPM_CORELPCNFG[ALLREFEN] to wakeup from VLLS0/1.
7. Set SPM_CORESC[VDDIOOVRIDE] and SPM_CORELPCNFG[ALLREFEN] to wakeup from VLLS0/1.
8. The corresponding LLWU0_ME[3] is reserved.
9. The corresponding LLWU0_ME[4] is reserved.
10. Set SPM_CORESC[VDDIOOVRIDE] to wakeup from VLLS0 or VLLS1.
11. The corresponding LLWU0_DE[3] is reserved.
12. The corresponding LLWU0_DE[7] is reserved.

### 34.1.2 Wake-up sources for Cortex-M0+ core (CPU1)

The device uses the following inputs as wakeup sources to the LLWU1 module. The LLWU1_Px inputs are connections to external physical pins. The LLWU1_MxIF inputs are connections to the internal peripheral interrupt flags from different modules that can operate in low-power modes. The LLWU1_MxDR inputs are connections to the internal peripheral Asynchronous DMA or module Triggers from different modules that can operate in low-power modes.

**NOTE**

In addition to the LLWU1 wakeup sources, the device also wakes from low power modes when NMI or RESET pins are enabled and the respective pin is asserted.

**Table 34-2. Wakeup Sources for LLWU1 inputs**

| LLWU1 pin | Module source or pin name |
|---|---|
| LLWU_P0 | PTA1 |
| LLWU_P1 | PTA2 |
| LLWU_P2 | PTA22 |
| LLWU_P3 | PTA30 |
| LLWU_P4 | PTB1 |
| LLWU_P5 | PTB2 |
| LLWU_P6 | PTB4 |
| LLWU_P7 | PTB6 |
| LLWU_P8 | PTB7 |
| LLWU_P9 | PTB8 |
| LLWU_P10 | PTB16 |
| LLWU_P11 | PTB20 |
| LLWU_P12 | PTB22 |
| LLWU_P13 | PTB25 |
| LLWU_P14 | PTB28 |
| LLWU_P15 | PTC7 |

*Table continues on the next page...*

## Table 34-2. Wakeup Sources for LLWU1 inputs (continued)

| LLWU1 pin | Module source or pin name |
|---|---|
| LLWU_P16 | PTC9 |
| LLWU_P17 | PTC11 |
| LLWU_P18 | PTC12 |
| LLWU_P19 | PTD8 |
| LLWU_P20 | PTD10 |
| LLWU_P21 | PTE1 |
| LLWU_P22 | PTE3 |
| LLWU_P23 | PTE8 |
| LLWU_P24 | PTE9 |
| LLWU_P25 | PTE10 |
| LLWU_P26 | PTE12 |
| LLWU_P27 | Reserved[1] |
| LLWU_P28 | Reserved[2] |
| LLWU_P29 | USB0 VREGIN |
| LLWU_P30 | USB0_DP[3] |
| LLWU_P31 | USB0_DM[3] |
| LLWU_M0IF | LPTMR0, LPTMR1, LPTMR2 (sharing M0IF)[4] |
| LLWU_M1IF | LPCMP0 |
| LLWU_M2IF | LPCMP1 |
| LLWU_M3IF | Wake-From-Radio (WFR) |
| LLWU_M4IF | Reserved[5] |
| LLWU_M5IF | Tamper Detect[6] |
| LLWU_M6IF | RTC Alarm[6] |
| LLWU_M7IF | RTC Seconds[6] |
| LLWU_M0DR | LPTMR0 Asynchronous DMA |
| LLWU_M1DR | LPTMR1 Asynchronous DMA |
| LLWU_M2DR | LPTMR2 Asynchronous DMA |
| LLWU_M3DR | Reserved[7] |
| LLWU_M4DR | LPTMR0 Trigger |
| LLWU_M5DR | LPTMR1 Trigger |
| LLWU_M6DR | LPTMR2 Trigger |
| LLWU_M7DR | Reserved[8] |

1. The corresponding LLWU1_PE[23:22], LLWU1_PF[27], LLWU1_PDC2[27], and LLWU1_PMC[27] are reserved.
2. The corresponding LLWU1_PE[25:24], LLWU1_PF[28], LLWU1_PDC2[28], and LLWU1_PMC[28] are reserved.
3. Set SPM_CORESC[USBOVRIDE] to wakeup from VLLS0 or 1.
4. Requires the peripheral and the peripheral interrupt to be enabled. The LLWU's WUME bit enables the internal module flag as a wakeup input. After wakeup, the flags are cleared based on the peripheral clearing mechanism.
5. The corresponding LLWU1_ME[4] is reserved.
6. Set SPM_CORESC[VDDIOOVRIDE] to wakeup from VLLS0 or VLLS1.
7. The corresponding LLWU1_DE[3] is reserved.
8. The corresponding LLWU1_DE[7] is reserved.

## 34.2 Introduction

The LLWU module allows the user to select external pins and internal modules as interrupt wake-up sources from LLS/VLLS power modes. It also allows pins and/or modules to generate a temporary wake-up from LLS to allow servicing of DMA or trigger events. Digital filtering is also available for the external wakeup pins.

The RESET_B pin is an additional source for triggering an exit from low-leakage power modes, and causes the MCU to exit both LLS and VLLS through a reset flow. Input sources to the LLWU are described in the device's chip configuration details.

See AN4503: Power Management for Kinetis MCUs for further details on using the LLWU.

### 34.2.1 Features

The LLWU module features include:

- Support for up to 32 external input pins and up to 8 internal modules with individual enable bits for MCU interrupt from low leakage modes.
- Input sources may be external pins or from internal modules capable of running in LLS or VLLS. See the chip configuration information for wakeup input sources for this device.
- Support to configure internal modules or external pins as DMA or trigger sources for temporary wakeup from LLS.
- External input pins programmable for falling-edge, rising-edge, or any-edge detection.
- Support to enable external input pin and filter detection in all power modes (not only LLS/VLLS).
- Optional digital filters provided to qualify an external pin detect. Note that when the LPO clock is disabled, filters are disabled and bypassed.

### 34.2.2 Modes of operation

The LLWU module becomes functional on entry into a low-leakage power mode. After recovery from LLS, the LLWU is immediately disabled. After recovery from VLLS, the LLWU continues to detect wake-up events until the user has reinitialized the system and deasserted isolation. Detection of external pin/filter events can also optionally remain enabled in all power modes using the LLWU_PMC/LLWU_FMC registers.

#### 34.2.2.1 LLS mode

Wake-up events due to external pin inputs (LLWU_Px) and internal modules (LLWU_MxIF) configured as interrupts, result in an interrupt flow when exiting LLS. Exit due to a pin interrupt will trigger the LLWU interrupt service routine, but exit due to module interrupt will trigger only that module's interrupt service routine. Wake-up events due to internal module requests (LLWU_MxDF) and external pin inputs configured as DMA/trigger events, cause the system to temporarily exit LLS with the CPU clock disabled, allowing the DMA/trigger request to be serviced. When the DMA/trigger event is complete, the system will re-enter LLS.

**NOTE**

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully recover from LLS after an external pin event.

#### 34.2.2.2 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow. Exit due to a pin interrupt will also trigger the LLWU interrupt service routine, but exit due to module interrupt will trigger only that module's interrupt service routine.

#### 34.2.2.3 Non-low leakage modes

The LLWU is inactive in all non-low leakage modes, with the exception of external pin/filter detection which can explicitly be kept enabled during all power modes using the LLLWU_PMC/LLWU_FMC registers. The LLWU registers are accessible in non-low leakage modes and are available for configuring and reading status when bus transactions are possible.

When an external wake-up pin filter is first enabled in the LLWU_FILT register, filter operation begins immediately. To allow the filter to initialize, the user should wait at least five LPO clock cycles before entering a low leakage mode or before enabling filter detection for all power modes in the LLWU_FMC register.

### 34.2.3 Block diagram

The following figure is the block diagram for the LLWU module.

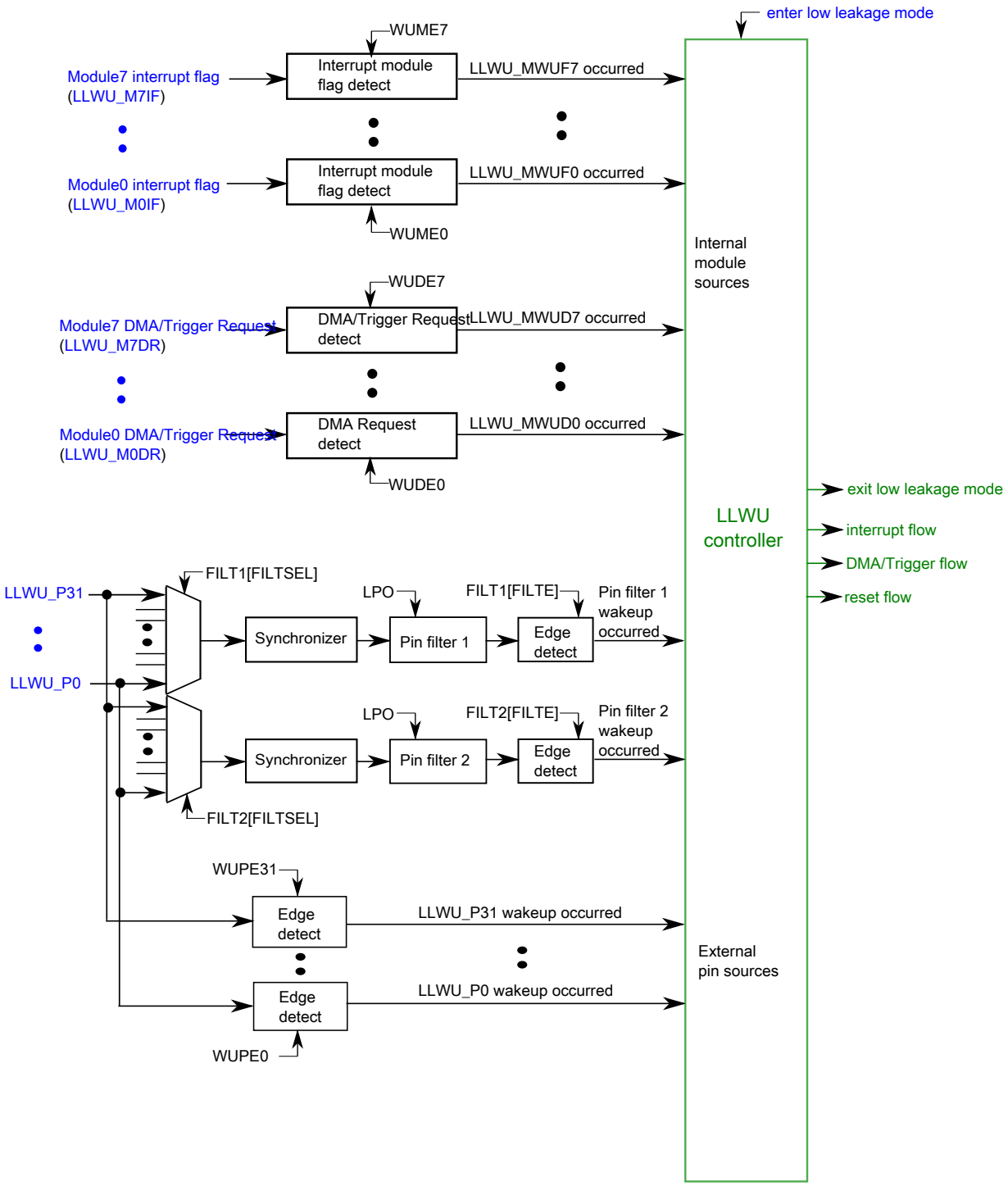



**Figure 34-1. LLWU block diagram**

## 34.3 LLWU signal descriptions

The signal properties of LLWU are shown in the table found here.

The external wakeup input pins can be enabled to detect either rising-edge, falling-edge, or on any change.

**Table 34-3. LLWU signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| LLWU_Pn | External Pin Wakeup inputs (n = 0-31) | I |
| LLWU_MnIF | Internal Module Interrupt flag (n = 0-7) | I |
| LLWU_MnDR | Internal Module DMA/trigger request (n = 0-7) | I |

## 34.4 Memory map/register definition

Information about LLWU registers can be found here.

### 34.4.1 LLWU0 register descriptions

**NOTE**

The LLWU registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

Registers for Pin/Module sources marked "Reserved" in the LLWU chip configuration information will not be writable and will return 0 when read.

All LLWU registers are reset by Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS.

#### 34.4.1.1 LLWU0 Memory map

LLWU0 base address: 4002_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0001h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Parameter Register (PARAM) | 32 | RO | 2020_0702h |
| 8h | Pin Enable 1 register (PE1) | 32 | RW | 0000_0000h |
| Ch | Pin Enable 2 register (PE2) | 32 | RW | 0000_0000h |
| 18h | Module Interrupt Enable register (ME) | 32 | RW | 0000_0000h |
| 1Ch | Module DMA/Trigger Enable register (DE) | 32 | RW | 0000_0000h |
| 20h | Pin Flag register (PF) | 32 | W1C | 0000_0000h |
| 30h | Pin Filter register (FILT) | 32 | RW | 0000_0000h |
| 38h | Pin DMA/Trigger Configuration 1 register (PDC1) | 32 | RW | 0000_0000h |
| 3Ch | Pin DMA/Trigger Configuration 2 register (PDC2) | 32 | RW | 0000_0000h |
| 48h | Pin Filter DMA/Trigger Configuration register (FDC) | 32 | RW | 0000_0000h |
| 50h | Pin Mode Configuration register (PMC) | 32 | RW | 0000_0000h |
| 58h | Pin Filter Mode Configuration register (FMC) | 32 | RW | 0000_0000h |

## 34.4.1.2 Version ID Register (VERID)

### 34.4.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 34.4.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### 34.4.1.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000000b - Standard features implemented<br>0000000000000001b - Support for DMA/Trigger generation from wakeup pins and filters enabled. Support for external pin/filter detection during all power modes enabled. |

### 34.4.1.3 Parameter Register (PARAM)

#### 34.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 34.4.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINS | | | | | | | | MODULES | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMAS | | | | | | | | FILTERS | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 34.4.1.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>PINS | Pin Number<br>Number of Pin wakeup sources supported |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>MODULES | Module Number<br>Number of Module wakeup sources supported |
| 15-8<br>DMAS | DMA Number<br>Number of DMA wakeup sources supported |
| 7-0<br>FILTERS | Filter Number<br>Number of Pin Filters implemented |

## 34.4.1.4 Pin Enable 1 register (PE1)

### 34.4.1.4.1 Offset

| Register | Offset |
|---|---|
| PE1 | 8h |

### 34.4.1.4.2 Function

The PE1 register contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P15-LLWU_P0.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPE*n* field always reads as 00.

### 34.4.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPE15 | | WUPE14 | | WUPE13 | | WUPE12 | | WUPE11 | | WUPE10 | | WUPE9 | | WUPE8 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPE7 | | WUPE6 | | WUPE5 | | WUPE4 | | WUPE3 | | WUPE2 | | WUPE1 | | WUPE0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPE15<br>29-28: WUPE14<br>27-26: WUPE13<br>25-24: WUPE12<br>23-22: WUPE11<br>21-20: WUPE10<br>19-18: WUPE9<br>17-16: WUPE8<br>15-14: WUPE7<br>13-12: WUPE6<br>11-10: WUPE5<br>9-8: WUPE4<br>7-6: WUPE3<br>5-4: WUPE2<br>3-2: WUPE1<br>1-0: WUPE0 | Wakeup pin enable for LLWU_Pn<br>Enables and configures the edge detection for the wakeup pin.<br>00b - External input pin disabled as wakeup input<br>01b - External input pin enabled with rising edge detection when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - External input pin enabled with falling edge detection when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - External input pin enabled with any change detection when configured as interrupt/DMA request |

## 34.4.1.5 Pin Enable 2 register (PE2)

### 34.4.1.5.1 Offset

| Register | Offset |
|---|---|
| PE2 | Ch |

### 34.4.1.5.2 Function

The PE2 register contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P31-LLWU_P16.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPE*n* field always reads as 00.

### 34.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPE31 | | WUPE30 | | WUPE29 | | 0 | | 0 | | WUPE26 | | WUPE25 | | WUPE24 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPE23 | | WUPE22 | | WUPE21 | | WUPE20 | | WUPE19 | | WUPE18 | | WUPE17 | | WUPE16 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPE31 | Wakeup pin enable for LLWU_Pn |
| 29-28: WUPE30<br>27-26: WUPE29<br>25-24: Reserved28<br>23-22: Reserved27<br>21-20: WUPE26<br>19-18: WUPE25<br>17-16: WUPE24<br>15-14: WUPE23<br>13-12: WUPE22 | Enables and configures the edge detection for the wakeup pin.<br>00b - External input pin disabled as wakeup input<br>01b - External input pin enabled with rising edge detection when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - External input pin enabled with falling edge detection when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - External input pin enabled with any change detection when configured as interrupt/DMA request |

| Field | Function |
|---|---|
| 11-10: WUPE21 | |
| 9-8: WUPE20 | |
| 7-6: WUPE19 | |
| 5-4: WUPE18 | |
| 3-2: WUPE17 | |
| 1-0: WUPE16 | |

## 34.4.1.6 Module Interrupt Enable register (ME)

### 34.4.1.6.1 Offset

| Register | Offset |
|---|---|
| ME | 18h |

### 34.4.1.6.2 Function

The ME register contains the bits to enable an internal module interrupt as a wakeup source for inputs LLWU_M7IF - LLWU_M0IF.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If a module interrupt is not supported on a given device, the corresponding WUME*n* bit always reads as a 0.

#### 34.4.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WUME7 | WUME6 | WUME5 | 0 | 0 | WUME2 | WUME1 | WUME0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 34.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>WUMEn | Wakeup module enable for module n<br>Enables an internal module as a wakeup source input.<br>0b - Internal module flag not used as wakeup source<br>1b - Internal module flag used as wakeup source |

### 34.4.1.7 Module DMA/Trigger Enable register (DE)

#### 34.4.1.7.1 Offset

| Register | Offset |
|---|---|
| DE | 1Ch |

#### 34.4.1.7.2 Function

The DE register contains the bits to enable an internal module DMA/Trigger request as a wakeup source for inputs LLWU_M7DR - LLWU_M0DR.

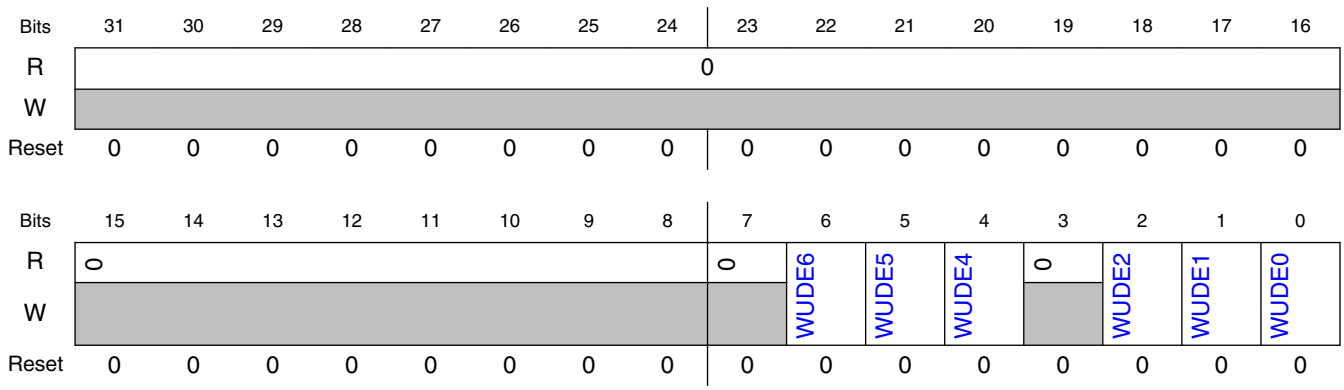
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If a module DMA/Trigger request is not supported on a given device, the corresponding WUDE*n* bit always reads as a 0.

### 34.4.1.7.3 Diagram




### 34.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>WUDEn | DMA/Trigger wakeup enable for module n<br>Enables an internal module as a DMA/Trigger wakeup source.<br>0b - Internal module request not enabled as a DMA/Trigger wakeup source<br>1b - Internal module request enabled as a DMA/Trigger wakeup source |

## 34.4.1.8 Pin Flag register (PF)

### 34.4.1.8.1 Offset

| Register | Offset |
|---|---|
| PF | 20h |

### 34.4.1.8.2 Function

The PF register contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUF*n* bit. The wakeup flag (WUF*n*), if set, will remain set if the associated WUPE*n* bit is cleared.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUF*n* bit always reads as a 0.

### 34.4.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUF31 | WUF30 | WUF29 | 0 | 0 | WUF26 | WUF25 | WUF24 | WUF23 | WUF22 | WUF21 | WUF20 | WUF19 | WUF18 | WUF17 | WUF16 |
| W | W1C | W1C | W1C | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUF15 | WUF14 | WUF13 | WUF12 | WUF11 | WUF10 | WUF9 | WUF8 | WUF7 | WUF6 | WUF5 | WUF4 | WUF3 | WUF2 | WUF1 | WUF0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.1.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>WUFn | Wakeup flag for LLWU_Pn<br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF*n*.<br>0b - LLWU_Pn input was not a wakeup source<br>1b - LLWU_Pn input was a wakeup source |

## 34.4.1.9 Pin Filter register (FILT)

### 34.4.1.9.1 Offset

| Register | Offset |
|---|---|
| FILT | 30h |

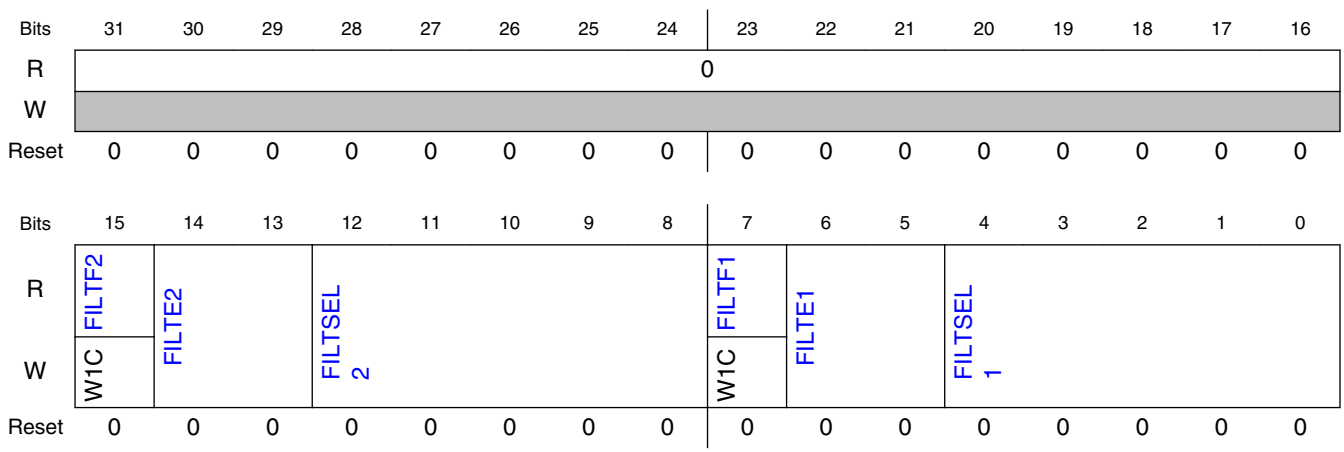
### 34.4.1.9.2 Function

The FILT register is a control and status register that is used to enable/disable the digital filter features for an external pin.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.1.9.3 Diagram




### 34.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>FILTF2 | Filter 2 Flag<br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF.<br>0b - Pin Filter 2 was not a wakeup source<br>1b - Pin Filter 2 was a wakeup source |
| 14-13<br>FILTE2 | Filter 2 Enable<br>Controls the digital filter 2 options for the external pin detect.<br>00b - Filter disabled<br>01b - Filter posedge detect enabled when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - Filter negedge detect enabled when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - Filter any edge detect enabled when configured as interrupt/DMA request |
| 12-8<br>FILTSEL2 | Filter 2 Pin Select<br>Selects 1 of the wakeup pins to be muxed into filter 2.<br>00000b - Select LLWU_P0 for filter<br>11111b - Select LLWU_P31 for filter |
| 7<br>FILTF1 | Filter 1 Flag<br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF.<br>0b - Pin Filter 1 was not a wakeup source<br>1b - Pin Filter 1 was a wakeup source |
| 6-5<br>FILTE1 | Filter 1 Enable<br>Controls the digital filter 1 options for the external pin detect.<br>00b - Filter disabled<br>01b - Filter posedge detect enabled when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - Filter negedge detect enabled when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - Filter any edge detect enabled when configured as interrupt/DMA request |
| 4-0<br>FILTSEL1 | Filter 1 Pin Select<br>Selects 1 of the wakeup pins to be muxed into filter 1.<br>00000b - Select LLWU_P0 for filter<br>11111b - Select LLWU_P31 for filter |

## 34.4.1.10 Pin DMA/Trigger Configuration 1 register (PDC1)

### 34.4.1.10.1 Offset

| Register | Offset |
|---|---|
| PDC1 | 38h |

### 34.4.1.10.2 Function

The PDC1 register configures the external wakeup input pins LLWU_P15-LLWU_P0 to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the PE1 register is detected and remains asserted until the corresponding flag is cleared in the PF register. When configured as a DMA request, the request is asserted when the edge programmed in the PE1 register is detected and remains asserted until either the corresponding flag is cleared in the PF register or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the PE1 register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the PF register will not be set.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPDC*n* field always reads as 00.

### 34.4.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPDC15 | | WUPDC14 | | WUPDC13 | | WUPDC12 | | WUPDC11 | | WUPDC10 | | WUPDC9 | | WUPDC8 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPDC7 | | WUPDC6 | | WUPDC5 | | WUPDC4 | | WUPDC3 | | WUPDC2 | | WUPDC1 | | WUPDC0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31-30:<br>WUPDC15 | Wakeup pin configuration for LLWU_Pn<br>Configures an external pin as an interrupt, DMA or trigger wakeup source.<br>00b - External input pin configured as interrupt |

| Field | Function |
|---|---|
| 29-28: WUPDC14<br>27-26: WUPDC13<br>25-24: WUPDC12<br>23-22: WUPDC11<br>21-20: WUPDC10<br>19-18: WUPDC9<br>17-16: WUPDC8<br>15-14: WUPDC7<br>13-12: WUPDC6<br>11-10: WUPDC5<br>9-8: WUPDC4<br>7-6: WUPDC3<br>5-4: WUPDC2<br>3-2: WUPDC1<br>1-0: WUPDC0 | 01b - External input pin configured as DMA request<br>10b - External input pin configured as trigger event<br>11b - Reserved |

## 34.4.1.11 Pin DMA/Trigger Configuration 2 register (PDC2)

### 34.4.1.11.1 Offset

| Register | Offset |
|---|---|
| PDC2 | 3Ch |

### 34.4.1.11.2 Function

The PDC2 register configures the external wakeup input pins LLWU_P31-LLWU_P16 to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the PE2 register is detected and remains asserted until the corresponding flag is cleared in the PF register. When configured as a DMA request, the request is asserted when the edge programmed in the PE2 register is detected and remains asserted until either the corresponding flag is cleared in the PF register or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the

PE2 register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the PF register will not be set.

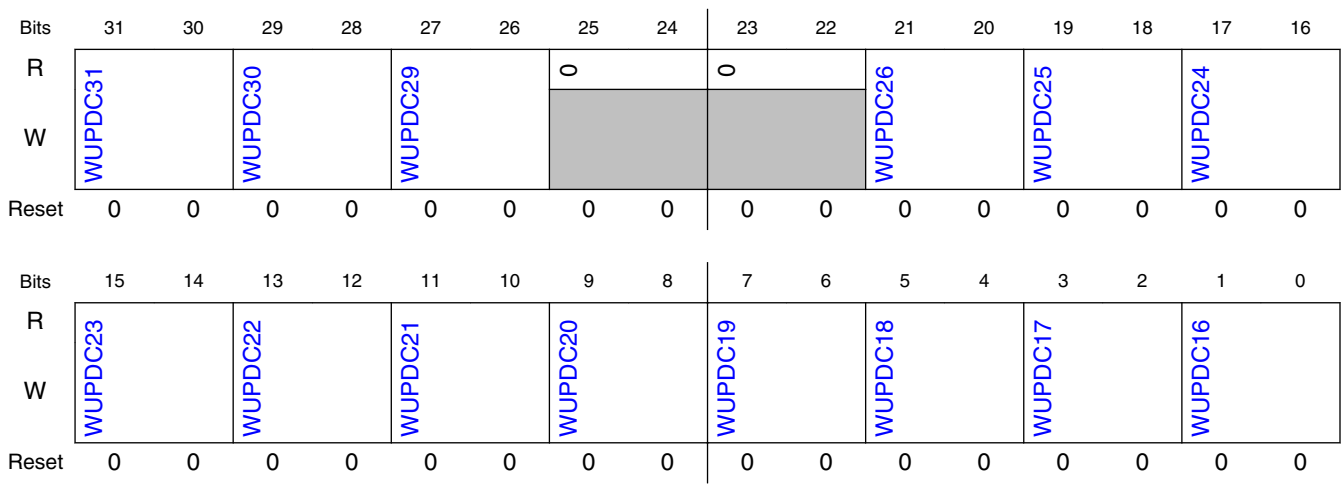
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPDC*n* field always reads as 00.

### 34.4.1.11.3 Diagram




### 34.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPDC31<br>29-28: WUPDC30<br>27-26: WUPDC29<br>25-24: Reserved28<br>23-22: Reserved27<br>21-20: WUPDC26 | Wakeup pin configuration for LLWU_Pn<br>Configures an external pin as an interrupt, DMA or trigger wakeup source.<br>00b - External input pin configured as interrupt<br>01b - External input pin configured as DMA request<br>10b - External input pin configured as trigger event<br>11b - Reserved |

| Field | Function |
|---|---|
| 19-18: WUPDC25 | |
| 17-16: WUPDC24 | |
| 15-14: WUPDC23 | |
| 13-12: WUPDC22 | |
| 11-10: WUPDC21 | |
| 9-8: WUPDC20 | |
| 7-6: WUPDC19 | |
| 5-4: WUPDC18 | |
| 3-2: WUPDC17 | |
| 1-0: WUPDC16 | |

## 34.4.1.12 Pin Filter DMA/Trigger Configuration register (FDC)

### 34.4.1.12.1 Offset

| Register | Offset |
|---|---|
| FDC | 48h |

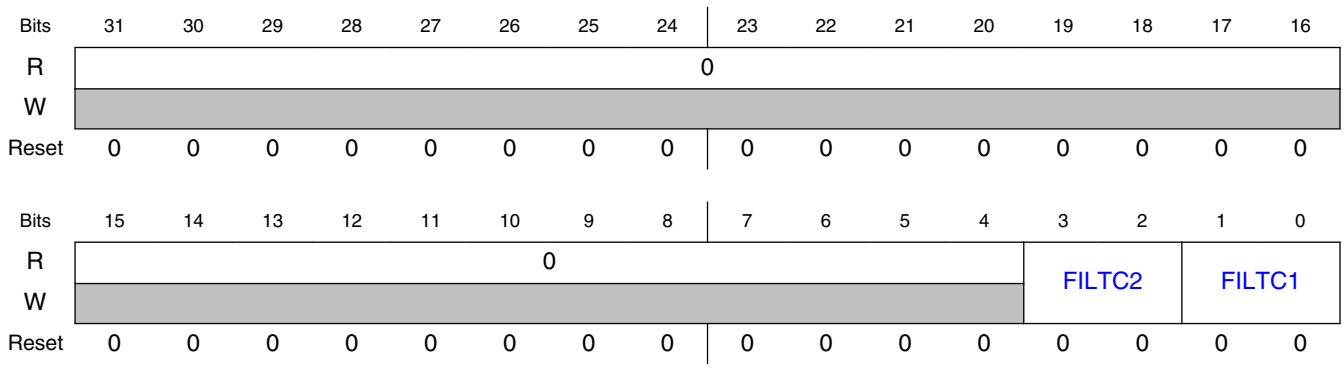
### 34.4.1.12.2 Function

The FDC register contains the field to configure the external pin filters to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the FILTEn field is detected and remains asserted until the corresponding flag is cleared in the FILTFn field. When configured as a DMA request, the request is asserted when the edge programmed in the FILTEn field is detected and remains asserted until either the corresponding flag is cleared in the FILTFn field or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the FILTEn register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the FILTFn field will not be set.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.1.12.3 Diagram




### 34.4.1.12.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved. |
| 3-2<br>FILTC2 | Filter configuration for FILT2<br>Configures the action taken for the filter.<br>00b - Filter output configured as interrupt<br>01b - Filter output configured as DMA request<br>10b - Filter output configured as trigger event<br>11b - Reserved |
| 1-0<br>FILTC1 | Filter configuration for FILT1<br>Configures a filter as an interrupt, DMA or trigger wakeup source.<br>00b - Filter output configured as interrupt<br>01b - Filter output configured as DMA request<br>10b - Filter output configured as trigger event<br>11b - Reserved |

## 34.4.1.13 Pin Mode Configuration register (PMC)

### 34.4.1.13.1 Offset

| Register | Offset |
|---|---|
| PMC | 50h |

### 34.4.1.13.2 Function

The PMC register configures the detection logic for the external wakeup input pins LLWU_P31-LLWU_P0 to remain enabled during all power modes, not just during LLS/VLLS.

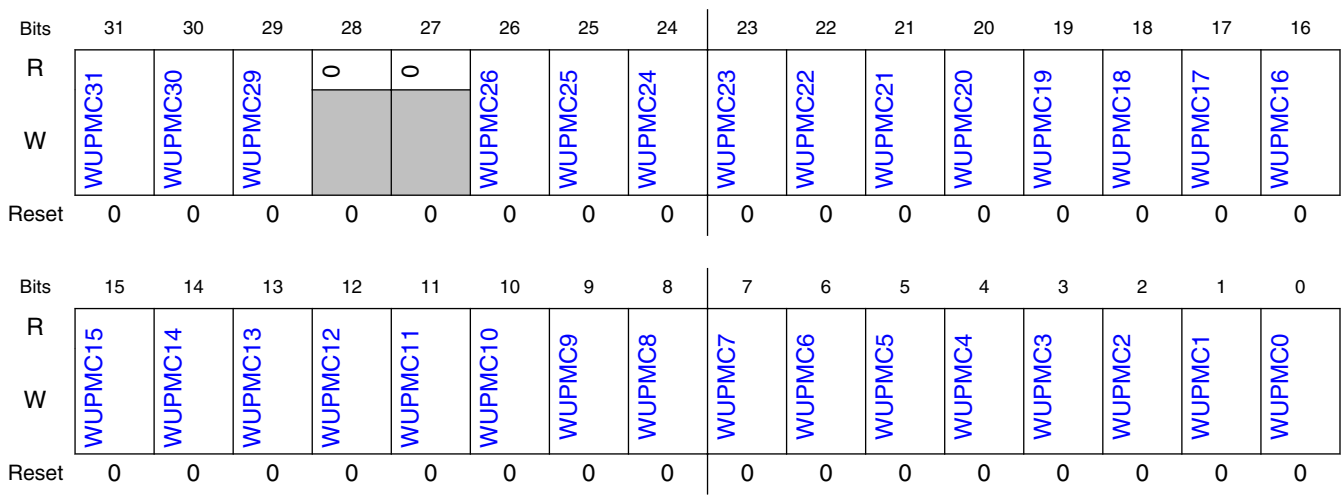
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPMC*n* bit always reads as a 0.

### 34.4.1.13.3 Diagram




### 34.4.1.13.4 Fields

| Field | Function |
|---|---|
| 31-0<br>WUPMCn | Wakeup pin mode for LLWU_Pn<br>Configures an external pin as a wakeup source during all power modes.<br>0b - External input pin detection active only during LLS/VLLS mode |

| Field | Function |
|---|---|
| | 1b - External input pin detection active during all power modes |

## 34.4.1.14 Pin Filter Mode Configuration register (FMC)

### 34.4.1.14.1 Offset

| Register | Offset |
|---|---|
| FMC | 58h |

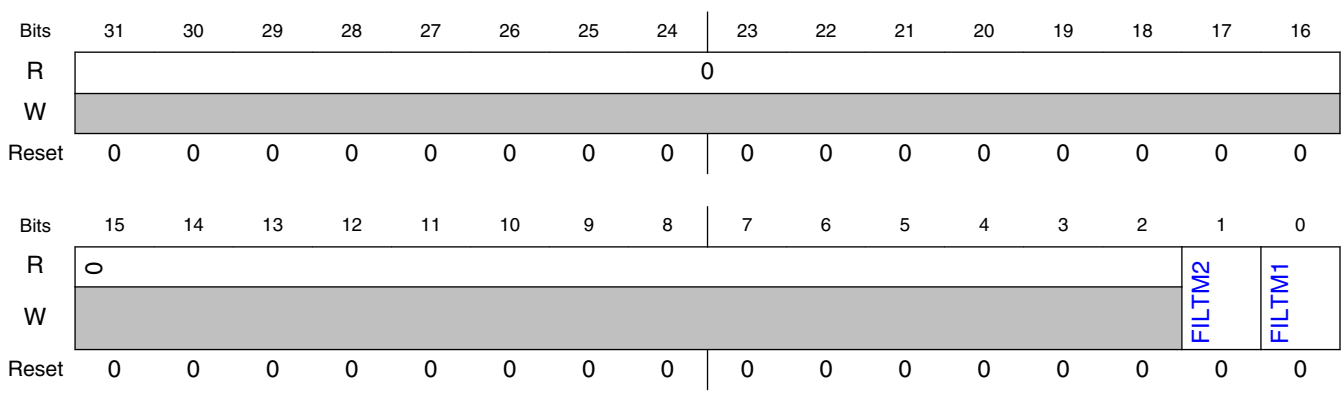
### 34.4.1.14.2 Function

The FMC register configures the detection logic for external pin filters to remain enabled during all power modes, not just during LLS/VLLS.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.1.14.3 Diagram




### 34.4.1.14.4 Fields

| Field | Function |
|---|---|
| 31-2 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 1<br>FILTM2 | Filter Mode for FILT2<br>Configures an external pin filter as a wakeup source during all power modes.<br>0b - External input pin filter detection active only during LLS/VLLS mode<br>1b - External input pin filter detection active during all power modes |
| 0<br>FILTM1 | Filter Mode for FILT1<br>Configures an external pin filter as a wakeup source during all power modes.<br>0b - External input pin filter detection active only during LLS/VLLS mode<br>1b - External input pin filter detection active during all power modes |

## 34.4.2 LLWU1 register descriptions

**NOTE**

The LLWU registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

Registers for Pin/Module sources marked "Reserved" in the LLWU chip configuration information will not be writable and will return 0 when read.

All LLWU registers are reset by Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.2.1 LLWU1 Memory map

LLWU1 base address: 4102_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0001h |
| 4h | Parameter Register (PARAM) | 32 | RO | 2020_0702h |
| 8h | Pin Enable 1 register (PE1) | 32 | RW | 0000_0000h |
| Ch | Pin Enable 2 register (PE2) | 32 | RW | 0000_0000h |
| 18h | Module Interrupt Enable register (ME) | 32 | RW | 0000_0000h |
| 1Ch | Module DMA/Trigger Enable register (DE) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20h | Pin Flag register (PF) | 32 | W1C | 0000_0000h |
| 30h | Pin Filter register (FILT) | 32 | RW | 0000_0000h |
| 38h | Pin DMA/Trigger Configuration 1 register (PDC1) | 32 | RW | 0000_0000h |
| 3Ch | Pin DMA/Trigger Configuration 2 register (PDC2) | 32 | RW | 0000_0000h |
| 48h | Pin Filter DMA/Trigger Configuration register (FDC) | 32 | RW | 0000_0000h |
| 50h | Pin Mode Configuration register (PMC) | 32 | RW | 0000_0000h |
| 58h | Pin Filter Mode Configuration register (FMC) | 32 | RW | 0000_0000h |

## 34.4.2.2 Version ID Register (VERID)

### 34.4.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 34.4.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 34.4.2.2.3 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | This read only field returns the major version number for the module specification. |
| 23-16 | Minor Version Number |
| MINOR | This read only field returns the minor version number for the module specification. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000000b - Standard features implemented<br>0000000000000001b - Support for DMA/Trigger generation from wakeup pins and filters enabled. Support for external pin/filter detection during all power modes enabled. |

## 34.4.2.3 Parameter Register (PARAM)

### 34.4.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 34.4.2.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINS | | | | | | | | MODULES | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMAS | | | | | | | | FILTERS | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

### 34.4.2.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>PINS | Pin Number<br>Number of Pin wakeup sources supported |
| 23-16<br>MODULES | Module Number<br>Number of Module wakeup sources supported |
| 15-8<br>DMAS | DMA Number<br>Number of DMA wakeup sources supported |
| 7-0<br>FILTERS | Filter Number<br>Number of Pin Filters implemented |

## 34.4.2.4 Pin Enable 1 register (PE1)

### 34.4.2.4.1 Offset

| Register | Offset |
|---|---|
| PE1 | 8h |

### 34.4.2.4.2 Function

The PE1 register contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P15-LLWU_P0.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPE*n* field always reads as 00.

### 34.4.2.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPE15 | | WUPE14 | | WUPE13 | | WUPE12 | | WUPE11 | | WUPE10 | | WUPE9 | | WUPE8 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPE7 | | WUPE6 | | WUPE5 | | WUPE4 | | WUPE3 | | WUPE2 | | WUPE1 | | WUPE0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.2.4.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPE15 | Wakeup pin enable for LLWU_Pn |
| 29-28: WUPE14 | Enables and configures the edge detection for the wakeup pin.<br>00b - External input pin disabled as wakeup input |

| Field | Function |
|---|---|
| 27-26: WUPE13<br>25-24: WUPE12<br>23-22: WUPE11<br>21-20: WUPE10<br>19-18: WUPE9<br>17-16: WUPE8<br>15-14: WUPE7<br>13-12: WUPE6<br>11-10: WUPE5<br>9-8: WUPE4<br>7-6: WUPE3<br>5-4: WUPE2<br>3-2: WUPE1<br>1-0: WUPE0 | 01b - External input pin enabled with rising edge detection when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - External input pin enabled with falling edge detection when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - External input pin enabled with any change detection when configured as interrupt/DMA request |

## 34.4.2.5 Pin Enable 2 register (PE2)

### 34.4.2.5.1 Offset

| Register | Offset |
|---|---|
| PE2 | Ch |

### 34.4.2.5.2 Function

The PE2 register contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P31-LLWU_P16.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPE*n* field always reads as 00.

### 34.4.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPE31 | | WUPE30 | | WUPE29 | | 0 | | 0 | | WUPE26 | | WUPE25 | | WUPE24 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPE23 | | WUPE22 | | WUPE21 | | WUPE20 | | WUPE19 | | WUPE18 | | WUPE17 | | WUPE16 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.2.5.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPE31 | Wakeup pin enable for LLWU_Pn |
| 29-28: WUPE30<br>27-26: WUPE29<br>25-24: Reserved28<br>23-22: Reserved27<br>21-20: WUPE26<br>19-18: WUPE25<br>17-16: WUPE24<br>15-14: WUPE23<br>13-12: WUPE22<br>11-10: WUPE21<br>9-8: WUPE20<br>7-6: WUPE19<br>5-4: WUPE18<br>3-2: WUPE17<br>1-0: WUPE16 | Enables and configures the edge detection for the wakeup pin.<br>00b - External input pin disabled as wakeup input<br>01b - External input pin enabled with rising edge detection when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - External input pin enabled with falling edge detection when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - External input pin enabled with any change detection when configured as interrupt/DMA request |

## 34.4.2.6 Module Interrupt Enable register (ME)

#### 34.4.2.6.1 Offset

| Register | Offset |
|---|---|
| ME | 18h |

#### 34.4.2.6.2 Function

The ME register contains the bits to enable an internal module interrupt as a wakeup source for inputs LLWU_M7IF - LLWU_M0IF.

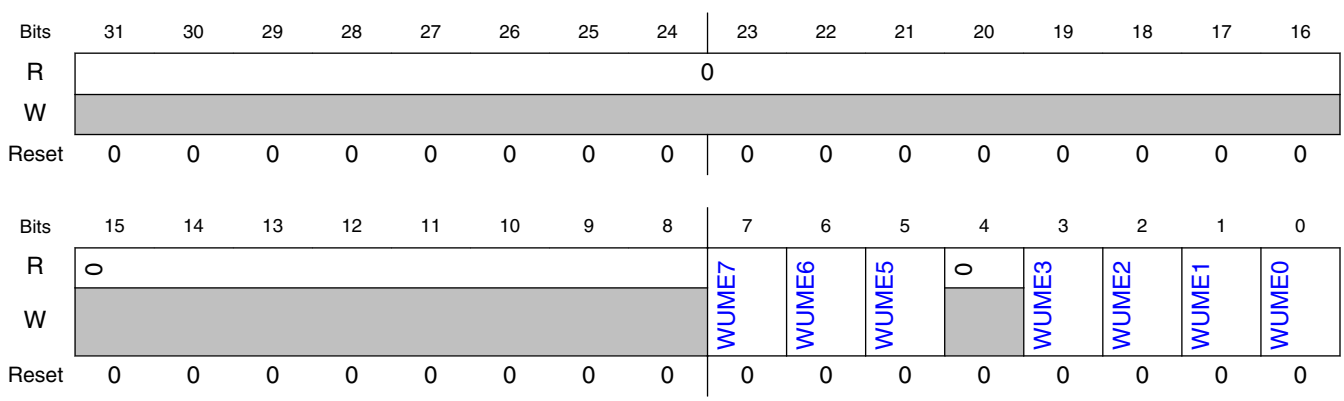
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If a module interrupt is not supported on a given device, the corresponding WUME*n* bit always reads as a 0.

#### 34.4.2.6.3 Diagram




#### 34.4.2.6.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>WUMEn | Wakeup module enable for module n<br>Enables an internal module as a wakeup source input.<br>0b - Internal module flag not used as wakeup source<br>1b - Internal module flag used as wakeup source |

## 34.4.2.7 Module DMA/Trigger Enable register (DE)

### 34.4.2.7.1 Offset

| Register | Offset |
|---|---|
| DE | 1Ch |

### 34.4.2.7.2 Function

The DE register contains the bits to enable an internal module DMA/Trigger request as a wakeup source for inputs LLWU_M7DR - LLWU_M0DR.

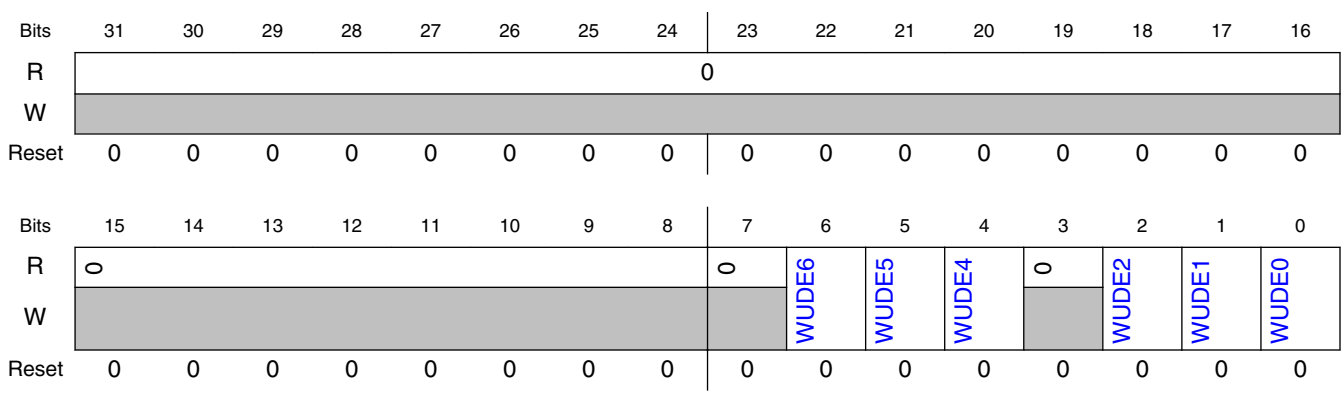
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If a module DMA/Trigger request is not supported on a given device, the corresponding WUDE*n* bit always reads as a 0.

### 34.4.2.7.3 Diagram




### 34.4.2.7.4 Fields

| Field | Function |
|---|---|
| 31-8 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 7-0<br>WUDEn | DMA/Trigger wakeup enable for module n<br>Enables an internal module as a DMA/Trigger wakeup source.<br>0b - Internal module request not enabled as a DMA/Trigger wakeup source<br>1b - Internal module request enabled as a DMA/Trigger wakeup source |

## 34.4.2.8 Pin Flag register (PF)

### 34.4.2.8.1 Offset

| Register | Offset |
|---|---|
| PF | 20h |

### 34.4.2.8.2 Function

The PF register contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUF*n* bit. The wakeup flag (WUF*n*), if set, will remain set if the associated WUPE*n* bit is cleared.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUF*n* bit always reads as a 0.

### 34.4.2.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUF31 | WUF30 | WUF29 | 0 | 0 | WUF26 | WUF25 | WUF24 | WUF23 | WUF22 | WUF21 | WUF20 | WUF19 | WUF18 | WUF17 | WUF16 |
| W | W1C | W1C | W1C | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUF15 | WUF14 | WUF13 | WUF12 | WUF11 | WUF10 | WUF9 | WUF8 | WUF7 | WUF6 | WUF5 | WUF4 | WUF3 | WUF2 | WUF1 | WUF0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.2.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>WUFn | Wakeup flag for LLWU_Pn<br>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF*n*.<br>0b - LLWU_Pn input was not a wakeup source<br>1b - LLWU_Pn input was a wakeup source |

## 34.4.2.9 Pin Filter register (FILT)

### 34.4.2.9.1 Offset

| Register | Offset |
|---|---|
| FILT | 30h |

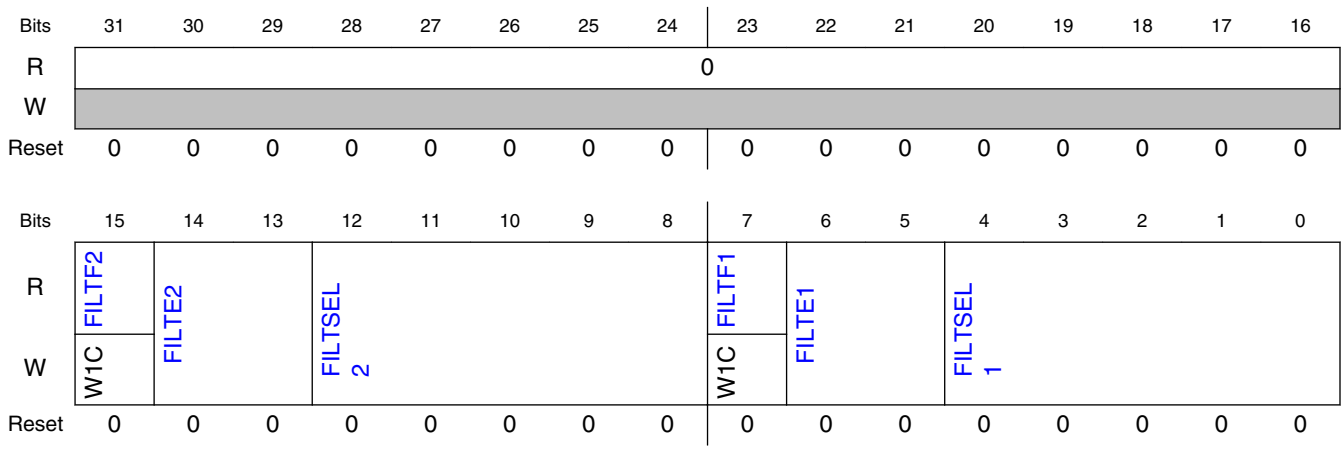
### 34.4.2.9.2 Function

The FILT register is a control and status register that is used to enable/disable the digital filter features for an external pin.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.2.9.3 Diagram




### 34.4.2.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15<br>FILTF2 | Filter 2 Flag<br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF.<br>0b - Pin Filter 2 was not a wakeup source<br>1b - Pin Filter 2 was a wakeup source |
| 14-13<br>FILTE2 | Filter 2 Enable<br>Controls the digital filter 2 options for the external pin detect.<br>00b - Filter disabled<br>01b - Filter posedge detect enabled when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - Filter negedge detect enabled when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - Filter any edge detect enabled when configured as interrupt/DMA request |
| 12-8<br>FILTSEL2 | Filter 2 Pin Select<br>Selects 1 of the wakeup pins to be muxed into filter 2.<br>00000b - Select LLWU_P0 for filter<br>11111b - Select LLWU_P31 for filter |
| 7<br>FILTF1 | Filter 1 Flag<br>Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Pin Filter 1 was not a wakeup source<br>1b - Pin Filter 1 was a wakeup source |
| 6-5<br>FILTE1 | Filter 1 Enable<br>Controls the digital filter 1 options for the external pin detect.<br>00b - Filter disabled<br>01b - Filter posedge detect enabled when configured as interrupt/DMA request or high level detection when configured as trigger request<br>10b - Filter negedge detect enabled when configured as interrupt/DMA request or low level detection when configured as trigger request<br>11b - Filter any edge detect enabled when configured as interrupt/DMA request |
| 4-0<br>FILTSEL1 | Filter 1 Pin Select<br>Selects 1 of the wakeup pins to be muxed into filter 1.<br>00000b - Select LLWU_P0 for filter<br>11111b - Select LLWU_P31 for filter |

## 34.4.2.10 Pin DMA/Trigger Configuration 1 register (PDC1)

### 34.4.2.10.1 Offset

| Register | Offset |
|---|---|
| PDC1 | 38h |

### 34.4.2.10.2 Function

The PDC1 register configures the external wakeup input pins LLWU_P15-LLWU_P0 to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the PE1 register is detected and remains asserted until the corresponding flag is cleared in the PF register. When configured as a DMA request, the request is asserted when the edge programmed in the PE1 register is detected and remains asserted until either the corresponding flag is cleared in the PF register or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the PE1 register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the PF register will not be set.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPDC*n* field always reads as 00.

### 34.4.2.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPDC15 | | WUPDC14 | | WUPDC13 | | WUPDC12 | | WUPDC11 | | WUPDC10 | | WUPDC9 | | WUPDC8 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPDC7 | | WUPDC6 | | WUPDC5 | | WUPDC4 | | WUPDC3 | | WUPDC2 | | WUPDC1 | | WUPDC0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.2.10.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPDC15<br>29-28: WUPDC14<br>27-26: WUPDC13<br>25-24: WUPDC12<br>23-22: WUPDC11<br>21-20: WUPDC10<br>19-18: WUPDC9<br>17-16: WUPDC8<br>15-14: WUPDC7<br>13-12: WUPDC6<br>11-10: WUPDC5<br>9-8: WUPDC4<br>7-6: WUPDC3<br>5-4: WUPDC2<br>3-2: WUPDC1<br>1-0: WUPDC0 | Wakeup pin configuration for LLWU_Pn<br><br>Configures an external pin as an interrupt, DMA or trigger wakeup source.<br>00b - External input pin configured as interrupt<br>01b - External input pin configured as DMA request<br>10b - External input pin configured as trigger event<br>11b - Reserved |

## 34.4.2.11 Pin DMA/Trigger Configuration 2 register (PDC2)

### 34.4.2.11.1 Offset

| Register | Offset |
|---|---|
| PDC2 | 3Ch |

### 34.4.2.11.2 Function

The PDC2 register configures the external wakeup input pins LLWU_P31-LLWU_P16 to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the PE2 register is detected and remains asserted until the corresponding flag is cleared in the PF register. When configured as a DMA request, the request is asserted when the edge programmed in the PE2 register is detected and remains asserted until either the corresponding flag is cleared in the PF register or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the PE2 register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the PF register will not be set.

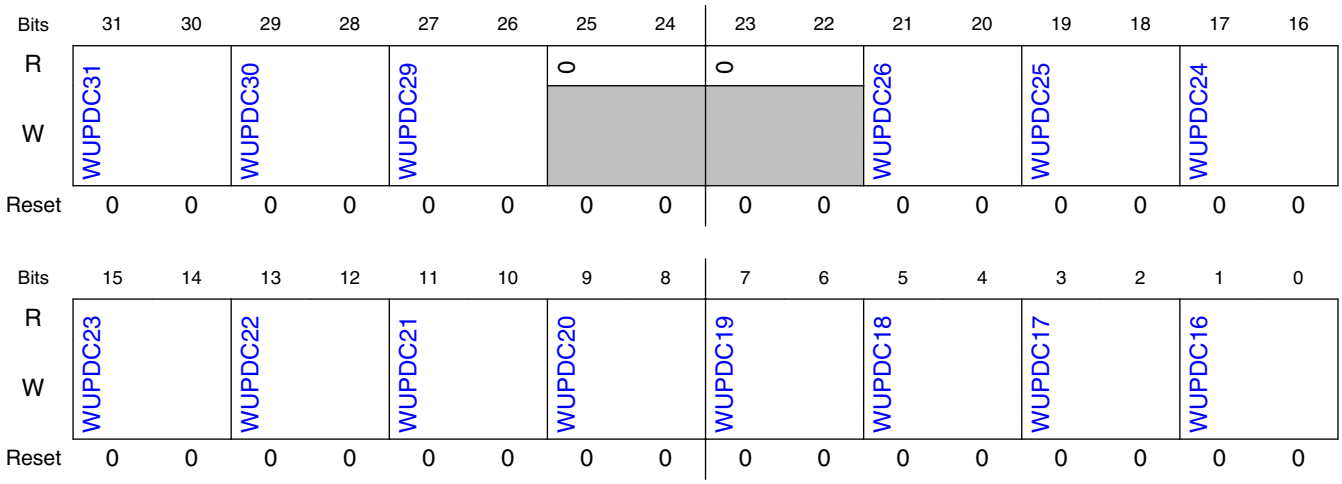
**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPDC*n* field always reads as 00.

### 34.4.2.11.3 Diagram




### 34.4.2.11.4 Fields

| Field | Function |
|---|---|
| 31-30: WUPDC31<br>29-28: WUPDC30<br>27-26: WUPDC29<br>25-24: Reserved28<br>23-22: Reserved27<br>21-20: WUPDC26<br>19-18: WUPDC25<br>17-16: WUPDC24<br>15-14: WUPDC23<br>13-12: WUPDC22<br>11-10: WUPDC21<br>9-8: WUPDC20<br>7-6: WUPDC19<br>5-4: WUPDC18<br>3-2: WUPDC17 | Wakeup pin configuration for LLWU_Pn<br>Configures an external pin as an interrupt, DMA or trigger wakeup source.<br>00b - External input pin configured as interrupt<br>01b - External input pin configured as DMA request<br>10b - External input pin configured as trigger event<br>11b - Reserved |

| Field | Function |
|---|---|
| 1-0: WUPDC16 | |

## 34.4.2.12 Pin Filter DMA/Trigger Configuration register (FDC)

### 34.4.2.12.1 Offset

| Register | Offset |
|---|---|
| FDC | 48h |

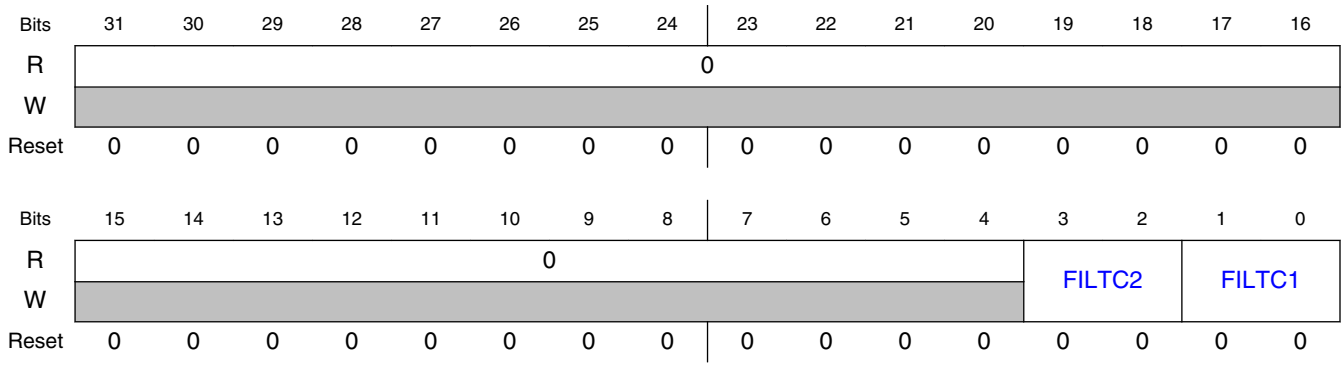
### 34.4.2.12.2 Function

The FDC register contains the field to configure the external pin filters to generate an interrupt, DMA or a trigger request when detected. When configured as an interrupt, the interrupt is asserted when the edge programmed in the FILTEn field is detected and remains asserted until the corresponding flag is cleared in the FILTFn field. When configured as a DMA request, the request is asserted when the edge programmed in the FILTEn field is detected and remains asserted until either the corresponding flag is cleared in the FILTFn field or until the requested DMA transfer is complete. When configured as a trigger request, the trigger is asserted when the level programmed in the FILTEn register is detected and remains asserted while the input pin is asserted. Note that when configured as a trigger request, the corresponding flag in the FILTFn field will not be set.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.2.12.3 Diagram




### 34.4.2.12.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved. |
| 3-2<br>FILTC2 | Filter configuration for FILT2<br>Configures the action taken for the filter.<br>00b - Filter output configured as interrupt<br>01b - Filter output configured as DMA request<br>10b - Filter output configured as trigger event<br>11b - Reserved |
| 1-0<br>FILTC1 | Filter configuration for FILT1<br>Configures a filter as an interrupt, DMA or trigger wakeup source.<br>00b - Filter output configured as interrupt<br>01b - Filter output configured as DMA request<br>10b - Filter output configured as trigger event<br>11b - Reserved |

## 34.4.2.13 Pin Mode Configuration register (PMC)

### 34.4.2.13.1 Offset

| Register | Offset |
|---|---|
| PMC | 50h |

### 34.4.2.13.2 Function

The PMC register configures the detection logic for the external wakeup input pins LLWU_P31-LLWU_P0 to remain enabled during all power modes, not just during LLS/VLLS.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

**NOTE**

If an external wakeup pin is not supported on a given device, the corresponding WUPMC*n* bit always reads as a 0.

### 34.4.2.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPMC31 | WUPMC30 | WUPMC29 | 0 | 0 | WUPMC26 | WUPMC25 | WUPMC24 | WUPMC23 | WUPMC22 | WUPMC21 | WUPMC20 | WUPMC19 | WUPMC18 | WUPMC17 | WUPMC16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WUPMC15 | WUPMC14 | WUPMC13 | WUPMC12 | WUPMC11 | WUPMC10 | WUPMC9 | WUPMC8 | WUPMC7 | WUPMC6 | WUPMC5 | WUPMC4 | WUPMC3 | WUPMC2 | WUPMC1 | WUPMC0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.4.2.13.4 Fields

| Field | Function |
|---|---|
| 31-0<br>WUPMCn | Wakeup pin mode for LLWU_Pn<br>Configures an external pin as a wakeup source during all power modes.<br>0b - External input pin detection active only during LLS/VLLS mode<br>1b - External input pin detection active during all power modes |

## 34.4.2.14 Pin Filter Mode Configuration register (FMC)

### 34.4.2.14.1 Offset

| Register | Offset |
|---|---|
| FMC | 58h |

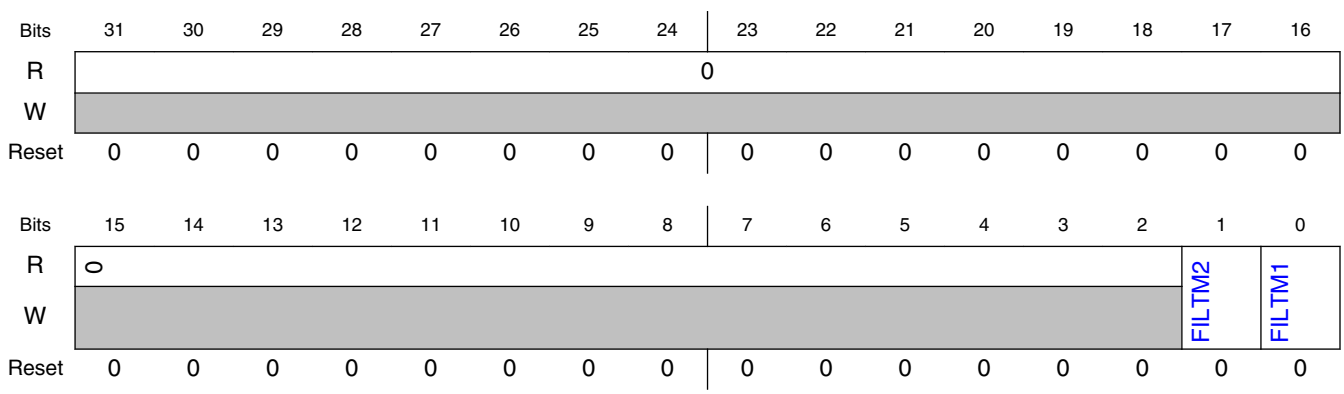
### 34.4.2.14.2 Function

The FMC register configures the detection logic for external pin filters to remain enabled during all power modes, not just during LLS/VLLS.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS.

### 34.4.2.14.3 Diagram




### 34.4.2.14.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved. |
| 1<br>FILTM2 | Filter Mode for FILT2<br>Configures an external pin filter as a wakeup source during all power modes.<br>0b - External input pin filter detection active only during LLS/VLLS mode<br>1b - External input pin filter detection active during all power modes |
| 0<br>FILTM1 | Filter Mode for FILT1<br>Configures an external pin filter as a wakeup source during all power modes.<br>0b - External input pin filter detection active only during LLS/VLLS mode<br>1b - External input pin filter detection active during all power modes |

## 34.5 Functional description

The low-leakage wakeup unit (LLWU) module allows internal peripherals and external input pins as a source of wakeup from low-leakage modes.

The LLWU module contains pin enables for each external pin and internal module. For each external pin, the user can disable or select the edge type for the wakeup with the following options:

- Falling-edge
- Rising-edge
- Either-edge

Exit due to a pin event will trigger an LLWU interrupt after wakeup is complete. When an external pin is enabled as a wakeup source, the pin must be configured as an input pin. Detection logic for a pin can optionally remain enabled during all power modes using the LLWU_PMC register. A pin can also be configured to generate an interrupt from LLS/VLLS or a DMA/Trigger request for temporary wakeup from LLS.

The LLWU implements optional 3-cycle glitch filters, based on the LPO clock. A detected external pin is required to remain asserted until the enabled glitch filter times out. Additional latency of up to 2 cycles is due to synchronization, which results in a total of up to 5 cycles of delay before the detect circuit alerts the system to the wakeup or reset event when the filter function is enabled. Two wakeup detect filters are available for selected external pins. Glitch filtering is not provided on the internal modules. A filter can be configured to generate an interrupt from LLS/VLLS or a DMA/Trigger request for temporary wakeup from LLS.Detection logic for a filter can optionally remain enabled during all power modes using the LLWU_FMC register.

For internal module interrupts, the WUMEx bit enables the associated module interrupt as a wakeup source. Exit due to a module interrupt will not trigger an LLWU interrupt.

### 34.5.1 LLS mode

Wakeup events triggered from either an external pin input or an internal module interrupt, result in a CPU interrupt flow to begin user code execution. Pin events will trigger the LLWU interrupt routine, but module interrupts will only trigger the module's interrupt routine.

Wakeup events triggered from an internal module or external pin DMA/Trigger request, result in a temporary wake-up from LLS with CPU remaining clock gated to allow the DMA/Trigger request to be serviced. After the DMA/Trigger request is cleared, the system will re-enter LLS mode.

An LLS reset event due to RESET_B pin assertion causes an exit via a system reset. State retention data is lost, and the I/O states return to their reset state.

### 34.5.2 VLLS modes

For any LLWU wakeup from VLLS, recovery is always via a reset flow. Pin events will trigger the LLWU interrupt routine and module interrupts will trigger the module's interrupt routine. A VLLS exit event due to RESET_B pin assertion causes an exit via a system reset with I/O returning to their reset state immediately.

### 34.5.3 Initialization

For an enabled peripheral wakeup input, the peripheral flag must be cleared by software before entering LLS or VLLS mode to avoid an immediate exit from the mode.

Flags associated with external input pins, filtered and unfiltered, must also be cleared by software prior to entry to LLS or VLLS mode.

After enabling an external pin filter or changing the source pin, wait at least five LPO clock cycles to allow the filter to initialize before entering LLS or VLLS mode or before configuring the FMC register for operation in any power mode.

**NOTE**

After recovering from a VLLS mode, user must restore chip configuration before clearing the PMC ACKISO bit. In particular, pin configuration for enabled LLWU wake-up pins must be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.

# Chapter 35
# System Integration Module (SIM)

## 35.1 Introduction

### 35.1.1 Overview

This section provides an introduction to the system integration module (SIM) module. The SIM provides system control and chip configuration registers.

### 35.1.2 Features

The SIM includes the following features:

- Flash and System RAM size configuration
- Flexbus configuration

## 35.2 Memory Map and Registers

### 35.2.1 SIM register descriptions

#### 35.2.1.1 SIM Memory map

SIM base address: 4002_6000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Chip Control Register (CHIPCTRL) | 32 | RW | 0000_0000h |
| 24h | System Device Identification Register (SDID) | 32 | RO | See description. |
| 4Ch | Flash Configuration Register 1 (FCFG1) | 32 | RW | CAA9_0400h |
| 50h | Flash Configuration Register 2 (FCFG2) | 32 | RO | See description. |
| 58h | Unique Identification Register High (UIDH) | 32 | RO | See description. |
| 5Ch | Unique Identification Register Mid Middle (UIDM) | 32 | RO | See description. |
| 60h | Unique Identification Register Mid Low (UIDL) | 32 | RO | See description. |
| 64h | RF Mac Address Low (RFADDRL) | 32 | RO | See description. |
| 68h | RF MAC Address High (RFADDRH) | 32 | RO | See description. |
| 70h | MISC2 Register (MISC2) | 32 | RW | 0000_0000h |

## 35.2.1.2 Chip Control Register (CHIPCTRL)

### 35.2.1.2.1 Offset

| Register | Offset |
|---|---|
| CHIPCTRL | 4h |

### 35.2.1.2.2 Function

This register is used for chip control.

#### 35.2.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | FBSL | | Reserved | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 35.2.1.2.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-8<br>FBSL | FLEXBUS security level<br>If flash security is enabled, then this field affects what CPU operation can access off-chip via the Flexbus or SDRAM interface. This field has no effect if flash security is not enabled<br>00b - All off-chip access(instruction and data) via the Flexbus or sdram are disallowed<br>01b - All off-chip access(instruction and data) via the Flexbus or sdram are disallowed<br>10b - off-chip instruction access are disallowed, data access are allowed<br>11b - off-chip instruction access and data access are allowed |
| 7-0<br>— | Reserved. |

### 35.2.1.3 System Device Identification Register (SDID)

#### 35.2.1.3.1 Offset

| Register | Offset |
|---|---|
| SDID | 24h |

## 35.2.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FAMID | | | | SUBFAMID | | | | SERIESID | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | u[1] | u | u | u | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REVID | | | | DIEID | | | | 0 | | | | PINID | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u |

1. Reset values are loaded out of IFR.

## 35.2.1.3.3 Fields

| Field | Function |
|---|---|
| 31-28<br>FAMID | FAMID<br>Kinetis family ID Specifies the Kinetis family of the device.<br>0000b - RV32 |
| 27-24<br>SUBFAMID | SUBFAMID<br>Kinetis Sub-Family ID Specifies the Kinetis sub-family of the device.<br>0100b - M1 |
| 23-20<br>SERIESID | SERIESID<br>Kinetis Series ID Specifies the Kinetis family of the device. |
| 19-16<br>— | Reserved. |
| 15-12<br>REVID | REVID<br>Device Revision Number Specifies the silicon implementation number for the device. |
| 11-7<br>DIEID | DIEID<br>Device Die Number Specifies the silicon implementation number for the device. This field always reads as 00000. |
| 6-4<br>— | RESERVED |
| 3-0<br>PINID | PINID<br>Pin count identification Specifies the pin count of the device.<br>1000b - 176-pin |

# 35.2.1.4 Flash Configuration Register 1 (FCFG1)

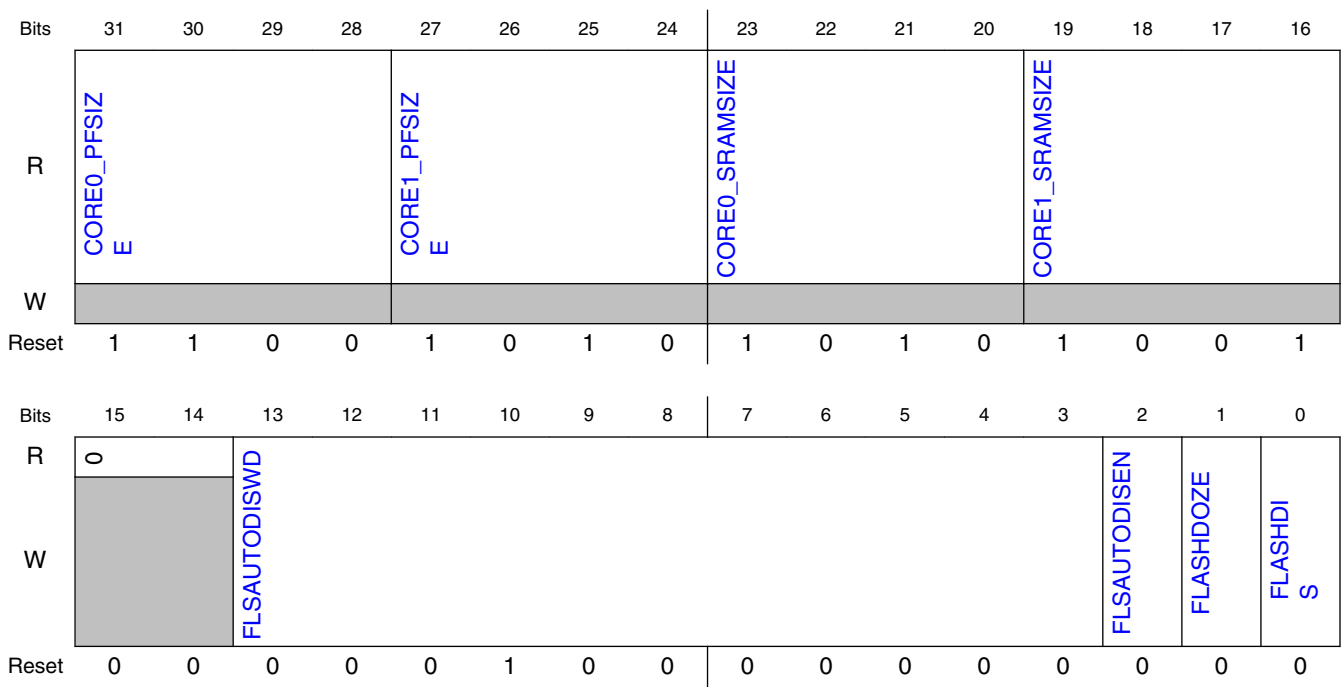
## 35.2.1.4.1 Offset

| Register | Offset |
|---|---|
| FCFG1 | 4Ch |

## 35.2.1.4.2 Function

It stores the flash and SRAM size of CM4 and CM0+ cores.

## 35.2.1.4.3 Diagram




## 35.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-28<br>CORE0_PFSIZE | The flash size for core0 (CM4)<br>1100b - CM4 has 1 MB flash size. |
| 27-24<br>CORE1_PFSIZE | The flash size for core1 (CM0+)<br>1010b - CM0+ has 256 KB flash size. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-20<br>CORE0_SRAM SIZE | The SRAM size for core0 (CM4)<br>1010b - CM4 has 256 KB SRAM |
| 19-16<br>CORE1_SRAM SIZE | The SRAM size for core1 (CM0+)<br>1001b - CM0+ has 128 KB SRAM |
| 15-14<br>— | Reserved. |
| 13-3<br>FLSAUTODISWD | The clock counter for time period of flash auto disable.<br>These 11 bits are used for clock counter setting. When the flash auto disable function is enabled, the internal clock starts to count the flash idle time. When the clock counter equals to these values. The flash will be automatically disabled.<br>• The flash auto disable function only works in RUN, HSRUN, and VLPR modes.<br>• When flash auto disable function is enabled, only flash read is supported.<br>• FLSAUTODISEN must be set to 0 if customers want to read and write flash register to program or erase flash.<br>• After enabling the feature, accessing flash will be delayed. |
| 2<br>FLSAUTODISEN | Flash auto disable enabled.<br>Flash Auto Disable function control. When enabled, flash will enter disable state to save power if no flash access is detected for a certain period. When this bit is enabled, a counter starts to accumulate if there is no flash access. When the counter reaches the FLSAUTODISWD, flash will enter disable state.<br>**NOTE:** There is a delay in re-enabling the flash. When flash is in disable state, flash register access is not permitted.<br>0b - Disable flash auto disable function<br>1b - Enable flash auto disable function |
| 1<br>FLASHDOZE | Flash Doze<br>When set, flash memory is disabled for the duration of Doze mode. This field must be clear during VLP modes. The flash will be automatically enabled again at the end of Doze mode so interrupt vectors do not need to be relocated out of flash memory. The wake-up time from Doze mode is extended when this field is set.<br>0b - Flash remains enabled during Doze mode<br>1b - Flash is disabled for the duration of Doze mode |
| 0<br>FLASHDIS | Flash disable<br>Flash accesses are disabled (and generate a bus error) and the flash memory is placed in a low-power state. This field should not be changed during VLP modes. Relocate the interrupt vectors out of Flash memory before disabling the Flash.<br>0b - Flash is enabled<br>1b - Flash is disabled |

## 35.2.1.5 Flash Configuration Register 2 (FCFG2)

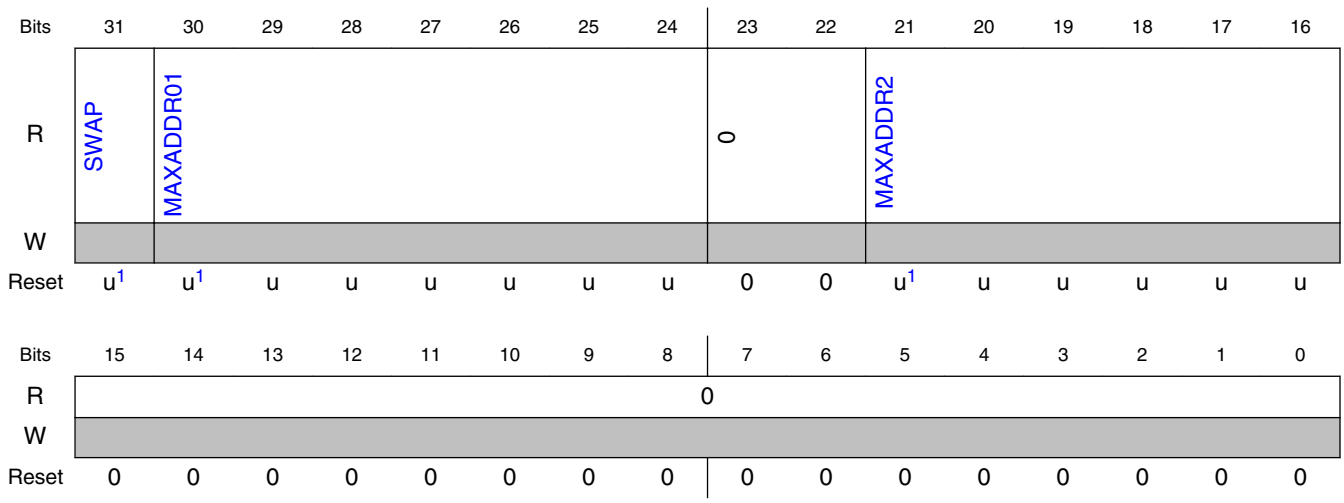
### 35.2.1.5.1 Offset

| Register | Offset |
|---|---|
| FCFG2 | 50h |

### 35.2.1.5.2 Function

This is read only register, any write to this register will cause transfer error. Reset value of MAXADDR loaded during System Reset from Flash IFR

### 35.2.1.5.3 Diagram




1. Reset values are loaded out of IFR.

### 35.2.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>SWAP | SWAP<br>SWAP bit from flash FCNFG register<br>0b - Logical P-flash Block 0 is located at relative address 0x0000<br>1b - Logical P-flash Block 1 is located at relative address 0x0000 |
| 30-24<br>MAXADDR01 | Max Address lock<br>This field shows the number of 8 Kbyte regions in logical Flash block 0 or 1 |
| 23-22<br>— | Reserved |
| 21-16<br>MAXADDR2 | Max Address lock<br>This field shows the number of 8 Kbyte regions in logical Flash block 2 |
| 15-0<br>— | Reserved |

## 35.2.1.6 Unique Identification Register High (UIDH)

### 35.2.1.6.1 Offset

| Register | Offset |
|---|---|
| UIDH | 58h |

### 35.2.1.6.2 Function

Unique identification for the device. It is set by IFR bits.

### 35.2.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

1. Reset values are loaded out of IFR.

### 35.2.1.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-0<br>UID | Unique Identification<br>Unique identification for the device. It is set by IFR bits. |

## 35.2.1.7 Unique Identification Register Mid Middle (UIDM)

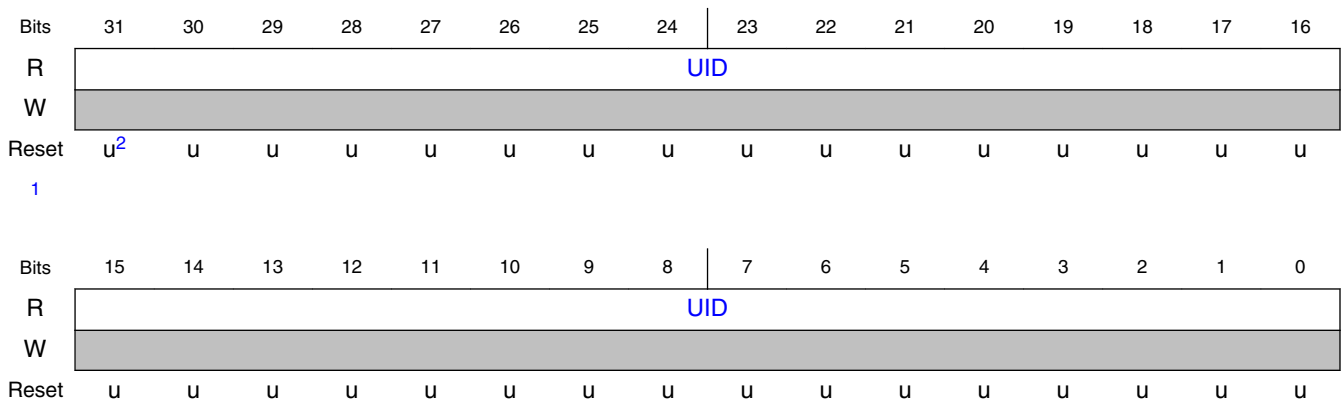
#### 35.2.1.7.1 Offset

| Register | Offset |
|---|---|
| UIDM | 5Ch |

#### 35.2.1.7.2 Function

Unique identification for the device. It is set by IFR bits.

#### 35.2.1.7.3 Diagram




1. Reset values are loaded out of IFR.
2. Reset values are loaded out of IFR.

#### 35.2.1.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>UID | Unique Identification<br>Unique identification for the device. It is set by IFR bits. |

### 35.2.1.8 Unique Identification Register Mid Low (UIDL)

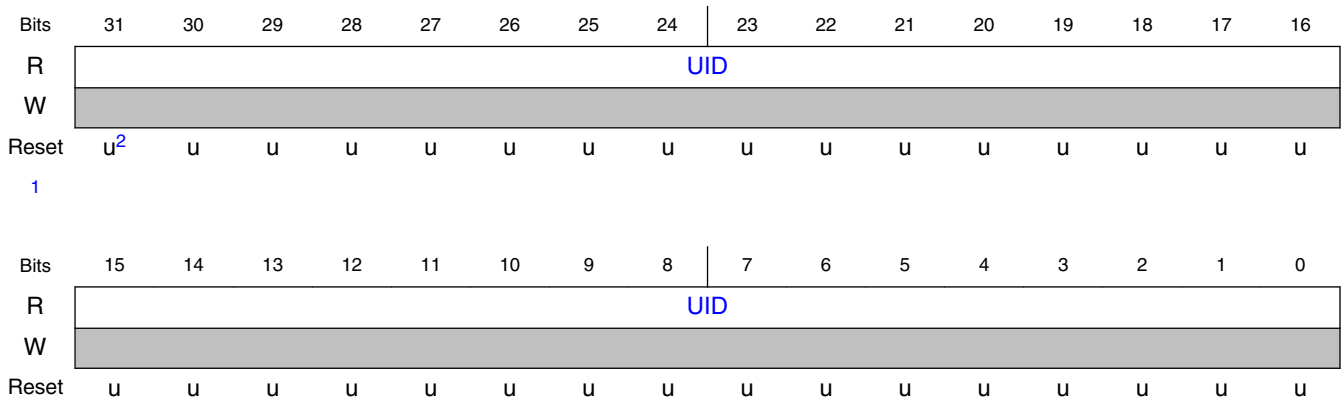
#### 35.2.1.8.1 Offset

| Register | Offset |
|---|---|
| UIDL | 60h |

### 35.2.1.8.2 Function

Unique identification for the device. It is set by IFR bits.

### 35.2.1.8.3 Diagram




1. Reset values are loaded out of IFR.
2. Reset values are loaded out of IFR.

### 35.2.1.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>UID | Unique Identification<br>Unique identification for the device. It is set by IFR bits. |

## 35.2.1.9 RF Mac Address Low (RFADDRL)

### 35.2.1.9.1 Offset

| Register | Offset |
|---|---|
| RFADDRL | 64h |

### 35.2.1.9.2 Function

MAC address loaded from IFR

### 35.2.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MACADDR3 | | | | | | | | MACADDR2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | u | u | u | u | u[1] | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MACADDR1 | | | | | | | | MACADDR0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | u | u | u | u | u[1] | u | u | u | u | u | u | u |

1. Reset values are loaded out of IFR.

### 35.2.1.9.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MACADDR3 | MACADDR3<br>RF address. Loaded from IFR |
| 23-16<br>MACADDR2 | MACADDR2<br>RF address. Loaded from IFR |
| 15-8<br>MACADDR1 | MACADDR1<br>RF address. Loaded from IFR |
| 7-0<br>MACADDR0 | MACADDR0<br>RF address. Loaded from IFR |

## 35.2.1.10 RF MAC Address High (RFADDRH)

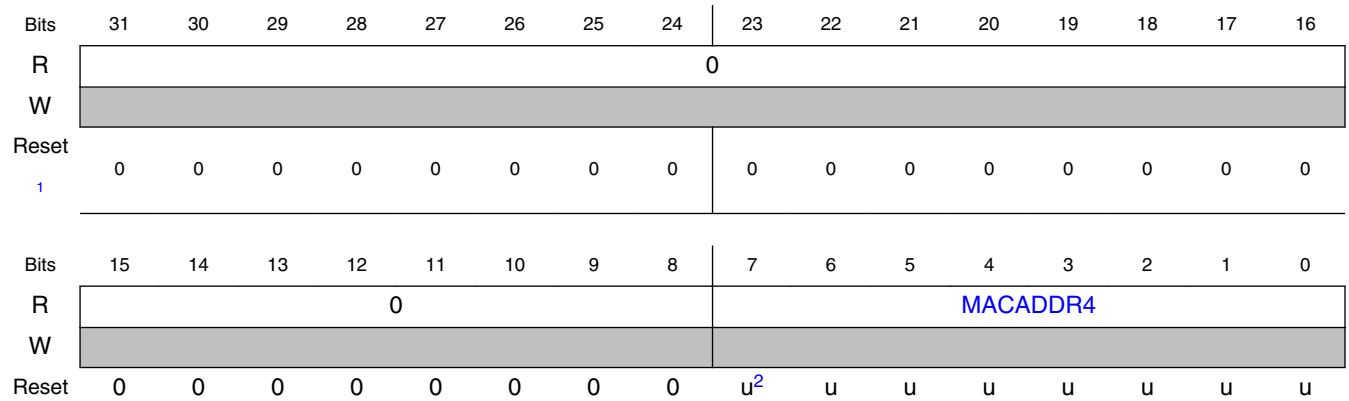
### 35.2.1.10.1 Offset

| Register | Offset |
|---|---|
| RFADDRH | 68h |

### 35.2.1.10.2 Function

MAC address loaded from IFR

### 35.2.1.10.3 Diagram




1. Reset values are loaded out of IFR.
2. Reset values are loaded out of IFR.

### 35.2.1.10.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>MACADDR4 | MACADDR4<br>RF address. Loaded from IFR |

## 35.2.1.11 MISC2 Register (MISC2)

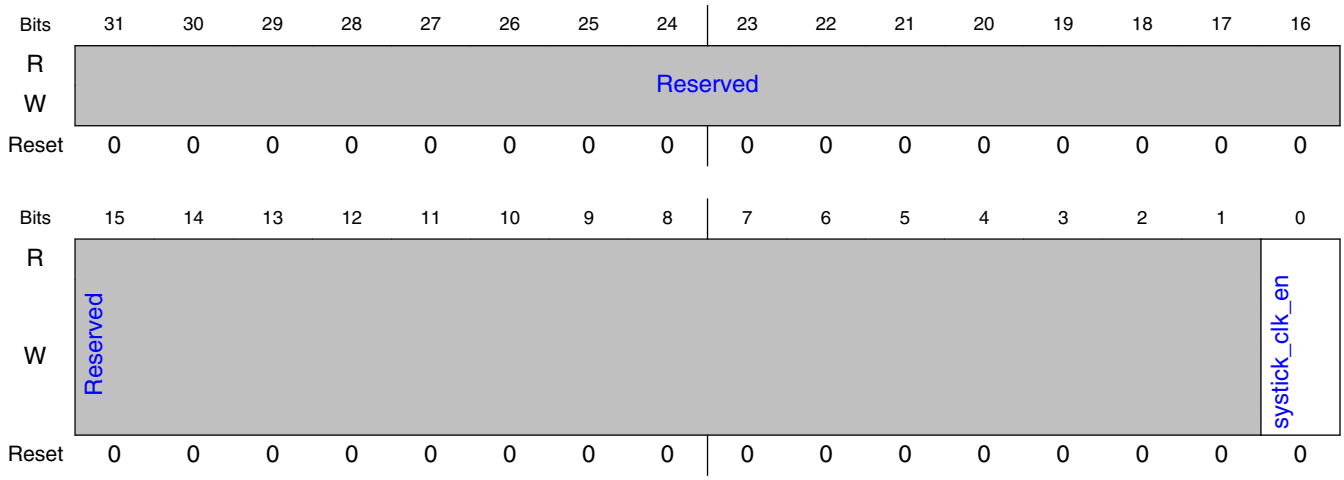
### 35.2.1.11.1 Offset

| Register | Offset |
|---|---|
| MISC2 | 70h |

### 35.2.1.11.2 Function

This register is used for some misc logic

#### 35.2.1.11.3 Diagram




#### 35.2.1.11.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>systick_clk_en | Systick clock enable<br>This bit is used to disable/enable systick clock.<br>0b - Systick clock is disabled<br>1b - Systick clock is enabled |

## 35.3 Application information

This section describes applications supported by the SIM module.

# Chapter 36
# Interrupt Multiplexer (INTMUX)

## 36.1 About this module

### 36.1.1 Introduction

The Interrupt Multiplexer (INTMUX) expands the number of peripherals that can interrupt the core via 4 channels of the NVIC module.

- The INTMUX module has 8 channels that are assigned to 8 NVIC interrupt slots.
- Each INTMUX channel can provide up to 32 additional interrupt sources, which can be logically ORed or ANDed.
- The Interrupt Multiplexer (INTMUX) routes the interrupt sources to the interrupt outputs.

### 36.1.2 Features

INTMUX features:

- Supports 8 multiplex channels
- Each channel receives 32 interrupt sources and has 1 interrupt output
- Each interrupt source can be enabled or disabled
- Each channel supports *Logic AND* or *Logic OR* of all enabled interrupt sources

### 36.1.3 Block diagram

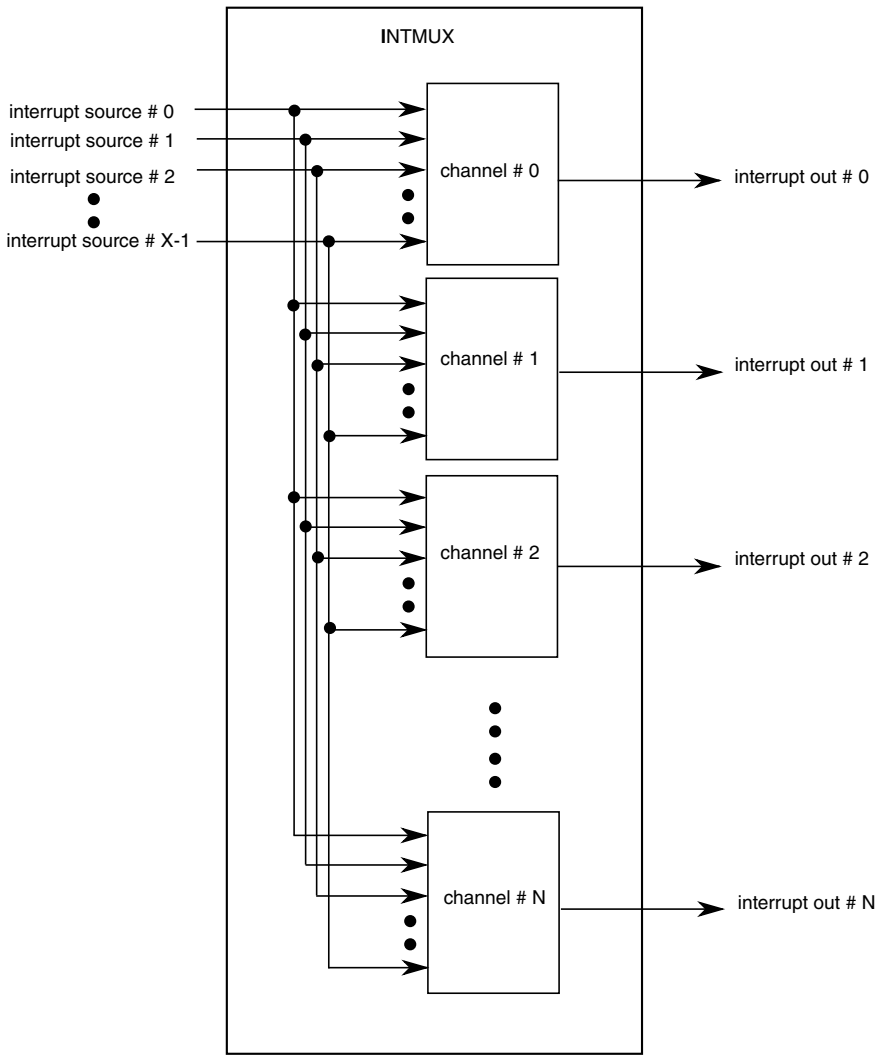



**Figure 36-1. INTMUX Block diagram**

N: Interrupt Channel Instance Number (N=7)

X: Interrupt Source Number for each channel (X=32)

## 36.2 Memory Map and register definition

This section includes the module memory map and detailed descriptions of all registers.

## 36.2.1 INTMUX0 register descriptions

### 36.2.1.1 INTMUX0 Memory map

INTMUX0 base address: 4004_F000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Channel n Control Status Register (CH0_CSR) | 32 | RW | 0000_0800h |
| 4h | Channel n Vector Number Register (CH0_VEC) | 32 | RO | 0000_0000h |
| 10h | Channel n Interrupt Enable Register (CH0_IER_31_0) | 32 | RW | 0000_0000h |
| 20h | Channel n Interrupt Pending Register (CH0_IPR_31_0) | 32 | RO | 0000_0000h |
| 40h | Channel n Control Status Register (CH1_CSR) | 32 | RW | 0000_0800h |
| 44h | Channel n Vector Number Register (CH1_VEC) | 32 | RO | 0000_0000h |
| 50h | Channel n Interrupt Enable Register (CH1_IER_31_0) | 32 | RW | 0000_0000h |
| 60h | Channel n Interrupt Pending Register (CH1_IPR_31_0) | 32 | RO | 0000_0000h |
| 80h | Channel n Control Status Register (CH2_CSR) | 32 | RW | 0000_0800h |
| 84h | Channel n Vector Number Register (CH2_VEC) | 32 | RO | 0000_0000h |
| 90h | Channel n Interrupt Enable Register (CH2_IER_31_0) | 32 | RW | 0000_0000h |
| A0h | Channel n Interrupt Pending Register (CH2_IPR_31_0) | 32 | RO | 0000_0000h |
| C0h | Channel n Control Status Register (CH3_CSR) | 32 | RW | 0000_0800h |
| C4h | Channel n Vector Number Register (CH3_VEC) | 32 | RO | 0000_0000h |
| D0h | Channel n Interrupt Enable Register (CH3_IER_31_0) | 32 | RW | 0000_0000h |
| E0h | Channel n Interrupt Pending Register (CH3_IPR_31_0) | 32 | RO | 0000_0000h |
| 100h | Channel n Control Status Register (CH4_CSR) | 32 | RW | 0000_0800h |
| 104h | Channel n Vector Number Register (CH4_VEC) | 32 | RO | 0000_0000h |
| 110h | Channel n Interrupt Enable Register (CH4_IER_31_0) | 32 | RW | 0000_0000h |
| 120h | Channel n Interrupt Pending Register (CH4_IPR_31_0) | 32 | RO | 0000_0000h |
| 140h | Channel n Control Status Register (CH5_CSR) | 32 | RW | 0000_0800h |
| 144h | Channel n Vector Number Register (CH5_VEC) | 32 | RO | 0000_0000h |
| 150h | Channel n Interrupt Enable Register (CH5_IER_31_0) | 32 | RW | 0000_0000h |
| 160h | Channel n Interrupt Pending Register (CH5_IPR_31_0) | 32 | RO | 0000_0000h |
| 180h | Channel n Control Status Register (CH6_CSR) | 32 | RW | 0000_0800h |
| 184h | Channel n Vector Number Register (CH6_VEC) | 32 | RO | 0000_0000h |
| 190h | Channel n Interrupt Enable Register (CH6_IER_31_0) | 32 | RW | 0000_0000h |
| 1A0h | Channel n Interrupt Pending Register (CH6_IPR_31_0) | 32 | RO | 0000_0000h |
| 1C0h | Channel n Control Status Register (CH7_CSR) | 32 | RW | 0000_0800h |
| 1C4h | Channel n Vector Number Register (CH7_VEC) | 32 | RO | 0000_0000h |
| 1D0h | Channel n Interrupt Enable Register (CH7_IER_31_0) | 32 | RW | 0000_0000h |
| 1E0h | Channel n Interrupt Pending Register (CH7_IPR_31_0) | 32 | RO | 0000_0000h |

## 36.2.1.2 Channel n Control Status Register (CH0_CSR - CH7_CSR)

### 36.2.1.2.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_CSR | 0h + (a × 40h) |

### 36.2.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQP | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CHIN | | | | 0 | | IRQN | | 0 | | AND | RST |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1[1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. **NOTE**
The reset value of CHIN depends on the channel number.

### 36.2.1.2.3 Fields

| Field | Function |
|---|---|
| 31<br>IRQP | Channel Interrupt Request Pending<br>This bit is available for polling purposes and represents the channel output.<br>0b - No interrupt is pending.<br>1b - The interrupt output of this channel is pending. |
| 30-12<br>— | Reserved |
| 11-8<br>CHIN | Channel Instance Number<br>These bits represent the instance number of this channel. |
| 7-6<br>— | Reserved |
| 5-4 | Channel Input Number |

*Table continues on the next page...*

| Field | Function |
|---|---|
| IRQN | These bits represent the interrupt input number of this channel.<br>00b - 32 interrupt inputs<br>01b - Reserved<br>10b - Reserved<br>11b - Reserved |
| 3-2<br>— | Reserved |
| 1<br>AND | Logic AND<br>This bit used to Logic AND or Logic OR all enabled interrupt inputs of this channel and generate the interrput output of this channel.<br>0b - Logic OR all enabled interrupt inputs.<br>1b - Logic AND all enabled interrupt inputs. |
| 0<br>RST | Software Reset<br>This bit is used to software reset this channel. This bit always reads as 0.<br>0b - No operation.<br>1b - Perform a software reset on this channel. |

## 36.2.1.3 Channel n Vector Number Register (CH0_VEC - CH7_VEC)

### 36.2.1.3.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_VEC | 4h + (a × 40h) |

### 36.2.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | VECN | | | | | | | | | | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 36.2.1.3.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-2<br>VECN | Vector Number<br>These bits represent the vector number for the highest priority interrupt source of this channel. VECN[13:2] = 48 ×(CPU vectors + NVIC vectors) + Number of the highest priority interrupt source of this channel.<br>**NOTE:** The interrupt priority is based on the Interrupt Pending Register. The smaller the interrupt source number, the higher the priority. |
| 1-0<br>— | Reserved |

### 36.2.1.4 Channel n Interrupt Enable Register (CH0_IER_31_0 - CH7_IER_31_0)

#### 36.2.1.4.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_IER_31_0 | 10h + (a × 40h) |

#### 36.2.1.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INTE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INTE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 36.2.1.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>INTE | Interrupt Enable<br>These bits are used to enable the interrupt sources of this channel. 0b: Interrupt is disabled. 1b: Interrupt is enabled. |

### 36.2.1.5 Channel n Interrupt Pending Register (CH0_IPR_31_0 - CH7_IPR_31_0)

#### 36.2.1.5.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_IPR_31_0 | 20h + (a × 40h) |

#### 36.2.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 36.2.1.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>INTP | Interrupt Pending<br>These bits represent the interrupt source pending status of this channel. 0b: No interrupt. 1b: Interrupt is pending. |

## 36.2.2 INTMUX1 register descriptions

### 36.2.2.1 INTMUX1 Memory map

INTMUX1 base address: 4102_2000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Channel n Control Status Register (CH0_CSR) | 32 | RW | 0000_0800h |
| 4h | Channel n Vector Number Register (CH0_VEC) | 32 | RO | 0000_0000h |
| 10h | Channel n Interrupt Enable Register (CH0_IER_31_0) | 32 | RW | 0000_0000h |
| 20h | Channel n Interrupt Pending Register (CH0_IPR_31_0) | 32 | RO | 0000_0000h |
| 40h | Channel n Control Status Register (CH1_CSR) | 32 | RW | 0000_0800h |
| 44h | Channel n Vector Number Register (CH1_VEC) | 32 | RO | 0000_0000h |
| 50h | Channel n Interrupt Enable Register (CH1_IER_31_0) | 32 | RW | 0000_0000h |
| 60h | Channel n Interrupt Pending Register (CH1_IPR_31_0) | 32 | RO | 0000_0000h |
| 80h | Channel n Control Status Register (CH2_CSR) | 32 | RW | 0000_0800h |
| 84h | Channel n Vector Number Register (CH2_VEC) | 32 | RO | 0000_0000h |
| 90h | Channel n Interrupt Enable Register (CH2_IER_31_0) | 32 | RW | 0000_0000h |
| A0h | Channel n Interrupt Pending Register (CH2_IPR_31_0) | 32 | RO | 0000_0000h |
| C0h | Channel n Control Status Register (CH3_CSR) | 32 | RW | 0000_0800h |
| C4h | Channel n Vector Number Register (CH3_VEC) | 32 | RO | 0000_0000h |
| D0h | Channel n Interrupt Enable Register (CH3_IER_31_0) | 32 | RW | 0000_0000h |
| E0h | Channel n Interrupt Pending Register (CH3_IPR_31_0) | 32 | RO | 0000_0000h |
| 100h | Channel n Control Status Register (CH4_CSR) | 32 | RW | 0000_0800h |
| 104h | Channel n Vector Number Register (CH4_VEC) | 32 | RO | 0000_0000h |
| 110h | Channel n Interrupt Enable Register (CH4_IER_31_0) | 32 | RW | 0000_0000h |
| 120h | Channel n Interrupt Pending Register (CH4_IPR_31_0) | 32 | RO | 0000_0000h |
| 140h | Channel n Control Status Register (CH5_CSR) | 32 | RW | 0000_0800h |
| 144h | Channel n Vector Number Register (CH5_VEC) | 32 | RO | 0000_0000h |
| 150h | Channel n Interrupt Enable Register (CH5_IER_31_0) | 32 | RW | 0000_0000h |
| 160h | Channel n Interrupt Pending Register (CH5_IPR_31_0) | 32 | RO | 0000_0000h |
| 180h | Channel n Control Status Register (CH6_CSR) | 32 | RW | 0000_0800h |
| 184h | Channel n Vector Number Register (CH6_VEC) | 32 | RO | 0000_0000h |
| 190h | Channel n Interrupt Enable Register (CH6_IER_31_0) | 32 | RW | 0000_0000h |
| 1A0h | Channel n Interrupt Pending Register (CH6_IPR_31_0) | 32 | RO | 0000_0000h |
| 1C0h | Channel n Control Status Register (CH7_CSR) | 32 | RW | 0000_0800h |
| 1C4h | Channel n Vector Number Register (CH7_VEC) | 32 | RO | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1D0h | Channel n Interrupt Enable Register (CH7_IER_31_0) | 32 | RW | 0000_0000h |
| 1E0h | Channel n Interrupt Pending Register (CH7_IPR_31_0) | 32 | RO | 0000_0000h |

## 36.2.2.2 Channel n Control Status Register (CH0_CSR - CH7_CSR)

### 36.2.2.2.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_CSR | 0h + (a × 40h) |

### 36.2.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IRQP | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CHIN | | | | 0 | | IRQN | | 0 | | AND | RST |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1[1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. **NOTE**
The reset value of CHIN depends on the channel number.

### 36.2.2.2.3 Fields

| Field | Function |
|---|---|
| 31<br>IRQP | Channel Interrupt Request Pending<br>This bit is available for polling purposes and represents the channel output.<br>0b - No interrupt is pending.<br>1b - The interrupt output of this channel is pending. |
| 30-12 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 11-8<br>CHIN | Channel Instance Number<br>These bits represent the instance number of this channel. |
| 7-6<br>— | Reserved |
| 5-4<br>IRQN | Channel Input Number<br>These bits represent the interrupt input number of this channel.<br>00b - 32 interrupt inputs<br>01b - Reserved<br>10b - Reserved<br>11b - Reserved |
| 3-2<br>— | Reserved |
| 1<br>AND | Logic AND<br>This bit used to Logic AND or Logic OR all enabled interrupt inputs of this channel and generate the interrput output of this channel.<br>0b - Logic OR all enabled interrupt inputs.<br>1b - Logic AND all enabled interrupt inputs. |
| 0<br>RST | Software Reset<br>This bit is used to software reset this channel. This bit always reads as 0.<br>0b - No operation.<br>1b - Perform a software reset on this channel. |

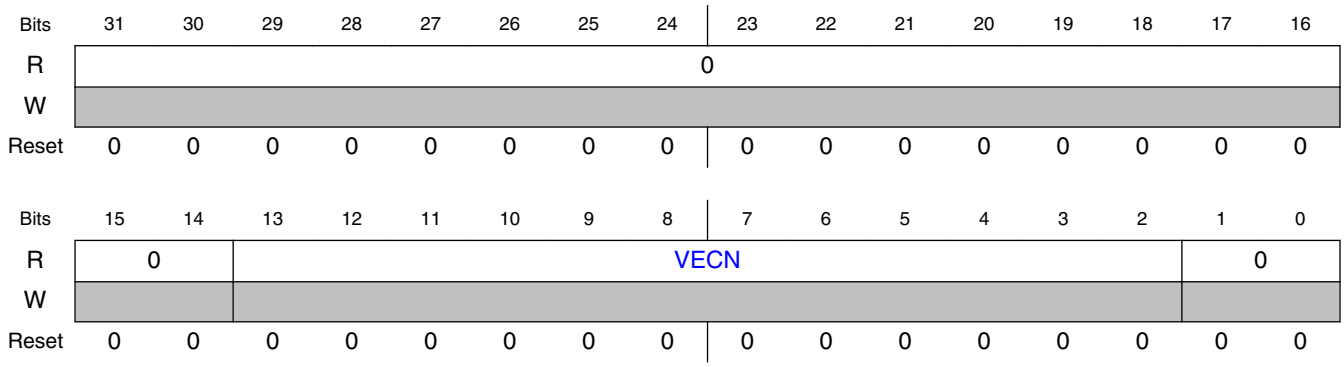
## 36.2.2.3 Channel n Vector Number Register (CH0_VEC - CH7_VEC)

### 36.2.2.3.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_VEC | 4h + (a × 40h) |

#### 36.2.2.3.2 Diagram




#### 36.2.2.3.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-2<br>VECN | Vector Number<br>These bits represent the vector number for the highest priority interrupt source of this channel. VECN[13:2] = 48 ×(CPU vectors + NVIC vectors) + Number of the highest priority interrupt source of this channel.<br>**NOTE:** The interrupt priority is based on the Interrupt Pending Register. The smaller the interrupt source number, the higher the priority. |
| 1-0<br>— | Reserved |

### 36.2.2.4 Channel n Interrupt Enable Register (CH0_IER_31_0 - CH7_IER_31_0)

#### 36.2.2.4.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_IER_31_0 | 10h + (a × 40h) |

### 36.2.2.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INTE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INTE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 36.2.2.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>INTE | Interrupt Enable<br>These bits are used to enable the interrupt sources of this channel. 0b: Interrupt is disabled. 1b: Interrupt is enabled. |

## 36.2.2.5 Channel n Interrupt Pending Register (CH0_IPR_31_0 - CH7_IPR_31_0)

### 36.2.2.5.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHa_IPR_31_0 | 20h + (a × 40h) |

#### 36.2.2.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 36.2.2.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>INTP | Interrupt Pending<br>These bits represent the interrupt source pending status of this channel. 0b: No interrupt. 1b: Interrupt is pending. |

## 36.3 Functional Description

The Interrupt Multiplexer (INTMUX) routes the interrupt sources to the interrupt outputs.

There is one interrupt output for each channel available in the INTMUX. All interrupt sources are available to each channel. There are 8 multiplex channels, and there are 32 interrupt sources.

The INTMUX module provides interrupt status registers to monitor interrupt pending status and vector numbers. This module also implements the ability to logical AND or OR enabled interrupts on a given channel. Software resets can also be performed on a given channel.

See the chip specific Interrupt section for more information regarding the interrupt vectors and sources.

### 36.3.1 Configuring Output Channels

There is a single set of 32 source interrupts for the INTMUX. Each of the 8 output channels can be configured to output any combination of the source interrupts.

To enable source interrupts on a channel, write a mask value to the Channel n Interrupt Enable Register (CHn_IER_31_0). Each bit of these registers corresponds to one of the 32 source interrupts. Bit n enables source interrupt n on the given channel.

Further control for each output channel is available by selecting the Boolean operation to perform on pending interrupts. The default is logical OR, but logical AND can be selected via the CHn_CSR[AND] register bit.

### 36.3.2 INTMUX Vectors

The INTMUX is designed so that it integrates with the standard NVIC vector table. Each INTMUX source interrupt has a vector associated with it. These source vectors are intended to immediately follow the NVIC vectors in the vector table pointed to by the CPU's VTOR register. Interrupt service routine addresses can be placed in the INTMUX source vectors, just like NVIC vectors.

The ISR for each of the INTMUX output channels should read the CHn_VEC register to determine the highest priority active source interrupt out of those enabled for the given channel. The CHn_VEC[VECN] bitfield returns the source interrupt number plus 48, which accounts for the number of CPU and NVIC vectors. The VECN field starts at bit 2 of the CHn_VEC register, so reading the register as a whole will return the VECN field multiplied by 4. This is the offset in bytes of the active source vector from the start of the vector table (i.e., VTOR).

For instance, consider if source interrupt 12 was enabled on output channel 1 and became active. The ISR for channel 1 would read the entire CH1_VEC register to get the active vector offset of 240 (i.e., (48 + 12) × 4). It then adds this value to the value of VTOR. The word at the resulting address is the address of the active interrupt's ISR, which can then be invoked.

If multiple source interrupts become pending simultaneously on a single channel, the CHn_VEC[VECN] field will read as the highest priority vector number. Priority of source interrupts is determined by the source interrupt number. Lower numbers are higher priortity, with source interrupt 0 being highest priority.

**NOTE**

Unlike the NVIC, the INTMUX does not latch pending source interrupts. This means that the INTMUX output channel ISRs must check for and handle a 0 value of the CHn_VEC register to account for spurious interrupts.

# Chapter 37
# Trigger MUX (TRGMUX)

## 37.1 Chip-specific TRGMUX information

### 37.1.1 CPU0 Trigger Multiplexer input sources

The trigger multiplexer allows triggers from one module to initiate operations on other modules. The following table shows the inputs to the trigger mutliplexer instantiated on the CPU0 crossbar.

**Table 37-1. CPU0 TRGMUX Inputs**

| Mux Select | Source Module | Source Function |
|---|---|---|
| 0 | — | Disabled |
| 1 | LLWU0 | Trigger Event |
| 2 | LPIT0 | Channel 0 |
| 3 | LPIT0 | Channel 1 |
| 4 | LPIT0 | Channel 2 |
| 5 | LPIT0 | Channel 3 |
| 6 | RTC | Alarm |
| 7 | RTC | Seconds |
| 8 | LPTMR0 | Trigger |
| 9 | LPTMR1 | Trigger |
| 10 | TPM0 | Channels (Even) |
| 11 | TPM0 | Channels (Odd) |
| 12 | TPM0 | Overflow |
| 13 | TPM1 | Channels (Even) |
| 14 | TPM1 | Channels (Odd) |
| 15 | TPM1 | Overflow |
| 16 | TPM2 | Channels (Even) |
| 17 | TPM2 | Channels (Odd) |

*Table continues on the next page...*

## Table 37-1. CPU0 TRGMUX Inputs (continued)

| Mux Select | Source Module | Source Function |
|---|---|---|
| 18 | TPM2 | Overflow |
| 19 | FlexIO0 | Timer 0 |
| 20 | FlexIO0 | Timer 1 |
| 21 | FlexIO0 | Timer 2 |
| 22 | FlexIO0 | Timer 3 |
| 23 | FlexIO0 | Timer 4 |
| 24 | FlexIO0 | Timer 5 |
| 25 | FlexIO0 | Timer 6 |
| 26 | FlexIO0 | Timer 7 |
| 27 | LPI2C0 | Master STOP |
| 28 | LPI2C0 | Slave STOP |
| 29 | LPI2C1 | Master STOP |
| 30 | LPI2C1 | Slave STOP |
| 31 | LPI2C2 | Master STOP |
| 32 | LPI2C2 | Slave STOP |
| 33 | SAI0 | RX Frame Sync |
| 34 | SAI0 | TX Frame Sync |
| 35 | LPSPI0 | Frame |
| 36 | LPSPI0 | RX Data |
| 37 | LPSPI1 | Frame |
| 38 | LPSPI1 | RX Data |
| 39 | LPSPI2 | Frame |
| 40 | LPSPI2 | RX Data |
| 41 | LPUART0 | RX Data |
| 42 | LPUART0 | RX Idle |
| 43 | LPUART0 | TX Data |
| 44 | LPUART1 | RX Data |
| 45 | LPUART1 | RX Idle |
| 46 | LPUART1 | TX Data |
| 47 | LPUART2 | RX Data |
| 48 | LPUART2 | RX Idle |
| 49 | LPUART2 | TX Data |
| 50 | USB0 | Start of Frame |
| 51 | PORTA | Pin Trigger |
| 52 | PORTB | Pin Trigger |
| 53 | PORTC | Pin Trigger |
| 54 | PORTD | Pin Trigger |
| 55 | LPCMP0 | Output |
| 56 | LPI2C3 | Master STOP |

*Table continues on the next page...*

**Table 37-1. CPU0 TRGMUX Inputs (continued)**

| Mux Select | Source Module | Source Function |
|---|---|---|
| 57 | LPI2C3 | Slave STOP |
| 58 | LPSPI3 | Frame |
| 59 | LPSPI3 | RX Data |
| 60 | LPUART3 | RX Data |
| 61 | LPUART3 | RX Idle |
| 62 | LPUART3 | TX Data |
| 63 | PORTE | Pin Trigger |

## 37.1.2 CPU0 Trigger Multiplexer outputs

The trigger multiplexer allows triggers from one module to initiate operations on other modules. The following table shows the outputs from the trigger mutliplexer instantiated on the CPU0 crossbar.

**Table 37-2. CPU0 TRGMUX Outputs**

| Destination Module | Destination Function | Number of Muxes |
|---|---|---|
| DMA MUX0 | Periodic Trigger | 4 |
| LPIT0 | Trigger | 4 |
| TPM0 | Input Capture, Trigger, Output Modulation | 3 |
| TPM1 | Input Capture, Trigger, Output Modulation | 3 |
| TPM2 | Input Capture, Trigger, Output Modulation | 3 |
| FlexIO0 | Trigger | 4 |
| LPI2C0 | Host Request | 1 |
| LPI2C1 | Host Request | 1 |
| LPI2C2 | Host Request | 1 |
| LPSPI0 | Host Request | 1 |
| LPSPI1 | Host Request | 1 |
| LPSPI2 | Host Request | 1 |
| LPUART0 | RX Input, CTS Input, Modulation | 1 |
| LPUART1 | RX Input, CTS Input, Modulation | 1 |
| LPUART2 | RX Input, CTS Input, Modulation | 1 |
| LPADC0 | Trigger | 4 |
| LPCMP0 | Window, Trigger | 1 |
| LPDAC0 | Trigger | 1 |
| DMA MUX1 | Periodic Trigger | 4 |
| LPIT1 | Trigger | 4 |
| TPM3 | Input Capture, Trigger, Output Modulation | 3 |

*Table continues on the next page...*

**Table 37-2. CPU0 TRGMUX Outputs (continued)**

| Destination Module | Destination Function | Number of Muxes |
|---|---|---|
| LPI2C3 | Host Request | 1 |
| LPSPI3 | Host Request | 1 |
| LPUART3 | RX Input, CTS Input, Modulation | 1 |
| LPCMP1 | Window, Trigger | 1 |

## 37.1.3 CPU1 Trigger Multiplexer input sources

The trigger multiplexer allows triggers from one module to initiate operations on other modules. The following table shows the inputs to the trigger mutliplexer instantiated on the CPU1 crossbar.

**Table 37-3. CPU1 TRGMUX Inputs**

| Mux Select | Source Module | Source Function |
|---|---|---|
| 0 | — | Disabled |
| 1 | LLWU1 | Trigger Event |
| 2 | LPIT1 | Channel 0 |
| 3 | LPIT1 | Channel 1 |
| 4 | LPIT1 | Channel 2 |
| 5 | LPIT1 | Channel 3 |
| 6 | LPTMR2 | Trigger |
| 7 | TPM3 | Channels (Even) |
| 8 | TPM3 | Channels (Odd) |
| 9 | TPM3 | Overflow |
| 10 | LPI2C3 | Master STOP |
| 11 | LPI2C3 | Slave STOP |
| 12 | LPSPI3 | Frame |
| 13 | LPSPI3 | RX Data |
| 14 | LPUART3 | RX Data |
| 15 | LPUART3 | RX Idle |
| 16 | LPUART3 | TX Data |
| 17 | PORTE | Pin Trigger |
| 18 | LPCMP1 | Output |
| 19 | RTC | Alarm |
| 20 | RTC | Seconds |
| 21 | LPTMR0 | Trigger |
| 22 | LPTMR1 | Trigger |
| 23 | TPM1 | Channels (Even) |

*Table continues on the next page...*

## Table 37-3. CPU1 TRGMUX Inputs (continued)

| Mux Select | Source Module | Source Function |
|---|---|---|
| 24 | TPM1 | Channels (Odd) |
| 25 | TPM1 | Overflow |
| 26 | TPM2 | Channels (Even) |
| 27 | TPM2 | Channels (Odd) |
| 28 | TPM2 | Overflow |
| 29 | FlexIO0 | Timer 0 |
| 30 | FlexIO0 | Timer 1 |
| 31 | FlexIO0 | Timer 2 |
| 32 | FlexIO0 | Timer 3 |
| 33 | FlexIO0 | Timer 4 |
| 34 | FlexIO0 | Timer 5 |
| 35 | FlexIO0 | Timer 6 |
| 36 | FlexIO0 | Timer 7 |
| 37 | LPI2C0 | Master STOP |
| 38 | LPI2C0 | Slave STOP |
| 39 | LPI2C1 | Master STOP |
| 40 | LPI2C1 | Slave STOP |
| 41 | LPI2C2 | Master STOP |
| 42 | LPI2C2 | Slave STOP |
| 43 | SAI0 | RX Frame Sync |
| 44 | SAI0 | TX Frame Sync |
| 45 | LPSPI0 | Frame |
| 46 | LPSPI0 | RX Data |
| 47 | LPSPI1 | Frame |
| 48 | LPSPI1 | RX Data |
| 49 | LPSPI2 | Frame |
| 50 | LPSPI2 | RX Data |
| 51 | LPUART0 | RX Data |
| 52 | LPUART0 | RX Idle |
| 53 | LPUART0 | TX Data |
| 54 | LPUART1 | RX Data |
| 55 | LPUART1 | RX Idle |
| 56 | LPUART1 | TX Data |
| 57 | LPUART2 | RX Data |
| 58 | LPUART2 | RX Idle |
| 59 | LPUART2 | TX Data |
| 60 | PORTA | Pin Trigger |
| 61 | PORTB | Pin Trigger |
| 62 | PORTC | Pin Trigger |

*Table continues on the next page...*

**Table 37-3. CPU1 TRGMUX Inputs (continued)**

| Mux Select | Source Module | Source Function |
|---|---|---|
| 63 | PORTD | Pin Trigger |

## 37.1.4 CPU1 Trigger Multiplexer outputs

The trigger multiplexer allows triggers from one module to initiate operations on other modules. The following table shows the outputs from the trigger mutliplexer instantiated on the CPU1 crossbar.

**Table 37-4. CPU1 TRGMUX Outputs**

| Destination Module | Destination Function | Number of Muxes |
|---|---|---|
| DMA MUX1 | Periodic Trigger | 4 |
| LPIT1 | Trigger | 4 |
| TPM3 | Input Capture, Trigger, Output Modulation | 3 |
| LPI2C3 | Host Request | 1 |
| LPSPI3 | Host Request | 1 |
| LPUART3 | RX Input, CTS Input, Modulation | 1 |
| LPCMP1 | Window, Trigger | 1 |
| DMA MUX0 | Periodic Trigger | 4 |
| LPIT0 | Trigger | 4 |
| TPM0 | Input Capture, Trigger, Output Modulation | 3 |
| TPM1 | Input Capture, Trigger, Output Modulation | 3 |
| TPM2 | Input Capture, Trigger, Output Modulation | 3 |
| FlexIO0 | Trigger | 4 |
| LPI2C0 | Host Request | 1 |
| LPI2C1 | Host Request | 1 |
| LPI2C2 | Host Request | 1 |
| LPSPI0 | Host Request | 1 |
| LPSPI1 | Host Request | 1 |
| LPSPI2 | Host Request | 1 |
| LPUART0 | RX Input, CTS Input, Modulation | 1 |
| LPUART1 | RX Input, CTS Input, Modulation | 1 |
| LPUART2 | RX Input, CTS Input, Modulation | 1 |
| LPADC0 | Trigger | 4 |
| LPCMP0 | Window, Trigger | 1 |
| LPDAC0 | Trigger | 1 |

## 37.2 Introduction

The trigger multiplexer (TRGMUX) module allows software to configure the trigger inputs for various peripherals.

## 37.3 Features

The TRGMUX module allows software to select the trigger source for peripherals. The block diagram below shows the trigger selection logic of the TRGMUX module.

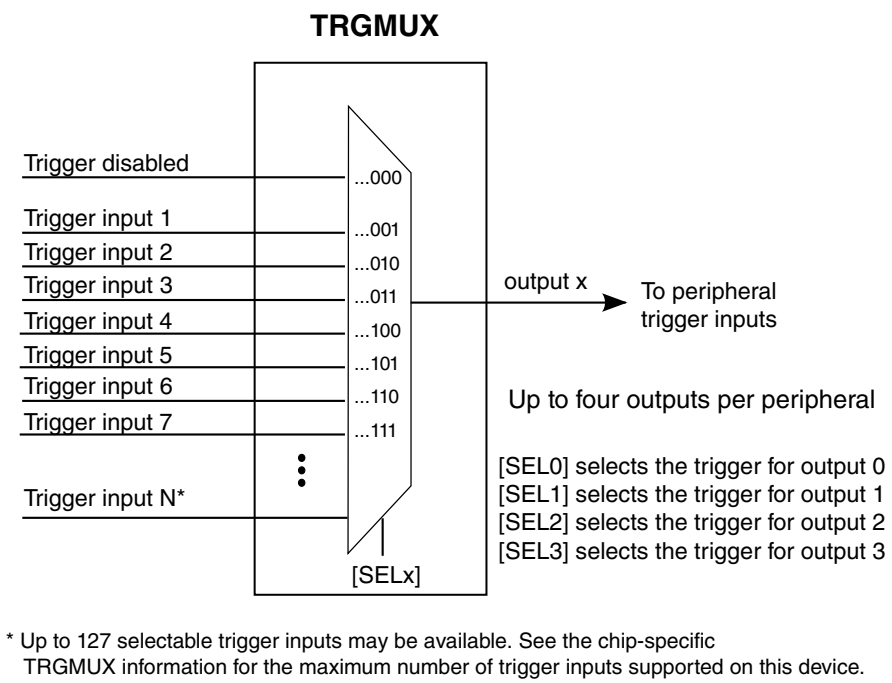



**Figure 37-1. TRGMUX block diagram**

Each peripheral has its own dedicated TRGMUX register. See each peripheral's TRGMUX register for details.

## 37.4 Memory map and register definition

The TRGMUX registers contain fields for selecting trigger sources for peripheral modules.

TRGMUX registers can be written only in supervisor mode.

## 37.4.1 TRGMUX0 register descriptions

### 37.4.1.1 TRGMUX0 Memory map

**Table 37-5. Select Bit Fields**

| Field | Function |
|---|---|
| SELx | This read/write field is used to configure the MUX select for the peripheral trigger inputs.<br>000_0000 - (0x00) Trigger function is disabled<br>000_0001 - (0x01) LLWU0 Trigger event input is selected<br>000_0010 - (0x02) LPIT0 Channel 0 input is selected<br>000_0011 - (0x03) LPIT0 Channel 1 input is selected<br>000_0100 - (0x04) LPIT0 Channel 2 input is selected<br>000_0101 - (0x05) LPIT0 Channel 3 input is selected<br>000_0110 - (0x06) RTC Alarm input is selected<br>000_0111 - (0x07) RTC Seconds input is selected<br>000_1000 - (0x08) LPTMR0 Trigger input is selected<br>000_1001 - (0x09) LPTMR1 Trigger input is selected<br>000_1010 - (0x0A) TPM0 Channel (Even) input is selected<br>000_1011 - (0x0B) TPM0 Channel (Odd) input is selected<br>000_1100 - (0x0C) TPM0 Overflow input is selected<br>000_1101 - (0x0D) TPM1 Channel (Even) input is selected<br>000_1110 - (0x0E) TPM1 Channel (Odd) input is selected<br>000_1111 - (0x0F) TPM1 Overflow input is selected<br>001_0000 - (0x10) TPM2 Channel (Even) input is selected<br>001_0001 - (0x11) TPM2 Channel (Odd) input is selected<br>001_0010 - (0x12) TPM2 Overflow input is selected<br>001_0011 - (0x13) FLEXIO0 Time 0 input is selected<br>001_0100 - (0x14) FLEXIO0 Time 1 input is selected<br>001_0101 - (0x15) FLEXIO0 Time 2 input is selected<br>001_0110 - (0x16) FLEXIO0 Time 3 input is selected<br>001_0111 - (0x17) FLEXIO0 Time 4 input is selected<br>001_1000 - (0x18) FLEXIO0 Time 5 input is selected<br>001_1001 - (0x19) FLEXIO0 Time 6 input is selected<br>001_1010 - (0x1A) FLEXIO0 Time 7 input is selected<br>001_1011 - (0x1B) LPI2C0 Master STOP input is selected<br>001_1100 - (0x1C) LPI2C0 Slave STOP input is selected<br>001_1101 - (0x1D) LPI2C1 Master STOP input is selected |

**Table 37-5. Select Bit Fields**

| Field | Function |
|---|---|
| | 001_1110 - (0x1E) LPI2C1 Slave STOP input is selected |
| | 001_1111 - (0x1F) LPI2C2 Master STOP input is selected |
| | 010_0000 - (0x20) LPI2C2 Slave STOP input is selected |
| | 010_0001 - (0x21) I2S0 RX Frame Sync input is selected |
| | 010_0010 - (0x22) I2S0 TX Frame Sync input is selected |
| | 010_0011 - (0x23) LPSPI0 Frame input is selected |
| | 010_0100 - (0x24) LPSPI0 RX data input is selected |
| | 010_0101 - (0x25) LPSPI1 Frame input is selected |
| | 010_0110 - (0x26) LPSPI1 RX data input is selected |
| | 010_0111 - (0x27) LPSPI2 Frame input is selected |
| | 010_1000 - (0x28) LPSPI2 RX data input is selected |
| | 010_1001 - (0x29) LPUART0 RX data input is selected |
| | 010_1010 - (0x2A) LPUART0 RX Idle input is selected |
| | 010_1011 - (0x2B) LPUART0 TX data input is selected |
| | 010_1100 - (0x2C) LPUART1 RX data input is selected |
| | 010_1101 - (0x2D) LPUART1 RX Idle input is selected |
| | 010_1110 - (0x2E) LPUART1 TX data input is selected |
| | 010_1111 - (0x2F) LPUART2 RX data input is selected |
| | 011_0000 - (0x30) LPUART2 RX Idle input is selected |
| | 011_0001 - (0x31) LPUART2 TX data input is selected |
| | 011_0010 - (0x32) USB Start of Frame input is selected |
| | 011_0011 - (0x33) PORTA Pin Trigger input is selected |
| | 011_0100 - (0x34) PORTB Pin Trigger input is selected |
| | 011_0101 - (0x35) PORTC Pin Trigger input is selected |
| | 011_0110 - (0x36) PORTD Pin Trigger input is selected |
| | 011_0111 - (0x37) LPCMP0 Output input is selected |
| | 011_1000 - (0x38) LPI2C3 Master STOP input is selected |
| | 011_1001 - (0x39) LPI2C3 Slave STOP input is selected |
| | 011_1010 - (0x3A) LPSPI3 Frame input is selected |
| | 011_1011 - (0x3B) LPSPI3 RX data input is selected |
| | 011_1100 - (0x3C) LPUART3 RX data input is selected |
| | 011_1101 - (0x3D) LPUART3 RX Idle input is selected |
| | 011_1110 - (0x3E) LPUART3 TX data input is selected |
| | 011_1111 - (0x3F) PORTE Pin Trigger input is selected |

TRGMUX0 base address: 4002_9000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TRGMUX DMAMUX0 Register (DMAMUX0) | 32 | RW | 0000_0000h |
| 4h | TRGMUX LPIT0 Register (LPIT0) | 32 | RW | 0000_0000h |
| 8h | TRGMUX TPM0 Register (TPM0) | 32 | RW | 0000_0000h |
| Ch | TRGMUX TPM1 Register (TPM1) | 32 | RW | 0000_0000h |
| 10h | TRGMUX TPM2 Register (TPM2) | 32 | RW | 0000_0000h |
| 14h | TRGMUX FLEXIO0 Register (FLEXIO0) | 32 | RW | 0000_0000h |
| 18h | TRGMUX LPI2C0 Register (LPI2C0) | 32 | RW | 0000_0000h |
| 1Ch | TRGMUX LPI2C1 Register (LPI2C1) | 32 | RW | 0000_0000h |
| 20h | TRGMUX LPI2C2 Register (LPI2C2) | 32 | RW | 0000_0000h |
| 24h | TRGMUX LPSPI0 Register (LPSPI0) | 32 | RW | 0000_0000h |
| 28h | TRGMUX LPSPI1 Register (LPSPI1) | 32 | RW | 0000_0000h |
| 2Ch | TRGMUX LPSPI2 Register (LPSPI2) | 32 | RW | 0000_0000h |
| 30h | TRGMUX LPUART0 Register (LPUART0) | 32 | RW | 0000_0000h |
| 34h | TRGMUX LPUART1 Register (LPUART1) | 32 | RW | 0000_0000h |
| 38h | TRGMUX LPUART2 Register (LPUART2) | 32 | RW | 0000_0000h |
| 3Ch | TRGMUX LPADC0 Register (LPADC0) | 32 | RW | 0000_0000h |
| 40h | TRGMUX LPCMP0 Register (LPCMP0) | 32 | RW | 0000_0000h |
| 44h | TRGMUX LPDAC0 Register (LPDAC0) | 32 | RW | 0000_0000h |
| 48h | TRGMUX DMAMUX1 Register (DMAMUX1) | 32 | RW | 0000_0000h |
| 4Ch | TRGMUX LPIT1 Register (LPIT1) | 32 | RW | 0000_0000h |
| 50h | TRGMUX TPM3 Register (TPM3) | 32 | RW | 0000_0000h |
| 54h | TRGMUX LPI2C3 Register (LPI2C3) | 32 | RW | 0000_0000h |
| 58h | TRGMUX LPSPI3 Register (LPSPI3) | 32 | RW | 0000_0000h |
| 5Ch | TRGMUX LPUART3 Register (LPUART3) | 32 | RW | 0000_0000h |
| 60h | TRGMUX LPCMP1 Register (LPCMP1) | 32 | RW | 0000_0000h |

## 37.4.1.2 TRGMUX DMAMUX0 Register (DMAMUX0)

### 37.4.1.2.1 Offset

| Register | Offset |
|---|---|
| DMAMUX0 | 0h |

### 37.4.1.2.2 Function

TRGMUX Register

### 37.4.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.3 TRGMUX LPIT0 Register (LPIT0)

### 37.4.1.3.1 Offset

| Register | Offset |
|---|---|
| LPIT0 | 4h |

### 37.4.1.3.2 Function

TRGMUX Register

### 37.4.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.4 TRGMUX TPM0 Register (TPM0)

### 37.4.1.4.1 Offset

| Register | Offset |
|---|---|
| TPM0 | 8h |

### 37.4.1.4.2 Function

TRGMUX Register

### 37.4.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.5 TRGMUX TPM1 Register (TPM1)

### 37.4.1.5.1 Offset

| Register | Offset |
|---|---|
| TPM1 | Ch |

### 37.4.1.5.2 Function

TRGMUX Register

### 37.4.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.6 TRGMUX TPM2 Register (TPM2)

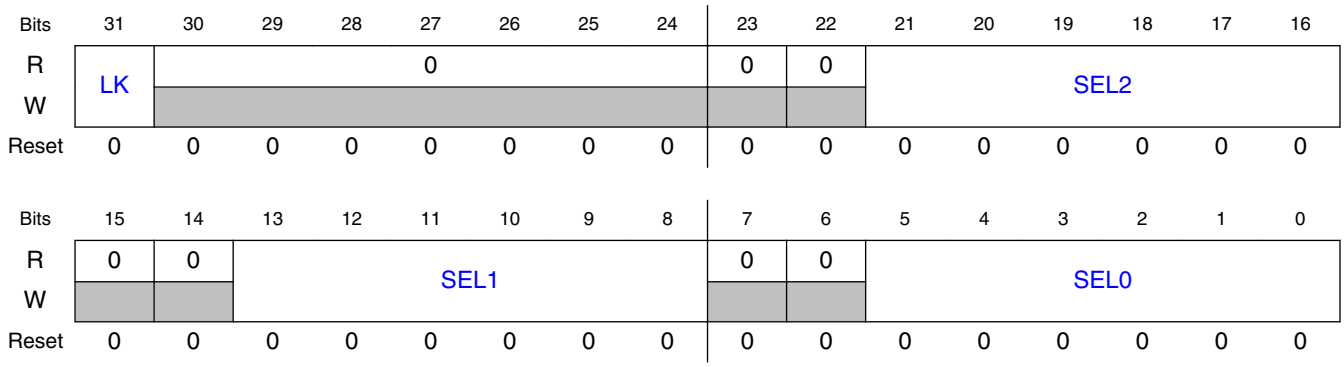
### 37.4.1.6.1 Offset

| Register | Offset |
|---|---|
| TPM2 | 10h |

### 37.4.1.6.2 Function

TRGMUX Register

### 37.4.1.6.3 Diagram




### 37.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0 | Trigger MUX Input 0 Source Select |

| Field | Function |
|---|---|
| SEL0 | This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.7 TRGMUX FLEXIO0 Register (FLEXIO0)

### 37.4.1.7.1 Offset

| Register | Offset |
|---|---|
| FLEXIO0 | 14h |

### 37.4.1.7.2 Function

TRGMUX Register

### 37.4.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.8 TRGMUX LPI2C0 Register (LPI2C0)

### 37.4.1.8.1 Offset

| Register | Offset |
|---|---|
| LPI2C0 | 18h |

### 37.4.1.8.2 Function

TRGMUX Register

## 37.4.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.8.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.1.9 TRGMUX LPI2C1 Register (LPI2C1)

## 37.4.1.9.1 Offset

| Register | Offset |
|---|---|
| LPI2C1 | 1Ch |

## 37.4.1.9.2 Function

TRGMUX Register

## 37.4.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.10 TRGMUX LPI2C2 Register (LPI2C2)

### 37.4.1.10.1 Offset

| Register | Offset |
|---|---|
| LPI2C2 | 20h |

### 37.4.1.10.2 Function

TRGMUX Register

### 37.4.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 37.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.11 TRGMUX LPSPI0 Register (LPSPI0)

### 37.4.1.11.1 Offset

| Register | Offset |
|---|---|
| LPSPI0 | 24h |

### 37.4.1.11.2 Function

TRGMUX Register

## 37.4.1.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.1.12 TRGMUX LPSPI1 Register (LPSPI1)

## 37.4.1.12.1 Offset

| Register | Offset |
|---|---|
| LPSPI1 | 28h |

## 37.4.1.12.2 Function

TRGMUX Register

## 37.4.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.12.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.13 TRGMUX LPSPI2 Register (LPSPI2)

### 37.4.1.13.1 Offset

| Register | Offset |
|---|---|
| LPSPI2 | 2Ch |

### 37.4.1.13.2 Function

TRGMUX Register

### 37.4.1.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | | | | 0 | | | | 0 | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | 0 | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.13.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.14 TRGMUX LPUART0 Register (LPUART0)

### 37.4.1.14.1 Offset

| Register | Offset |
|---|---|
| LPUART0 | 30h |

### 37.4.1.14.2 Function

TRGMUX Register

## 37.4.1.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.14.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.1.15 TRGMUX LPUART1 Register (LPUART1)

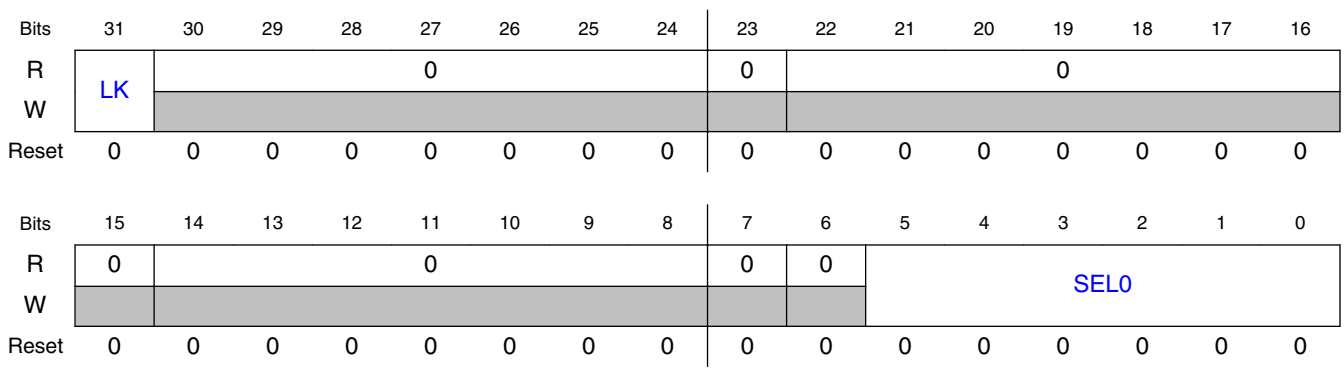
## 37.4.1.15.1 Offset

| Register | Offset |
|---|---|
| LPUART1 | 34h |

## 37.4.1.15.2 Function

TRGMUX Register

## 37.4.1.15.3 Diagram




## 37.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.16 TRGMUX LPUART2 Register (LPUART2)

### 37.4.1.16.1 Offset

| Register | Offset |
|---|---|
| LPUART2 | 38h |

### 37.4.1.16.2 Function

TRGMUX Register

### 37.4.1.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 37.4.1.16.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.17 TRGMUX LPADC0 Register (LPADC0)

### 37.4.1.17.1 Offset

| Register | Offset |
|---|---|
| LPADC0 | 3Ch |

### 37.4.1.17.2 Function

TRGMUX Register

## 37.4.1.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.17.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.18 TRGMUX LPCMP0 Register (LPCMP0)

### 37.4.1.18.1 Offset

| Register | Offset |
|---|---|
| LPCMP0 | 40h |

### 37.4.1.18.2 Function

TRGMUX Register

### 37.4.1.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.18.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.19 TRGMUX LPDAC0 Register (LPDAC0)

### 37.4.1.19.1 Offset

| Register | Offset |
|---|---|
| LPDAC0 | 44h |

### 37.4.1.19.2 Function

TRGMUX Register

## 37.4.1.19.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.19.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.1.20 TRGMUX DMAMUX1 Register (DMAMUX1)

## 37.4.1.20.1 Offset

| Register | Offset |
|---|---|
| DMAMUX1 | 48h |

## 37.4.1.20.2 Function

TRGMUX Register

## 37.4.1.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.20.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.21 TRGMUX LPIT1 Register (LPIT1)

### 37.4.1.21.1 Offset

| Register | Offset |
|---|---|
| LPIT1 | 4Ch |

### 37.4.1.21.2 Function

TRGMUX Register

## 37.4.1.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.21.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.22 TRGMUX TPM3 Register (TPM3)

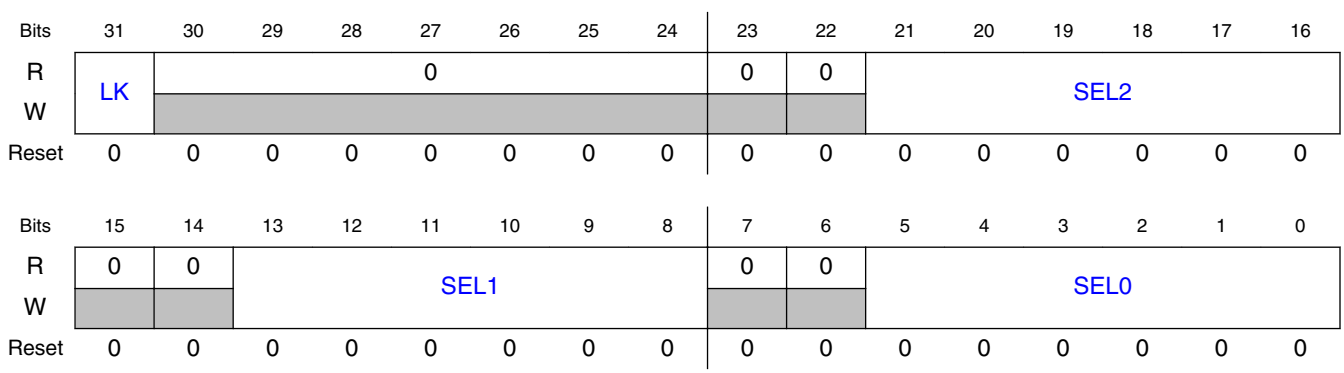
### 37.4.1.22.1 Offset

| Register | Offset |
|---|---|
| TPM3 | 50h |

### 37.4.1.22.2 Function

TRGMUX Register

### 37.4.1.22.3 Diagram




### 37.4.1.22.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.23 TRGMUX LPI2C3 Register (LPI2C3)

### 37.4.1.23.1 Offset

| Register | Offset |
|---|---|
| LPI2C3 | 54h |

### 37.4.1.23.2 Function

TRGMUX Register

## 37.4.1.23.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.23.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.1.24 TRGMUX LPSPI3 Register (LPSPI3)

## 37.4.1.24.1 Offset

| Register | Offset |
|---|---|
| LPSPI3 | 58h |

## 37.4.1.24.2 Function

TRGMUX Register

## 37.4.1.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.24.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.25 TRGMUX LPUART3 Register (LPUART3)

### 37.4.1.25.1 Offset

| Register | Offset |
|---|---|
| LPUART3 | 5Ch |

### 37.4.1.25.2 Function

TRGMUX Register

### 37.4.1.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.1.25.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.1.26 TRGMUX LPCMP1 Register (LPCMP1)

### 37.4.1.26.1 Offset

| Register | Offset |
|---|---|
| LPCMP1 | 60h |

### 37.4.1.26.2 Function

TRGMUX Register

## 37.4.1.26.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.1.26.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2 TRGMUX1 register descriptions

### 37.4.2.1 TRGMUX1 Memory map

**Table 37-6. Select Bit Fields**

| Field | Function |
|---|---|
| SELx | This read/write field is used to configure the MUX select for the peripheral trigger inputs.<br>000_0000 - (0x00) Trigger function is disabled<br>000_0001 - (0x01) LLWU1 Trigger Event input is selected<br>000_0010 - (0x02) LPIT1 Channel 0 input is selected<br>000_0011 - (0x03) LPIT1 Channel 1 input is selected<br>000_0100 - (0x04) LPIT1 Channel 2 input is selected<br>000_0101 - (0x05) LPIT1 Channel 3 input is selected<br>000_0110 - (0x06) LPTMR2 Trigger input is selected<br>000_0111 - (0x07) TPM3 Channel (Even) input is selected<br>000_1000 - (0x08) TPM3 Channel (Odd) input is selected<br>000_1001 - (0x09) TPM3 Overflow input is selected<br>000_1010 - (0x0A) LPI2C3 Master STOP input is selected<br>000_1011 - (0x0B) LPI2C3 Slave STOP input is selected<br>000_1100 - (0x0C) LPSPI3 Frame input is selected<br>000_1101 - (0x0D) LPSPI3 RX data input is selected<br>000_1110 - (0x0E) LPUART3 RX data input is selected<br>000_1111 - (0x0F) LPUART3 RX Idle input is selected<br>001_0000 - (0x10) LPUART3 TX data input is selected<br>001_0001 - (0x11) PORTE Pin Trigger input is selected<br>001_0010 - (0x12) LPCMP Output input is selected<br>001_0011 - (0x13) RTC Alarm input is selected<br>001_0100 - (0x14) RTC Seconds input is selected<br>001_0101 - (0x15) LPTMR0 Trigger input is selected<br>001_0110 - (0x16) LPTMR1 Trigger input is selected<br>001_0111 - (0x17) TPM1 Channel (Even) input is selected<br>001_1000 - (0x18) TPM1 Channel (Odd) input is selected<br>001_1001 - (0x19) TPM1 Overflow input is selected<br>001_1010 - (0x1A) TPM2 Channel (Even) input is selected<br>001_1011 - (0x1B) TPM2 Channel (Odd) input is selected<br>001_1100 - (0x1C) TPM2 Overflow input is selected<br>001_1101 - (0x1D) FLEXIO0 Time 0 input is selected<br>001_1110 - (0x1E) FLEXIO0 Time 1 input is selected<br>001_1111 - (0x1F) FLEXIO0 Time 2 input is selected |

**Table 37-6. Select Bit Fields**

| Field | Function |
|---|---|
| | 010_0000 - (0x20) FLEXIO0 Time 3 input is selected |
| | 010_0001 - (0x21) FLEXIO0 Time 4 input is selected |
| | 010_0010 - (0x22) FLEXIO0 Time 5 input is selected |
| | 010_0011 - (0x23) FLEXIO0 Time 6 input is selected |
| | 010_0100 - (0x24) FLEXIO0 Time 7 input is selected |
| | 010_0101 - (0x25) LPI2C0 Master STOP input is selected |
| | 010_0110 - (0x26) LPI2C0 Slave STOP input is selected |
| | 010_0111 - (0x27) LPI2C1 Master STOP input is selected |
| | 010_1000 - (0x28) LPI2C1 Slave STOP input is selected |
| | 010_1001 - (0x29) LPI2C2 Master STOP input is selected |
| | 010_1010 - (0x2A) LPI2C2 Slave STOP input is selected |
| | 010_1011 - (0x2B) I2S0 RX Frame Sync input is selected |
| | 010_1100 - (0x2C) I2S0 TX Frame Sync input is selected |
| | 010_1101 - (0x2D) LPSPI0 Frame input is selected |
| | 010_1110 - (0x2E) LPSPI0 RX data input is selected |
| | 010_1111 - (0x2F) LPSPI1 Frame input is selected |
| | 011_0000 - (0x30) LPSPI1 RX data input is selected |
| | 011_0001 - (0x31) LPSPI2 Frame input is selected |
| | 011_0010 - (0x32) LPSPI2 RX data input is selected |
| | 011_0011 - (0x33) LPUART0 RX data input is selected |
| | 011_0100 - (0x34) LPUART0 RX Idle input is selected |
| | 011_0101 - (0x35) LPUART0 TX data input is selected |
| | 011_0110 - (0x36) LPUART1 RX data input is selected |
| | 011_0111 - (0x37) LPUART1 RX Idle input is selected |
| | 011_1000 - (0x38) LPUART1 TX data input is selected |
| | 011_1001 - (0x39) LPUART2 RX data input is selected |
| | 011_1010 - (0x3A) LPUART2 RX Idle input is selected |
| | 011_1011 - (0x3B) LPUART2 TX data input is selected |
| | 011_1100 - (0x3C) PORTA Pin Trigger input is selected |
| | 011_1101 - (0x3D) PORTB Pin Trigger input is selected |
| | 011_1110 - (0x3E) PORTC Pin Trigger input is selected |
| | 011_1111 - (0x3F) PORTD Pin Trigger input is selected |

TRGMUX1 base address: 4102_5000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TRGMUX DMAMUX1 Register (DMAMUX1) | 32 | RW | 0000_0000h |
| 4h | TRGMUX LPIT1 Register (LPIT1) | 32 | RW | 0000_0000h |
| 8h | TRGMUX TPM3 Register (TPM3) | 32 | RW | 0000_0000h |
| Ch | TRGMUX LPI2C3 Register (LPI2C3) | 32 | RW | 0000_0000h |
| 10h | TRGMUX LPSPI3 Register (LPSPI3) | 32 | RW | 0000_0000h |
| 14h | TRGMUX LPUART3 Register (LPUART3) | 32 | RW | 0000_0000h |
| 18h | TRGMUX LPCMP1 Register (LPCMP1) | 32 | RW | 0000_0000h |
| 1Ch | TRGMUX DMAMUX0 Register (DMAMUX0) | 32 | RW | 0000_0000h |
| 20h | TRGMUX LPIT0 Register (LPIT0) | 32 | RW | 0000_0000h |
| 24h | TRGMUX TPM0 Register (TPM0) | 32 | RW | 0000_0000h |
| 28h | TRGMUX TPM1 Register (TPM1) | 32 | RW | 0000_0000h |
| 2Ch | TRGMUX TPM2 Register (TPM2) | 32 | RW | 0000_0000h |
| 30h | TRGMUX FLEXIO0 Register (FLEXIO0) | 32 | RW | 0000_0000h |
| 34h | TRGMUX LPI2C0 Register (LPI2C0) | 32 | RW | 0000_0000h |
| 38h | TRGMUX LPI2C1 Register (LPI2C1) | 32 | RW | 0000_0000h |
| 3Ch | TRGMUX LPI2C2 Register (LPI2C2) | 32 | RW | 0000_0000h |
| 40h | TRGMUX LPSPI0 Register (LPSPI0) | 32 | RW | 0000_0000h |
| 44h | TRGMUX LPSPI1 Register (LPSPI1) | 32 | RW | 0000_0000h |
| 48h | TRGMUX LPSPI2 Register (LPSPI2) | 32 | RW | 0000_0000h |
| 4Ch | TRGMUX LPUART0 Register (LPUART0) | 32 | RW | 0000_0000h |
| 50h | TRGMUX LPUART1 Register (LPUART1) | 32 | RW | 0000_0000h |
| 54h | TRGMUX LPUART2 Register (LPUART2) | 32 | RW | 0000_0000h |
| 58h | TRGMUX LPADC0 Register (LPADC0) | 32 | RW | 0000_0000h |
| 5Ch | TRGMUX LPCMP0 Register (LPCMP0) | 32 | RW | 0000_0000h |
| 60h | TRGMUX LPDAC0 Register (LPDAC0) | 32 | RW | 0000_0000h |

## 37.4.2.2 TRGMUX DMAMUX1 Register (DMAMUX1)

### 37.4.2.2.1 Offset

| Register | Offset |
|---|---|
| DMAMUX1 | 0h |

### 37.4.2.2.2 Function

TRGMUX Register

### 37.4.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.2.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.3 TRGMUX LPIT1 Register (LPIT1)

### 37.4.2.3.1 Offset

| Register | Offset |
|---|---|
| LPIT1 | 4h |

### 37.4.2.3.2 Function

TRGMUX Register

### 37.4.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.3.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.4 TRGMUX TPM3 Register (TPM3)

### 37.4.2.4.1 Offset

| Register | Offset |
|---|---|
| TPM3 | 8h |

### 37.4.2.4.2 Function

TRGMUX Register

### 37.4.2.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.4.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.5 TRGMUX LPI2C3 Register (LPI2C3)

### 37.4.2.5.1 Offset

| Register | Offset |
|---|---|
| LPI2C3 | Ch |

### 37.4.2.5.2 Function

TRGMUX Register

### 37.4.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.5.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.6 TRGMUX LPSPI3 Register (LPSPI3)

### 37.4.2.6.1 Offset

| Register | Offset |
|---|---|
| LPSPI3 | 10h |

### 37.4.2.6.2 Function

TRGMUX Register

### 37.4.2.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.6.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.2.7 TRGMUX LPUART3 Register (LPUART3)

## 37.4.2.7.1 Offset

| Register | Offset |
|---|---|
| LPUART3 | 14h |

## 37.4.2.7.2 Function

TRGMUX Register

## 37.4.2.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.7.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.8 TRGMUX LPCMP1 Register (LPCMP1)

### 37.4.2.8.1 Offset

| Register | Offset |
|---|---|
| LPCMP1 | 18h |

### 37.4.2.8.2 Function

TRGMUX Register

### 37.4.2.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 37.4.2.8.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.9 TRGMUX DMAMUX0 Register (DMAMUX0)

### 37.4.2.9.1 Offset

| Register | Offset |
|---|---|
| DMAMUX0 | 1Ch |

### 37.4.2.9.2 Function

TRGMUX Register

### 37.4.2.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.9.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.10 TRGMUX LPIT0 Register (LPIT0)

### 37.4.2.10.1 Offset

| Register | Offset |
|---|---|
| LPIT0 | 20h |

### 37.4.2.10.2 Function

TRGMUX Register

### 37.4.2.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.10.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.11 TRGMUX TPM0 Register (TPM0)

### 37.4.2.11.1 Offset

| Register | Offset |
|---|---|
| TPM0 | 24h |

### 37.4.2.11.2 Function

TRGMUX Register

### 37.4.2.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.11.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.12 TRGMUX TPM1 Register (TPM1)

### 37.4.2.12.1 Offset

| Register | Offset |
|---|---|
| TPM1 | 28h |

### 37.4.2.12.2 Function

TRGMUX Register

### 37.4.2.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.12.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24 | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.13 TRGMUX TPM2 Register (TPM2)

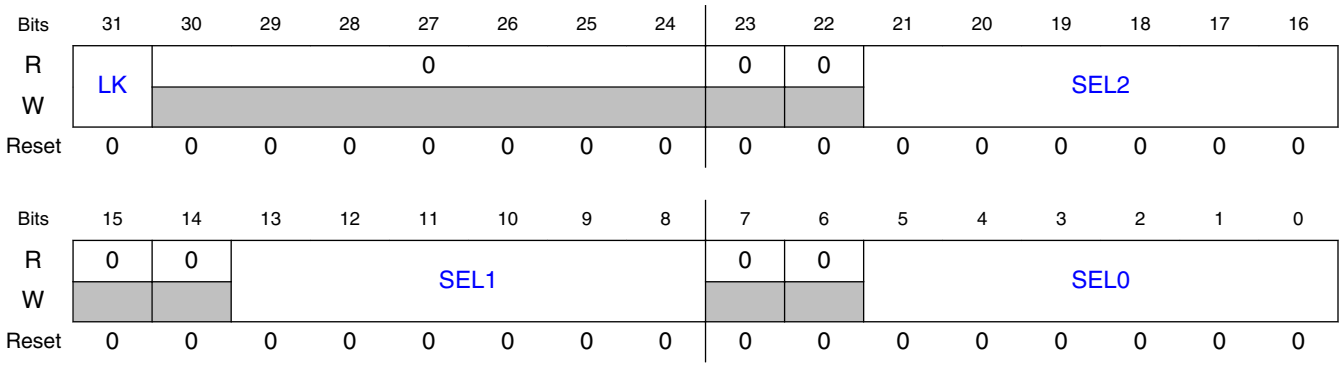
### 37.4.2.13.1 Offset

| Register | Offset |
|---|---|
| TPM2 | 2Ch |

### 37.4.2.13.2 Function

TRGMUX Register

### 37.4.2.13.3 Diagram




### 37.4.2.13.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0 | Trigger MUX Input 0 Source Select |

| Field | Function |
|---|---|
| SEL0 | This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.14 TRGMUX FLEXIO0 Register (FLEXIO0)

### 37.4.2.14.1 Offset

| Register | Offset |
|---|---|
| FLEXIO0 | 30h |

### 37.4.2.14.2 Function

TRGMUX Register

### 37.4.2.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.14.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.15 TRGMUX LPI2C0 Register (LPI2C0)

### 37.4.2.15.1 Offset

| Register | Offset |
|---|---|
| LPI2C0 | 34h |

### 37.4.2.15.2 Function

TRGMUX Register

## 37.4.2.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.15.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.2.16 TRGMUX LPI2C1 Register (LPI2C1)

## 37.4.2.16.1 Offset

| Register | Offset |
|---|---|
| LPI2C1 | 38h |

## 37.4.2.16.2 Function

TRGMUX Register

## 37.4.2.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.16.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.17 TRGMUX LPI2C2 Register (LPI2C2)

### 37.4.2.17.1 Offset

| Register | Offset |
|---|---|
| LPI2C2 | 3Ch |

### 37.4.2.17.2 Function

TRGMUX Register

### 37.4.2.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.17.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.18 TRGMUX LPSPI0 Register (LPSPI0)

### 37.4.2.18.1 Offset

| Register | Offset |
|---|---|
| LPSPI0 | 40h |

### 37.4.2.18.2 Function

TRGMUX Register

## 37.4.2.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.18.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.2.19 TRGMUX LPSPI1 Register (LPSPI1)

## 37.4.2.19.1 Offset

| Register | Offset |
|---|---|
| LPSPI1 | 44h |

## 37.4.2.19.2 Function

TRGMUX Register

## 37.4.2.19.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.19.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.20 TRGMUX LPSPI2 Register (LPSPI2)

### 37.4.2.20.1 Offset

| Register | Offset |
|---|---|
| LPSPI2 | 48h |

### 37.4.2.20.2 Function

TRGMUX Register

### 37.4.2.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 37.4.2.20.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.21 TRGMUX LPUART0 Register (LPUART0)

### 37.4.2.21.1 Offset

| Register | Offset |
|---|---|
| LPUART0 | 4Ch |

### 37.4.2.21.2 Function

TRGMUX Register

## 37.4.2.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.21.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# 37.4.2.22 TRGMUX LPUART1 Register (LPUART1)

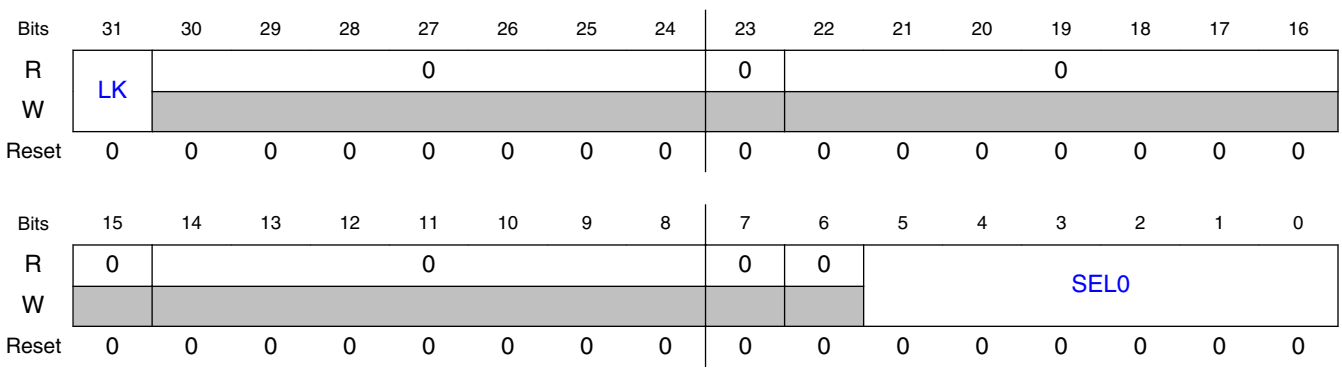
## 37.4.2.22.1 Offset

| Register | Offset |
|---|---|
| LPUART1 | 50h |

## 37.4.2.22.2 Function

TRGMUX Register

## 37.4.2.22.3 Diagram




## 37.4.2.22.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.23 TRGMUX LPUART2 Register (LPUART2)

### 37.4.2.23.1 Offset

| Register | Offset |
|---|---|
| LPUART2 | 54h |

### 37.4.2.23.2 Function

TRGMUX Register

### 37.4.2.23.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 37.4.2.23.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.24 TRGMUX LPADC0 Register (LPADC0)

### 37.4.2.24.1 Offset

| Register | Offset |
|---|---|
| LPADC0 | 58h |

### 37.4.2.24.2 Function

TRGMUX Register

## 37.4.2.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | SEL3 | | | | | | 0 | 0 | SEL2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SEL1 | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.24.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30<br>— | This read-only bit field is reserved and always has the value 0. |
| 29-24<br>SEL3 | Trigger MUX Input 3 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 3. For the field setting definitions, see Memory map and register definition. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22<br>— | This read-only bit field is reserved and always has the value 0. |
| 21-16<br>SEL2 | Trigger MUX Input 2 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 2. For the field setting definitions, see Memory map and register definition. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14<br>— | This read-only bit field is reserved and always has the value 0. |
| 13-8<br>SEL1 | Trigger MUX Input 1 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 1. For the field setting definitions, see Memory map and register definition. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.25 TRGMUX LPCMP0 Register (LPCMP0)

### 37.4.2.25.1 Offset

| Register | Offset |
|---|---|
| LPCMP0 | 5Ch |

### 37.4.2.25.2 Function

TRGMUX Register

### 37.4.2.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 37.4.2.25.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

## 37.4.2.26 TRGMUX LPDAC0 Register (LPDAC0)

### 37.4.2.26.1 Offset

| Register | Offset |
|---|---|
| LPDAC0 | 60h |

### 37.4.2.26.2 Function

TRGMUX Register

## 37.4.2.26.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | 0 | SEL0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 37.4.2.26.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX register lock.<br>This bit shows whether the register can be written or not. The LK bit can only be written once after any system reset. Once LK is set, the SELx bits in this TRGMUX register cannot be changed until the next system reset clears LK.<br>0b - Register can be written.<br>1b - Register cannot be written until the next system Reset. |
| 30-24<br>— | This read-only bit field is reserved and always has the value 0. |
| 23<br>— | This read-only bit field is reserved and always has the value 0. |
| 22-16<br>— | This read-only bit field is reserved and always has the value 0. |
| 15<br>— | This read-only bit field is reserved and always has the value 0. |
| 14-8<br>— | This read-only bit field is reserved and always has the value 0. |
| 7<br>— | This read-only bit field is reserved and always has the value 0. |
| 6<br>— | This read-only bit field is reserved and always has the value 0. |
| 5-0<br>SEL0 | Trigger MUX Input 0 Source Select<br>This read/write bit field is used to configure the MUX select for peripheral trigger input 0. For the field setting definitions, see Memory map and register definition. |

# Chapter 38
# Enhanced Direct Memory Access (eDMA)

## 38.1 Chip-specific eDMA information

This device has two DMAs. DMA0 has 16 channels and DMA1 has 8 channels.

## 38.2 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 16 channels

### 38.2.1 eDMA system block diagram

Figure 38-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

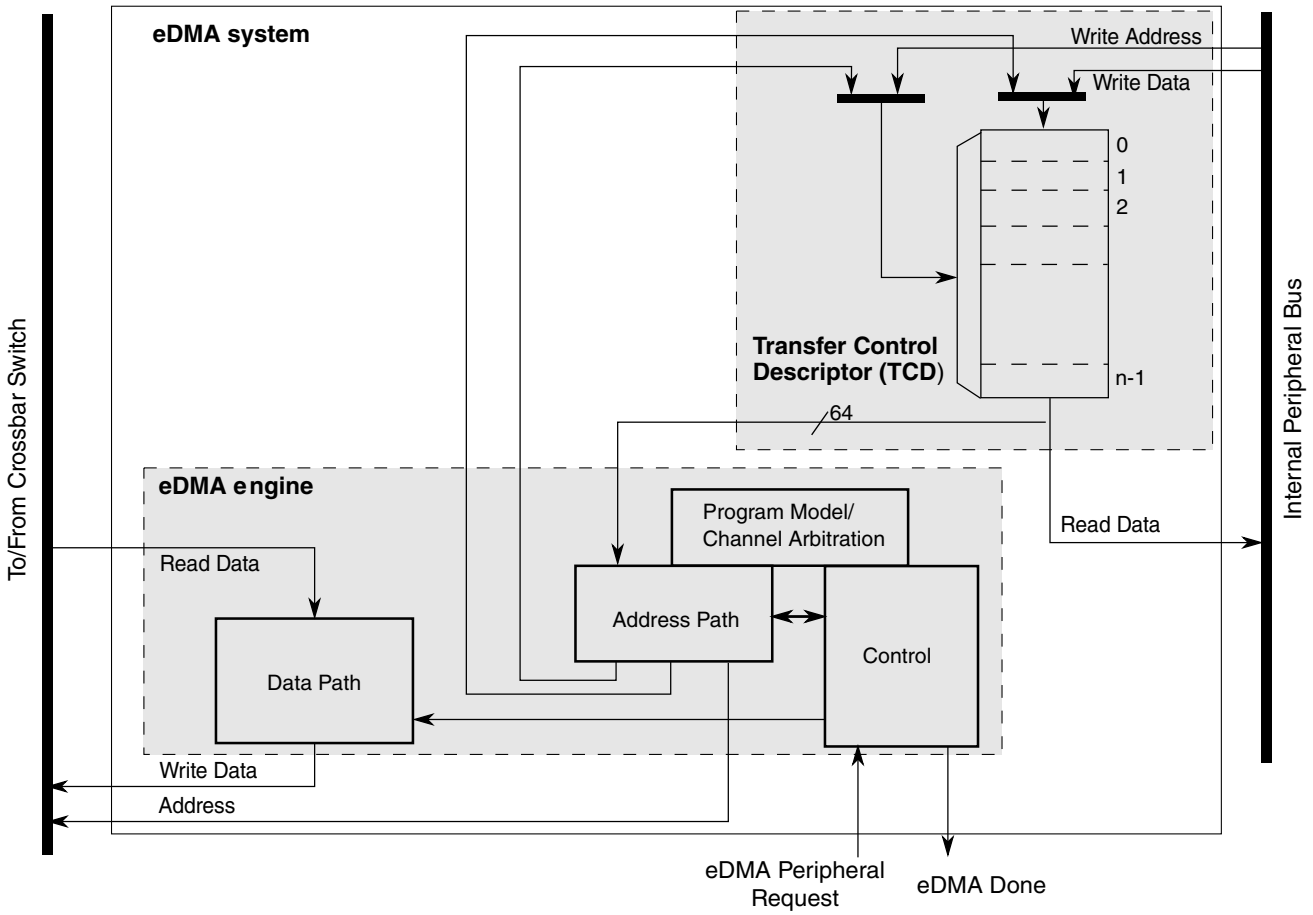



**Figure 38-1. eDMA system block diagram**

## 38.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

**Table 38-1. eDMA engine submodules**

| Submodule | Function |
|---|---|
| Address path | This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.<br><br>When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes |

*Table continues on the next page...*

**Table 38-1. eDMA engine submodules (continued)**

| Submodule | Function |
|---|---|
| | the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation. |
| Data path | This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.<br>The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase). |
| Program model/channel arbitration | This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic). |
| Control | This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write. |

The transfer-control descriptor local memory is further partitioned into:

**Table 38-2. Transfer control descriptor memory**

| Submodule | Description |
|---|---|
| Memory controller | This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled. |
| Memory array | TCD storage for each channel's transfer profile. |

## 38.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes

- 16-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

## 38.3 Modes of operation

The eDMA operates in the following modes:

**Table 38-3. Modes of operation**

| Mode | Description |
|---|---|
| Normal | In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.<br>A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data. |
| Debug | DMA operation is configurable in Debug mode via the control register:<br>• If CR[EDBG] is cleared, the DMA continues to operate.<br>• If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires. |
| Wait | Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode. |

## 38.4 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

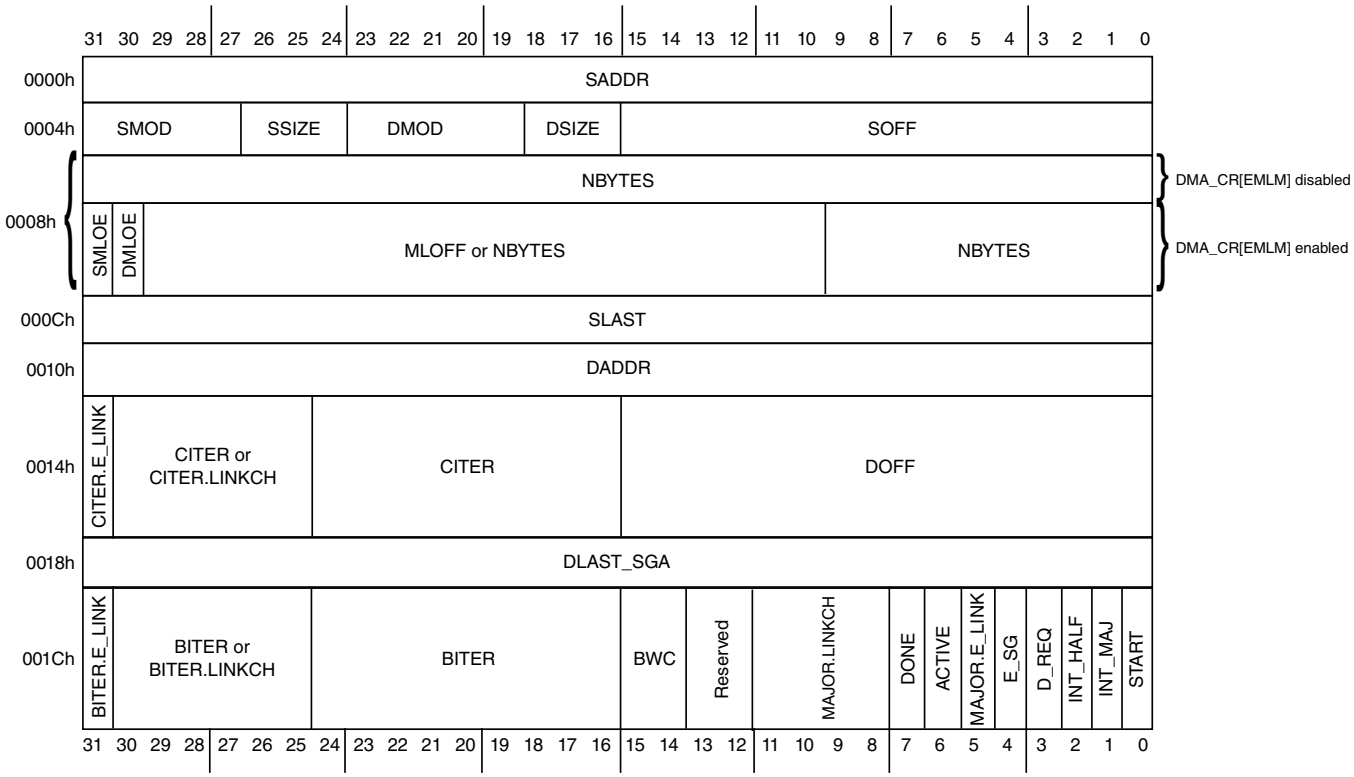
### 38.4.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 15. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

### 38.4.2 TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

## 38.4.3 TCD structure




## 38.4.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

## 38.4.5 DMA0 register descriptions

### 38.4.5.1 DMA0 Memory map

DMA0 base address: 4000_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control Register (CR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Error Status Register (ES) | 32 | RO | 0000_0000h |
| Ch | Enable Request Register (ERQ) | 32 | RW | 0000_0000h |
| 14h | Enable Error Interrupt Register (EEI) | 32 | RW | 0000_0000h |
| 18h | Clear Enable Error Interrupt Register (CEEI) | 8 | WORZ | 00h |
| 19h | Set Enable Error Interrupt Register (SEEI) | 8 | WORZ | 00h |
| 1Ah | Clear Enable Request Register (CERQ) | 8 | WORZ | 00h |
| 1Bh | Set Enable Request Register (SERQ) | 8 | WORZ | 00h |
| 1Ch | Clear DONE Status Bit Register (CDNE) | 8 | WORZ | 00h |
| 1Dh | Set START Bit Register (SSRT) | 8 | WORZ | 00h |
| 1Eh | Clear Error Register (CERR) | 8 | WORZ | 00h |
| 1Fh | Clear Interrupt Request Register (CINT) | 8 | WORZ | 00h |
| 24h | Interrupt Request Register (INT) | 32 | W1C | 0000_0000h |
| 2Ch | Error Register (ERR) | 32 | W1C | 0000_0000h |
| 34h | Hardware Request Status Register (HRS) | 32 | RO | 0000_0000h |
| 44h | Enable Asynchronous Request in Stop Register (EARS) | 32 | RW | 0000_0000h |
| 100h | Channel Priority Register (DCHPRI3) | 8 | RW | 03h |
| 101h | Channel Priority Register (DCHPRI2) | 8 | RW | 02h |
| 102h | Channel Priority Register (DCHPRI1) | 8 | RW | 01h |
| 103h | Channel Priority Register (DCHPRI0) | 8 | RW | 00h |
| 104h | Channel Priority Register (DCHPRI7) | 8 | RW | 07h |
| 105h | Channel Priority Register (DCHPRI6) | 8 | RW | 06h |
| 106h | Channel Priority Register (DCHPRI5) | 8 | RW | 05h |
| 107h | Channel Priority Register (DCHPRI4) | 8 | RW | 04h |
| 108h | Channel Priority Register (DCHPRI11) | 8 | RW | 0Bh |
| 109h | Channel Priority Register (DCHPRI10) | 8 | RW | 0Ah |
| 10Ah | Channel Priority Register (DCHPRI9) | 8 | RW | 09h |
| 10Bh | Channel Priority Register (DCHPRI8) | 8 | RW | 08h |
| 10Ch | Channel Priority Register (DCHPRI15) | 8 | RW | 0Fh |
| 10Dh | Channel Priority Register (DCHPRI14) | 8 | RW | 0Eh |
| 10Eh | Channel Priority Register (DCHPRI13) | 8 | RW | 0Dh |
| 10Fh | Channel Priority Register (DCHPRI12) | 8 | RW | 0Ch |
| 1000h | TCD Source Address (TCD0_SADDR) | 32 | RW | See description. |
| 1004h | TCD Signed Source Address Offset (TCD0_SOFF) | 16 | RW | See description. |
| 1006h | TCD Transfer Attributes (TCD0_ATTR) | 16 | RW | See description. |
| 1008h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO) | 32 | RW | See description. |
| 1008h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1008h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 100Ch | TCD Last Source Address Adjustment (TCD0_SLAST) | 32 | RW | See description. |
| 1010h | TCD Destination Address (TCD0_DADDR) | 32 | RW | See description. |
| 1014h | TCD Signed Destination Address Offset (TCD0_DOFF) | 16 | RW | See description. |
| 1016h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO) | 16 | RW | See description. |
| 1016h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES) | 16 | RW | See description. |
| 1018h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA) | 32 | RW | See description. |
| 101Ch | TCD Control and Status (TCD0_CSR) | 16 | RW | See description. |
| 101Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO) | 16 | RW | See description. |
| 101Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES) | 16 | RW | See description. |
| 1020h | TCD Source Address (TCD1_SADDR) | 32 | RW | See description. |
| 1024h | TCD Signed Source Address Offset (TCD1_SOFF) | 16 | RW | See description. |
| 1026h | TCD Transfer Attributes (TCD1_ATTR) | 16 | RW | See description. |
| 1028h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD1_NBYTES_MLNO) | 32 | RW | See description. |
| 1028h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD1_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1028h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD1_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 102Ch | TCD Last Source Address Adjustment (TCD1_SLAST) | 32 | RW | See description. |
| 1030h | TCD Destination Address (TCD1_DADDR) | 32 | RW | See description. |
| 1034h | TCD Signed Destination Address Offset (TCD1_DOFF) | 16 | RW | See description. |
| 1036h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_CITER_ELINKNO) | 16 | RW | See description. |
| 1036h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_CITER_ELINKYES) | 16 | RW | See description. |
| 1038h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD1_DLASTSGA) | 32 | RW | See description. |
| 103Ch | TCD Control and Status (TCD1_CSR) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 103Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_BITER_ELINKNO) | 16 | RW | See description. |
| 103Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_BITER_ELINKYES) | 16 | RW | See description. |
| 1040h | TCD Source Address (TCD2_SADDR) | 32 | RW | See description. |
| 1044h | TCD Signed Source Address Offset (TCD2_SOFF) | 16 | RW | See description. |
| 1046h | TCD Transfer Attributes (TCD2_ATTR) | 16 | RW | See description. |
| 1048h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD2_NBYTES_MLNO) | 32 | RW | See description. |
| 1048h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD2_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1048h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD2_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 104Ch | TCD Last Source Address Adjustment (TCD2_SLAST) | 32 | RW | See description. |
| 1050h | TCD Destination Address (TCD2_DADDR) | 32 | RW | See description. |
| 1054h | TCD Signed Destination Address Offset (TCD2_DOFF) | 16 | RW | See description. |
| 1056h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_CITER_ELINKNO) | 16 | RW | See description. |
| 1056h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_CITER_ELINKYES) | 16 | RW | See description. |
| 1058h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD2_DLASTSGA) | 32 | RW | See description. |
| 105Ch | TCD Control and Status (TCD2_CSR) | 16 | RW | See description. |
| 105Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_BITER_ELINKNO) | 16 | RW | See description. |
| 105Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_BITER_ELINKYES) | 16 | RW | See description. |
| 1060h | TCD Source Address (TCD3_SADDR) | 32 | RW | See description. |
| 1064h | TCD Signed Source Address Offset (TCD3_SOFF) | 16 | RW | See description. |
| 1066h | TCD Transfer Attributes (TCD3_ATTR) | 16 | RW | See description. |
| 1068h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD3_NBYTES_MLNO) | 32 | RW | See description. |
| 1068h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD3_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1068h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD3_NBYTES_MLOFFYES) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 106Ch | TCD Last Source Address Adjustment (TCD3_SLAST) | 32 | RW | See description. |
| 1070h | TCD Destination Address (TCD3_DADDR) | 32 | RW | See description. |
| 1074h | TCD Signed Destination Address Offset (TCD3_DOFF) | 16 | RW | See description. |
| 1076h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_CITER_ELINKNO) | 16 | RW | See description. |
| 1076h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_CITER_ELINKYES) | 16 | RW | See description. |
| 1078h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD3_DLASTSGA) | 32 | RW | See description. |
| 107Ch | TCD Control and Status (TCD3_CSR) | 16 | RW | See description. |
| 107Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_BITER_ELINKNO) | 16 | RW | See description. |
| 107Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_BITER_ELINKYES) | 16 | RW | See description. |
| 1080h | TCD Source Address (TCD4_SADDR) | 32 | RW | See description. |
| 1084h | TCD Signed Source Address Offset (TCD4_SOFF) | 16 | RW | See description. |
| 1086h | TCD Transfer Attributes (TCD4_ATTR) | 16 | RW | See description. |
| 1088h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD4_NBYTES_MLNO) | 32 | RW | See description. |
| 1088h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD4_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1088h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD4_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 108Ch | TCD Last Source Address Adjustment (TCD4_SLAST) | 32 | RW | See description. |
| 1090h | TCD Destination Address (TCD4_DADDR) | 32 | RW | See description. |
| 1094h | TCD Signed Destination Address Offset (TCD4_DOFF) | 16 | RW | See description. |
| 1096h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_CITER_ELINKNO) | 16 | RW | See description. |
| 1096h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_CITER_ELINKYES) | 16 | RW | See description. |
| 1098h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD4_DLASTSGA) | 32 | RW | See description. |
| 109Ch | TCD Control and Status (TCD4_CSR) | 16 | RW | See description. |
| 109Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_BITER_ELINKNO) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 109Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_BITER_ELINKYES) | 16 | RW | See description. |
| 10A0h | TCD Source Address (TCD5_SADDR) | 32 | RW | See description. |
| 10A4h | TCD Signed Source Address Offset (TCD5_SOFF) | 16 | RW | See description. |
| 10A6h | TCD Transfer Attributes (TCD5_ATTR) | 16 | RW | See description. |
| 10A8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD5_NBYTES_MLNO) | 32 | RW | See description. |
| 10A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD5_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD5_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10ACh | TCD Last Source Address Adjustment (TCD5_SLAST) | 32 | RW | See description. |
| 10B0h | TCD Destination Address (TCD5_DADDR) | 32 | RW | See description. |
| 10B4h | TCD Signed Destination Address Offset (TCD5_DOFF) | 16 | RW | See description. |
| 10B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_CITER_ELINKNO) | 16 | RW | See description. |
| 10B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_CITER_ELINKYES) | 16 | RW | See description. |
| 10B8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD5_DLASTSGA) | 32 | RW | See description. |
| 10BCh | TCD Control and Status (TCD5_CSR) | 16 | RW | See description. |
| 10BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_BITER_ELINKNO) | 16 | RW | See description. |
| 10BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_BITER_ELINKYES) | 16 | RW | See description. |
| 10C0h | TCD Source Address (TCD6_SADDR) | 32 | RW | See description. |
| 10C4h | TCD Signed Source Address Offset (TCD6_SOFF) | 16 | RW | See description. |
| 10C6h | TCD Transfer Attributes (TCD6_ATTR) | 16 | RW | See description. |
| 10C8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD6_NBYTES_MLNO) | 32 | RW | See description. |
| 10C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD6_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD6_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10CCh | TCD Last Source Address Adjustment (TCD6_SLAST) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10D0h | TCD Destination Address (TCD6_DADDR) | 32 | RW | See description. |
| 10D4h | TCD Signed Destination Address Offset (TCD6_DOFF) | 16 | RW | See description. |
| 10D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_CITER_ELINKNO) | 16 | RW | See description. |
| 10D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_CITER_ELINKYES) | 16 | RW | See description. |
| 10D8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD6_DLASTSGA) | 32 | RW | See description. |
| 10DCh | TCD Control and Status (TCD6_CSR) | 16 | RW | See description. |
| 10DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_BITER_ELINKNO) | 16 | RW | See description. |
| 10DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_BITER_ELINKYES) | 16 | RW | See description. |
| 10E0h | TCD Source Address (TCD7_SADDR) | 32 | RW | See description. |
| 10E4h | TCD Signed Source Address Offset (TCD7_SOFF) | 16 | RW | See description. |
| 10E6h | TCD Transfer Attributes (TCD7_ATTR) | 16 | RW | See description. |
| 10E8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD7_NBYTES_MLNO) | 32 | RW | See description. |
| 10E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD7_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD7_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10ECh | TCD Last Source Address Adjustment (TCD7_SLAST) | 32 | RW | See description. |
| 10F0h | TCD Destination Address (TCD7_DADDR) | 32 | RW | See description. |
| 10F4h | TCD Signed Destination Address Offset (TCD7_DOFF) | 16 | RW | See description. |
| 10F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_CITER_ELINKNO) | 16 | RW | See description. |
| 10F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_CITER_ELINKYES) | 16 | RW | See description. |
| 10F8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD7_DLASTSGA) | 32 | RW | See description. |
| 10FCh | TCD Control and Status (TCD7_CSR) | 16 | RW | See description. |
| 10FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_BITER_ELINKNO) | 16 | RW | See description. |
| 10FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_BITER_ELINKYES) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1100h | TCD Source Address (TCD8_SADDR) | 32 | RW | See description. |
| 1104h | TCD Signed Source Address Offset (TCD8_SOFF) | 16 | RW | See description. |
| 1106h | TCD Transfer Attributes (TCD8_ATTR) | 16 | RW | See description. |
| 1108h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD8_NBYTES_MLNO) | 32 | RW | See description. |
| 1108h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD8_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1108h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD8_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 110Ch | TCD Last Source Address Adjustment (TCD8_SLAST) | 32 | RW | See description. |
| 1110h | TCD Destination Address (TCD8_DADDR) | 32 | RW | See description. |
| 1114h | TCD Signed Destination Address Offset (TCD8_DOFF) | 16 | RW | See description. |
| 1116h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD8_CITER_ELINKNO) | 16 | RW | See description. |
| 1116h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD8_CITER_ELINKYES) | 16 | RW | See description. |
| 1118h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD8_DLASTSGA) | 32 | RW | See description. |
| 111Ch | TCD Control and Status (TCD8_CSR) | 16 | RW | See description. |
| 111Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD8_BITER_ELINKNO) | 16 | RW | See description. |
| 111Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD8_BITER_ELINKYES) | 16 | RW | See description. |
| 1120h | TCD Source Address (TCD9_SADDR) | 32 | RW | See description. |
| 1124h | TCD Signed Source Address Offset (TCD9_SOFF) | 16 | RW | See description. |
| 1126h | TCD Transfer Attributes (TCD9_ATTR) | 16 | RW | See description. |
| 1128h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD9_NBYTES_MLNO) | 32 | RW | See description. |
| 1128h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD9_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1128h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD9_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 112Ch | TCD Last Source Address Adjustment (TCD9_SLAST) | 32 | RW | See description. |
| 1130h | TCD Destination Address (TCD9_DADDR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1134h | TCD Signed Destination Address Offset (TCD9_DOFF) | 16 | RW | See description. |
| 1136h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD9_CITER_ELINKNO) | 16 | RW | See description. |
| 1136h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD9_CITER_ELINKYES) | 16 | RW | See description. |
| 1138h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD9_DLASTSGA) | 32 | RW | See description. |
| 113Ch | TCD Control and Status (TCD9_CSR) | 16 | RW | See description. |
| 113Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD9_BITER_ELINKNO) | 16 | RW | See description. |
| 113Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD9_BITER_ELINKYES) | 16 | RW | See description. |
| 1140h | TCD Source Address (TCD10_SADDR) | 32 | RW | See description. |
| 1144h | TCD Signed Source Address Offset (TCD10_SOFF) | 16 | RW | See description. |
| 1146h | TCD Transfer Attributes (TCD10_ATTR) | 16 | RW | See description. |
| 1148h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD10_NBYTES_MLNO) | 32 | RW | See description. |
| 1148h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD10_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1148h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD10_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 114Ch | TCD Last Source Address Adjustment (TCD10_SLAST) | 32 | RW | See description. |
| 1150h | TCD Destination Address (TCD10_DADDR) | 32 | RW | See description. |
| 1154h | TCD Signed Destination Address Offset (TCD10_DOFF) | 16 | RW | See description. |
| 1156h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD10_CITER_ELINKNO) | 16 | RW | See description. |
| 1156h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD10_CITER_ELINKYES) | 16 | RW | See description. |
| 1158h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD10_DLASTSGA) | 32 | RW | See description. |
| 115Ch | TCD Control and Status (TCD10_CSR) | 16 | RW | See description. |
| 115Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD10_BITER_ELINKNO) | 16 | RW | See description. |
| 115Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD10_BITER_ELINKYES) | 16 | RW | See description. |
| 1160h | TCD Source Address (TCD11_SADDR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1164h | TCD Signed Source Address Offset (TCD11_SOFF) | 16 | RW | See description. |
| 1166h | TCD Transfer Attributes (TCD11_ATTR) | 16 | RW | See description. |
| 1168h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD11_NBYTES_MLNO) | 32 | RW | See description. |
| 1168h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD11_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1168h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD11_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 116Ch | TCD Last Source Address Adjustment (TCD11_SLAST) | 32 | RW | See description. |
| 1170h | TCD Destination Address (TCD11_DADDR) | 32 | RW | See description. |
| 1174h | TCD Signed Destination Address Offset (TCD11_DOFF) | 16 | RW | See description. |
| 1176h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD11_CITER_ELINKNO) | 16 | RW | See description. |
| 1176h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD11_CITER_ELINKYES) | 16 | RW | See description. |
| 1178h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD11_DLASTSGA) | 32 | RW | See description. |
| 117Ch | TCD Control and Status (TCD11_CSR) | 16 | RW | See description. |
| 117Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD11_BITER_ELINKNO) | 16 | RW | See description. |
| 117Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD11_BITER_ELINKYES) | 16 | RW | See description. |
| 1180h | TCD Source Address (TCD12_SADDR) | 32 | RW | See description. |
| 1184h | TCD Signed Source Address Offset (TCD12_SOFF) | 16 | RW | See description. |
| 1186h | TCD Transfer Attributes (TCD12_ATTR) | 16 | RW | See description. |
| 1188h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD12_NBYTES_MLNO) | 32 | RW | See description. |
| 1188h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD12_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1188h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD12_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 118Ch | TCD Last Source Address Adjustment (TCD12_SLAST) | 32 | RW | See description. |
| 1190h | TCD Destination Address (TCD12_DADDR) | 32 | RW | See description. |
| 1194h | TCD Signed Destination Address Offset (TCD12_DOFF) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1196h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD12_CITER_ELINKNO) | 16 | RW | See description. |
| 1196h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD12_CITER_ELINKYES) | 16 | RW | See description. |
| 1198h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD12_DLASTSGA) | 32 | RW | See description. |
| 119Ch | TCD Control and Status (TCD12_CSR) | 16 | RW | See description. |
| 119Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD12_BITER_ELINKNO) | 16 | RW | See description. |
| 119Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD12_BITER_ELINKYES) | 16 | RW | See description. |
| 11A0h | TCD Source Address (TCD13_SADDR) | 32 | RW | See description. |
| 11A4h | TCD Signed Source Address Offset (TCD13_SOFF) | 16 | RW | See description. |
| 11A6h | TCD Transfer Attributes (TCD13_ATTR) | 16 | RW | See description. |
| 11A8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD13_NBYTES_MLNO) | 32 | RW | See description. |
| 11A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD13_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 11A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD13_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 11ACh | TCD Last Source Address Adjustment (TCD13_SLAST) | 32 | RW | See description. |
| 11B0h | TCD Destination Address (TCD13_DADDR) | 32 | RW | See description. |
| 11B4h | TCD Signed Destination Address Offset (TCD13_DOFF) | 16 | RW | See description. |
| 11B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD13_CITER_ELINKNO) | 16 | RW | See description. |
| 11B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD13_CITER_ELINKYES) | 16 | RW | See description. |
| 11B8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD13_DLASTSGA) | 32 | RW | See description. |
| 11BCh | TCD Control and Status (TCD13_CSR) | 16 | RW | See description. |
| 11BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD13_BITER_ELINKNO) | 16 | RW | See description. |
| 11BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD13_BITER_ELINKYES) | 16 | RW | See description. |
| 11C0h | TCD Source Address (TCD14_SADDR) | 32 | RW | See description. |
| 11C4h | TCD Signed Source Address Offset (TCD14_SOFF) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 11C6h | TCD Transfer Attributes (TCD14_ATTR) | 16 | RW | See description. |
| 11C8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD14_NBYTES_MLNO) | 32 | RW | See description. |
| 11C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD14_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 11C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD14_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 11CCh | TCD Last Source Address Adjustment (TCD14_SLAST) | 32 | RW | See description. |
| 11D0h | TCD Destination Address (TCD14_DADDR) | 32 | RW | See description. |
| 11D4h | TCD Signed Destination Address Offset (TCD14_DOFF) | 16 | RW | See description. |
| 11D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD14_CITER_ELINKNO) | 16 | RW | See description. |
| 11D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD14_CITER_ELINKYES) | 16 | RW | See description. |
| 11D8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD14_DLASTSGA) | 32 | RW | See description. |
| 11DCh | TCD Control and Status (TCD14_CSR) | 16 | RW | See description. |
| 11DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD14_BITER_ELINKNO) | 16 | RW | See description. |
| 11DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD14_BITER_ELINKYES) | 16 | RW | See description. |
| 11E0h | TCD Source Address (TCD15_SADDR) | 32 | RW | See description. |
| 11E4h | TCD Signed Source Address Offset (TCD15_SOFF) | 16 | RW | See description. |
| 11E6h | TCD Transfer Attributes (TCD15_ATTR) | 16 | RW | See description. |
| 11E8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD15_NBYTES_MLNO) | 32 | RW | See description. |
| 11E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD15_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 11E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD15_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 11ECh | TCD Last Source Address Adjustment (TCD15_SLAST) | 32 | RW | See description. |
| 11F0h | TCD Destination Address (TCD15_DADDR) | 32 | RW | See description. |
| 11F4h | TCD Signed Destination Address Offset (TCD15_DOFF) | 16 | RW | See description. |
| 11F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD15_CITER_ELINKNO) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 11F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD15_CITER_ELINKYES) | 16 | RW | See description. |
| 11F8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD15_DLASTSGA) | 32 | RW | See description. |
| 11FCh | TCD Control and Status (TCD15_CSR) | 16 | RW | See description. |
| 11FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD15_BITER_ELINKNO) | 16 | RW | See description. |
| 11FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD15_BITER_ELINKYES) | 16 | RW | See description. |

## 38.4.5.2 Control Register (CR)

### 38.4.5.2.1 Offset

| Register | Offset |
|---|---|
| CR | 0h |

### 38.4.5.2.2 Function

The CR defines the basic operating configuration of the DMA.

Arbitration can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels are cycled through (from high to low channel number) without regard to priority.

**NOTE**

For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

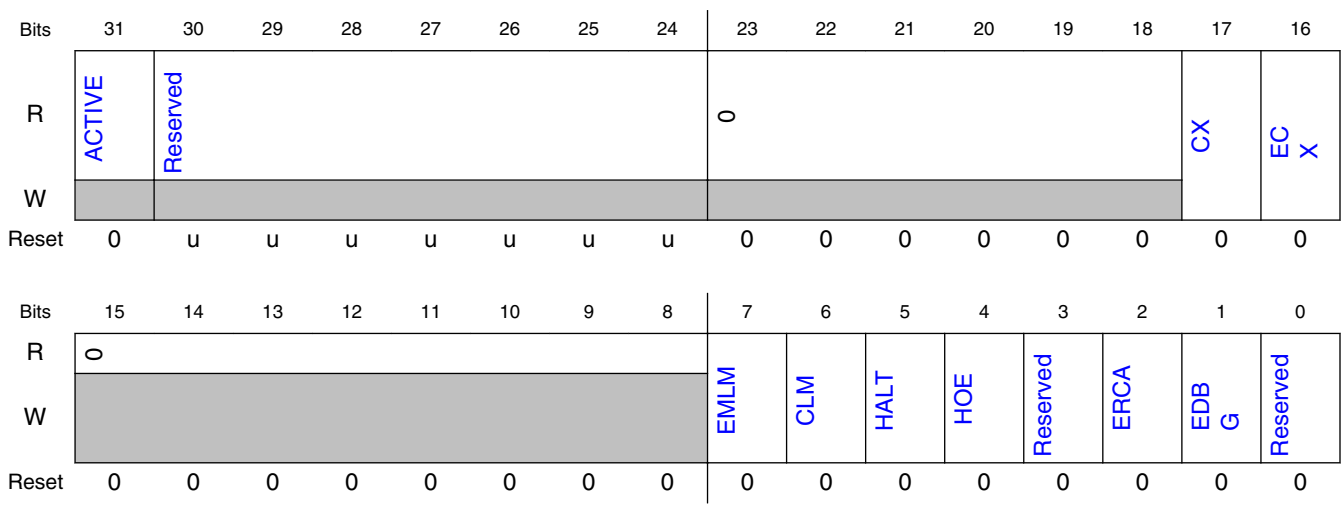
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is

complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

### 38.4.5.2.3 Diagram




### 38.4.5.2.4 Fields

| Field | Function |
|---|---|
| 31<br>ACTIVE | DMA Active Status<br>0b - eDMA is idle.<br>1b - eDMA is executing a channel. |
| 30-24<br>— | eDMA version number<br>Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-18<br>— | Reserved |
| 17<br>CX | Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed. |
| 16<br>ECX | Error Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt. |
| 15-8<br>— | Reserved |
| 7<br>EMLM | Enable Minor Loop Mapping<br>0b - Disabled. TCDn.word2 is defined as a 32-bit NBYTES field.<br>1b - Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled. |
| 6<br>CLM | Continuous Link Mode<br>**NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing.<br>0b - A minor loop channel link made to itself goes through channel arbitration before being activated again.<br>1b - A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. |
| 5<br>HALT | Halt DMA Operations<br>0b - Normal operation<br>1b - Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared. |
| 4<br>HOE | Halt On Error<br>0b - Normal operation<br>1b - Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared. |
| 3<br>— | Reserved<br>Reserved |
| 2<br>ERCA | Enable Round Robin Channel Arbitration<br>0b - Fixed priority arbitration is used for channel selection .<br>1b - Round robin arbitration is used for channel selection . |
| 1<br>EDBG | Enable Debug<br>0b - When in debug mode, the DMA continues to operate. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared. |
| 0 | Reserved |
| — | Reserved |

## 38.4.5.3 Error Status Register (ES)

### 38.4.5.3.1 Offset

| Register | Offset |
|---|---|
| ES | 4h |

### 38.4.5.3.2 Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

### 38.4.5.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VLD | 0 | | | | | | | | | | | | | | ECX |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CPE | 0 | | ERRCHN | | | | SAE | SOE | DAE | DOE | NCE | SGE | SBE | DBE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.3.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | VLD<br>Logical OR of all ERR status bits<br>0b - No ERR bits are set.<br>1b - At least one ERR bit is set indicating a valid error exists that has not been cleared. |
| 30-17<br>— | Reserved |
| 16<br>ECX | Transfer Canceled<br>0b - No canceled transfers<br>1b - The last recorded entry was a canceled transfer by the error cancel transfer input |
| 15<br>— | Reserved |
| 14<br>CPE | Channel Priority Error<br>0b - No channel priority error<br>1b - The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique. |
| 13-12<br>— | Reserved |
| 11-8<br>ERRCHN | Error Channel Number or Canceled Channel Number<br>The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer. |
| 7<br>SAE | Source Address Error<br>0b - No source address configuration error.<br>1b - The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. |
| 6<br>SOE | Source Offset Error<br>0b - No source offset configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. |
| 5<br>DAE | Destination Address Error<br>0b - No destination address configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. |
| 4<br>DOE | Destination Offset Error<br>0b - No destination offset configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. |
| 3<br>NCE | NBYTES/CITER Configuration Error<br>0b - No NBYTES/CITER configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], orTCDn_CITER[CITER] is equal to zero, orTCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] |
| 2<br>SGE | Scatter/Gather Configuration Error<br>0b - No scatter/gather configuration error |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary. |
| 1<br>SBE | Source Bus Error<br>0b - No source bus error<br>1b - The last recorded error was a bus error on a source read |
| 0<br>DBE | Destination Bus Error<br>0b - No destination bus error<br>1b - The last recorded error was a bus error on a destination write |

## 38.4.5.4 Enable Request Register (ERQ)

### 38.4.5.4.1 Offset

| Register | Offset |
|---|---|
| ERQ | Ch |

### 38.4.5.4.2 Function

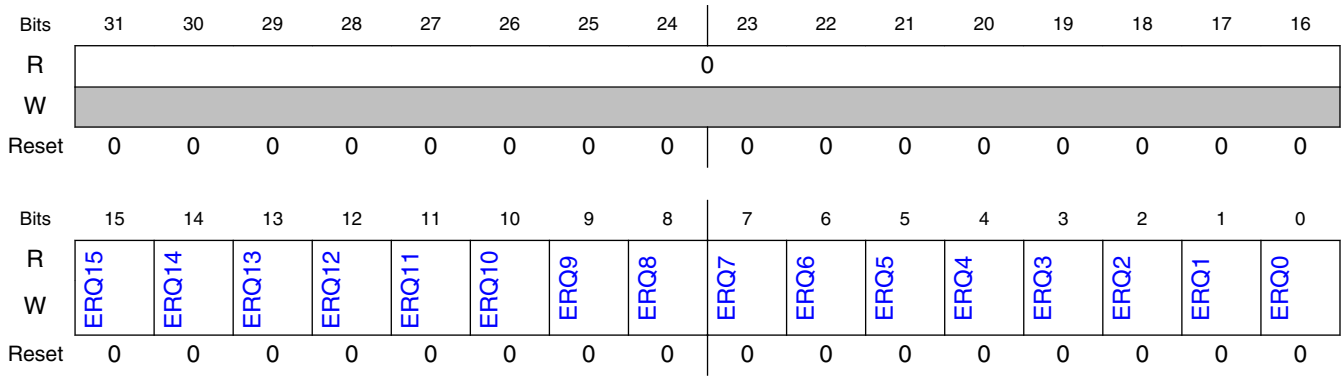
The ERQ register provides a bit map for the 16 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

**NOTE**

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

### 38.4.5.4.3 Diagram




### 38.4.5.4.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>ERQ15 | Enable DMA Request 15<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 14<br>ERQ14 | Enable DMA Request 14<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 13<br>ERQ13 | Enable DMA Request 13<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 12<br>ERQ12 | Enable DMA Request 12<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 11<br>ERQ11 | Enable DMA Request 11<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 10<br>ERQ10 | Enable DMA Request 10<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 9<br>ERQ9 | Enable DMA Request 9<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 8<br>ERQ8 | Enable DMA Request 8<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 7<br>ERQ7 | Enable DMA Request 7<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 6 | Enable DMA Request 6<br>0b - The DMA request signal for the corresponding channel is disabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ERQ6 | 1b - The DMA request signal for the corresponding channel is enabled |
| 5<br>ERQ5 | Enable DMA Request 5<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 4<br>ERQ4 | Enable DMA Request 4<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 3<br>ERQ3 | Enable DMA Request 3<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 2<br>ERQ2 | Enable DMA Request 2<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 1<br>ERQ1 | Enable DMA Request 1<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 0<br>ERQ0 | Enable DMA Request 0<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |

## 38.4.5.5 Enable Error Interrupt Register (EEI)

### 38.4.5.5.1 Offset

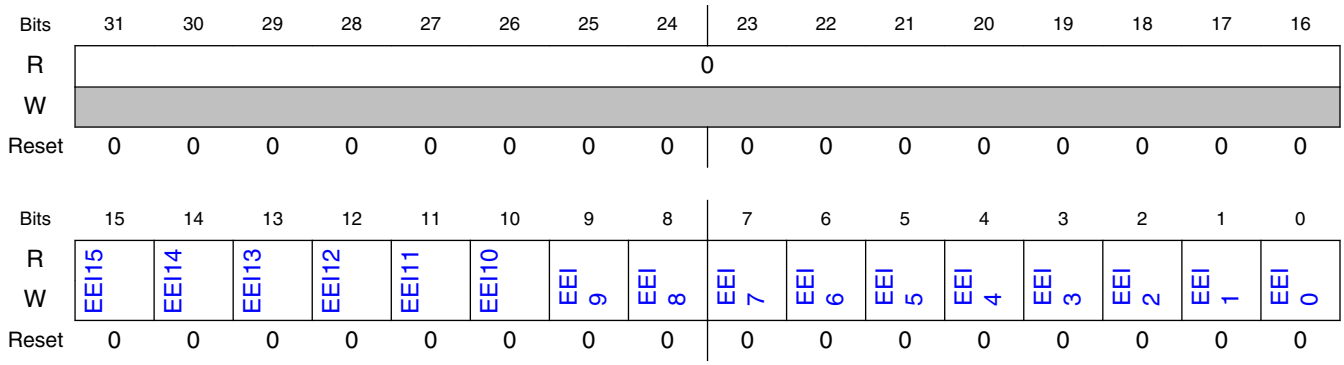
| Register | Offset |
|---|---|
| EEI | 14h |

### 38.4.5.5.2 Function

The EEI register provides a bit map for the 16 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

### 38.4.5.5.3 Diagram




### 38.4.5.5.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>EEI15 | Enable Error Interrupt 15<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 14<br>EEI14 | Enable Error Interrupt 14<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 13<br>EEI13 | Enable Error Interrupt 13<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 12<br>EEI12 | Enable Error Interrupt 12<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 11<br>EEI11 | Enable Error Interrupt 11<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 10<br>EEI10 | Enable Error Interrupt 10<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 9<br>EEI9 | Enable Error Interrupt 9<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 8<br>EEI8 | Enable Error Interrupt 8<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 7<br>EEI7 | Enable Error Interrupt 7<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 6<br>EEI6 | Enable Error Interrupt 6<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5<br>EEI5 | Enable Error Interrupt 5<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 4<br>EEI4 | Enable Error Interrupt 4<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 3<br>EEI3 | Enable Error Interrupt 3<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 2<br>EEI2 | Enable Error Interrupt 2<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 1<br>EEI1 | Enable Error Interrupt 1<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 0<br>EEI0 | Enable Error Interrupt 0<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |

## 38.4.5.6 Clear Enable Error Interrupt Register (CEEI)

### 38.4.5.6.1 Offset

| Register | Offset |
|---|---|
| CEEI | 18h |

### 38.4.5.6.2 Function

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.6.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | CAEE | 0 | | CEEI | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.6.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAEE | Clear All Enable Error Interrupts<br>0b - Clear only the EEI bit specified in the CEEI field<br>1b - Clear all bits in EEI |
| 5-4<br>— | Reserved |
| 3-0<br>CEEI | Clear Enable Error Interrupt<br>Clears the corresponding bit in EEI |

## 38.4.5.7 Set Enable Error Interrupt Register (SEEI)

### 38.4.5.7.1 Offset

| Register | Offset |
|---|---|
| SEEI | 19h |

### 38.4.5.7.2 Function

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.7.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | SAEE | 0 | | SEEI | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.7.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>SAEE | Sets All Enable Error Interrupts<br>0b - Set only the EEI bit specified in the SEEI field.<br>1b - Sets all bits in EEI |
| 5-4<br>— | Reserved |
| 3-0<br>SEEI | Set Enable Error Interrupt<br>Sets the corresponding bit in EEI |

## 38.4.5.8 Clear Enable Request Register (CERQ)

### 38.4.5.8.1 Offset

| Register | Offset |
|---|---|
| CERQ | 1Ah |

### 38.4.5.8.2 Function

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

**NOTE**

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

### 38.4.5.8.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | CAER | 0 | | CERQ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.8.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAER | Clear All Enable Requests<br>0b - Clear only the ERQ bit specified in the CERQ field<br>1b - Clear all bits in ERQ |
| 5-4<br>— | Reserved |
| 3-0<br>CERQ | Clear Enable Request<br>Clears the corresponding bit in ERQ. |

## 38.4.5.9 Set Enable Request Register (SERQ)

### 38.4.5.9.1 Offset

| Register | Offset |
|---|---|
| SERQ | 1Bh |

### 38.4.5.9.2 Function

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.9.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | SAER | 0 | | SERQ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.9.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>SAER | Set All Enable Requests<br>0b - Set only the ERQ bit specified in the SERQ field<br>1b - Set all bits in ERQ |
| 5-4<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>SERQ | Set Enable Request<br>Sets the corresponding bit in ERQ. |

## 38.4.5.10 Clear DONE Status Bit Register (CDNE)

### 38.4.5.10.1 Offset

| Register | Offset |
|---|---|
| CDNE | 1Ch |

### 38.4.5.10.2 Function

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.10.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | CADN | 0 | | CDNE | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.10.4 Fields

| Field | Function |
|---|---|
| 7 | No Op enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| NOP | 0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CADN | Clears All DONE Bits<br>0b - Clears only the TCDn_CSR[DONE] bit specified in the CDNE field<br>1b - Clears all bits in TCDn_CSR[DONE] |
| 5-4<br>— | Reserved |
| 3-0<br>CDNE | Clear DONE Bit<br>Clears the corresponding bit in TCDn_CSR[DONE] |

## 38.4.5.11 Set START Bit Register (SSRT)

### 38.4.5.11.1 Offset

| Register | Offset |
|---|---|
| SSRT | 1Dh |

### 38.4.5.11.2 Function

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.11.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | SAST | 0 | | SSRT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 38.4.5.11.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>SAST | Set All START Bits (activates all channels)<br>0b - Set only the TCDn_CSR[START] bit specified in the SSRT field<br>1b - Set all bits in TCDn_CSR[START] |
| 5-4<br>— | Reserved |
| 3-0<br>SSRT | Set START Bit<br>Sets the corresponding bit in TCDn_CSR[START] |

### 38.4.5.12 Clear Error Register (CERR)

#### 38.4.5.12.1 Offset

| Register | Offset |
|---|---|
| CERR | 1Eh |

#### 38.4.5.12.2 Function

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

#### 38.4.5.12.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | CAEI | 0 | | CER | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 38.4.5.12.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAEI | Clear All Error Indicators<br>0b - Clear only the ERR bit specified in the CERR field<br>1b - Clear all bits in ERR |
| 5-4<br>— | Reserved |
| 3-0<br>CERR | Clear Error Indicator<br>Clears the corresponding bit in ERR |

### 38.4.5.13 Clear Interrupt Request Register (CINT)

#### 38.4.5.13.1 Offset

| Register | Offset |
|---|---|
| CINT | 1Fh |

#### 38.4.5.13.2 Function

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.5.13.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | 0 | | | |
| W | NOP | CAIR | 0 | | CINT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.13.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAIR | Clear All Interrupt Requests<br>0b - Clear only the INT bit specified in the CINT field<br>1b - Clear all bits in INT |
| 5-4<br>— | Reserved |
| 3-0<br>CINT | Clear Interrupt Request<br>Clears the corresponding bit in INT |

## 38.4.5.14 Interrupt Request Register (INT)

### 38.4.5.14.1 Offset

| Register | Offset |
|---|---|
| INT | 24h |

### 38.4.5.14.2 Function

The INT register provides a bit map for the 16 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

### 38.4.5.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INT15 | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 | INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 | INT0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.14.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>INT15 | Interrupt Request 15<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 14<br>INT14 | Interrupt Request 14<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 13<br>INT13 | Interrupt Request 13<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 12<br>INT12 | Interrupt Request 12<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 11<br>INT11 | Interrupt Request 11<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 10<br>INT10 | Interrupt Request 10<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 9<br>INT9 | Interrupt Request 9<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 8<br>INT8 | Interrupt Request 8<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 7<br>INT7 | Interrupt Request 7<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 6<br>INT6 | Interrupt Request 6<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 5<br>INT5 | Interrupt Request 5<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 4<br>INT4 | Interrupt Request 4<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 3<br>INT3 | Interrupt Request 3<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 2<br>INT2 | Interrupt Request 2<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 1<br>INT1 | Interrupt Request 1<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 0<br>INT0 | Interrupt Request 0<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |

## 38.4.5.15 Error Register (ERR)

### 38.4.5.15.1 Offset

| Register | Offset |
|---|---|
| ERR | 2Ch |

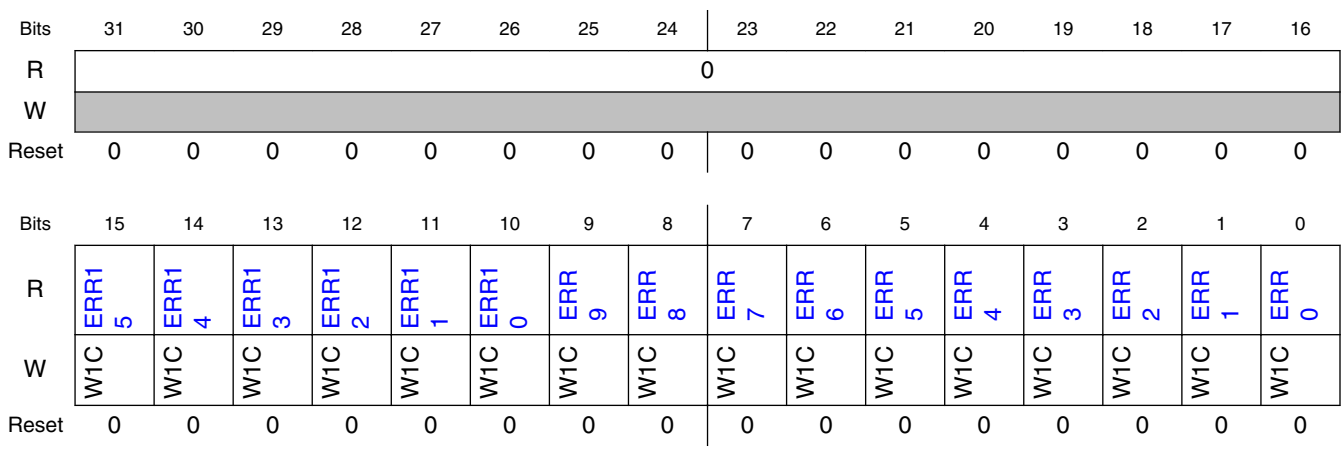
### 38.4.5.15.2 Function

The ERR register provides a bit map for the 16 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the

contents of the EEI register, and then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

### 38.4.5.15.3 Diagram




### 38.4.5.15.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>ERR15 | Error In Channel 15<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 14<br>ERR14 | Error In Channel 14<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 13 | Error In Channel 13 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ERR13 | 0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 12<br>ERR12 | Error In Channel 12<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 11<br>ERR11 | Error In Channel 11<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 10<br>ERR10 | Error In Channel 10<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 9<br>ERR9 | Error In Channel 9<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 8<br>ERR8 | Error In Channel 8<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 7<br>ERR7 | Error In Channel 7<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 6<br>ERR6 | Error In Channel 6<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 5<br>ERR5 | Error In Channel 5<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 4<br>ERR4 | Error In Channel 4<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 3<br>ERR3 | Error In Channel 3<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 2<br>ERR2 | Error In Channel 2<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 1<br>ERR1 | Error In Channel 1<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 0<br>ERR0 | Error In Channel 0<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |

## 38.4.5.16 Hardware Request Status Register (HRS)

### 38.4.5.16.1 Offset

| Register | Offset |
|---|---|
| HRS | 34h |

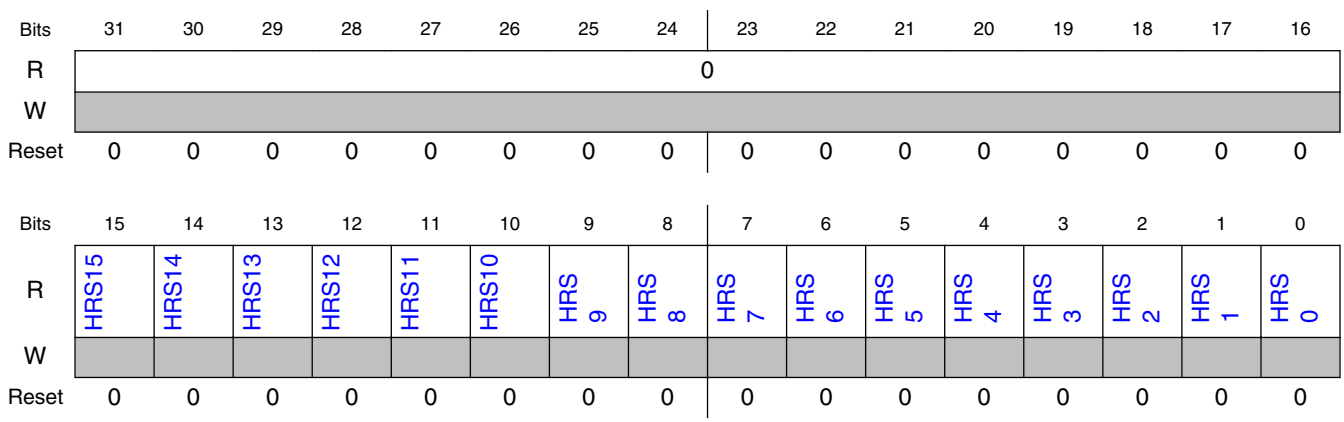
### 38.4.5.16.2 Function

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

**NOTE**

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

### 38.4.5.16.3 Diagram




### 38.4.5.16.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>HRS15 | Hardware Request Status Channel 15<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - A hardware service request for channel 15 is not present<br>1b - A hardware service request for channel 15 is present |
| 14<br>HRS14 | Hardware Request Status Channel 14<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 14 is not present<br>1b - A hardware service request for channel 14 is present |
| 13<br>HRS13 | Hardware Request Status Channel 13<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 13 is not present<br>1b - A hardware service request for channel 13 is present |
| 12<br>HRS12 | Hardware Request Status Channel 12<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 12 is not present<br>1b - A hardware service request for channel 12 is present |
| 11<br>HRS11 | Hardware Request Status Channel 11<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 11 is not present<br>1b - A hardware service request for channel 11 is present |
| 10<br>HRS10 | Hardware Request Status Channel 10<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 10 is not present<br>1b - A hardware service request for channel 10 is present |
| 9<br>HRS9 | Hardware Request Status Channel 9<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 9 is not present<br>1b - A hardware service request for channel 9 is present |
| 8<br>HRS8 | Hardware Request Status Channel 8<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 8 is not present<br>1b - A hardware service request for channel 8 is present |
| 7<br>HRS7 | Hardware Request Status Channel 7<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 7 is not present<br>1b - A hardware service request for channel 7 is present |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 6<br>HRS6 | Hardware Request Status Channel 6<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 6 is not present<br>1b - A hardware service request for channel 6 is present |
| 5<br>HRS5 | Hardware Request Status Channel 5<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 5 is not present<br>1b - A hardware service request for channel 5 is present |
| 4<br>HRS4 | Hardware Request Status Channel 4<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 4 is not present<br>1b - A hardware service request for channel 4 is present |
| 3<br>HRS3 | Hardware Request Status Channel 3<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 3 is not present<br>1b - A hardware service request for channel 3 is present |
| 2<br>HRS2 | Hardware Request Status Channel 2<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 2 is not present<br>1b - A hardware service request for channel 2 is present |
| 1<br>HRS1 | Hardware Request Status Channel 1<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 1 is not present<br>1b - A hardware service request for channel 1 is present |
| 0<br>HRS0 | Hardware Request Status Channel 0<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 0 is not present<br>1b - A hardware service request for channel 0 is present |

## 38.4.5.17 Enable Asynchronous Request in Stop Register (EARS)

### 38.4.5.17.1 Offset

| Register | Offset |
|---|---|
| EARS | 44h |

### 38.4.5.17.2 Function

The EARS register is used to enable or disable the DMA requests in Enable Request Register (ERQ) by AND'ing the bits of these two registers.

### 38.4.5.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EDREQ_15 | EDREQ_14 | EDREQ_13 | EDREQ_12 | EDREQ_11 | EDREQ_10 | EDREQ_9 | EDREQ_8 | EDREQ_7 | EDREQ_6 | EDREQ_5 | EDREQ_4 | EDREQ_3 | EDREQ_2 | EDREQ_1 | EDREQ_0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.5.17.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>EDREQ_15 | Enable asynchronous DMA request in stop mode for channel 15<br>0b - Disable asynchronous DMA request for channel 15.<br>1b - Enable asynchronous DMA request for channel 15. |
| 14<br>EDREQ_14 | Enable asynchronous DMA request in stop mode for channel 14<br>0b - Disable asynchronous DMA request for channel 14.<br>1b - Enable asynchronous DMA request for channel 14. |
| 13<br>EDREQ_13 | Enable asynchronous DMA request in stop mode for channel 13<br>0b - Disable asynchronous DMA request for channel 13.<br>1b - Enable asynchronous DMA request for channel 13. |
| 12<br>EDREQ_12 | Enable asynchronous DMA request in stop mode for channel 12<br>0b - Disable asynchronous DMA request for channel 12.<br>1b - Enable asynchronous DMA request for channel 12. |
| 11 | Enable asynchronous DMA request in stop mode for channel 11<br>0b - Disable asynchronous DMA request for channel 11. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| EDREQ_11 | 1b - Enable asynchronous DMA request for channel 11. |
| 10<br>EDREQ_10 | Enable asynchronous DMA request in stop mode for channel 10<br>0b - Disable asynchronous DMA request for channel 10.<br>1b - Enable asynchronous DMA request for channel 10. |
| 9<br>EDREQ_9 | Enable asynchronous DMA request in stop mode for channel 9<br>0b - Disable asynchronous DMA request for channel 9.<br>1b - Enable asynchronous DMA request for channel 9. |
| 8<br>EDREQ_8 | Enable asynchronous DMA request in stop mode for channel 8<br>0b - Disable asynchronous DMA request for channel 8.<br>1b - Enable asynchronous DMA request for channel 8. |
| 7<br>EDREQ_7 | Enable asynchronous DMA request in stop mode for channel 7<br>0b - Disable asynchronous DMA request for channel 7.<br>1b - Enable asynchronous DMA request for channel 7. |
| 6<br>EDREQ_6 | Enable asynchronous DMA request in stop mode for channel 6<br>0b - Disable asynchronous DMA request for channel 6.<br>1b - Enable asynchronous DMA request for channel 6. |
| 5<br>EDREQ_5 | Enable asynchronous DMA request in stop mode for channel 5<br>0b - Disable asynchronous DMA request for channel 5.<br>1b - Enable asynchronous DMA request for channel 5. |
| 4<br>EDREQ_4 | Enable asynchronous DMA request in stop mode for channel 4<br>0b - Disable asynchronous DMA request for channel 4.<br>1b - Enable asynchronous DMA request for channel 4. |
| 3<br>EDREQ_3 | Enable asynchronous DMA request in stop mode for channel 3.<br>0b - Disable asynchronous DMA request for channel 3.<br>1b - Enable asynchronous DMA request for channel 3. |
| 2<br>EDREQ_2 | Enable asynchronous DMA request in stop mode for channel 2.<br>0b - Disable asynchronous DMA request for channel 2.<br>1b - Enable asynchronous DMA request for channel 2. |
| 1<br>EDREQ_1 | Enable asynchronous DMA request in stop mode for channel 1.<br>0b - Disable asynchronous DMA request for channel 1<br>1b - Enable asynchronous DMA request for channel 1. |
| 0<br>EDREQ_0 | Enable asynchronous DMA request in stop mode for channel 0.<br>0b - Disable asynchronous DMA request for channel 0.<br>1b - Enable asynchronous DMA request for channel 0. |

## 38.4.5.18 Channel Priority Register (DCHPRI3 - DCHPRI12)

### 38.4.5.18.1 Offset

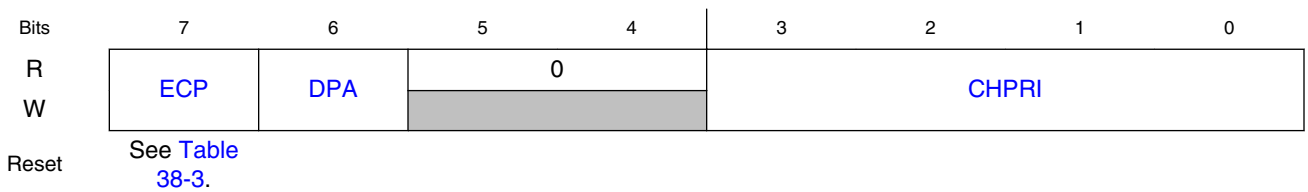
For n = 0 to 15:

| Register | Offset |
|---|---|
| DCHPRIn | FDh + (n × 1h) |

### 38.4.5.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

### 38.4.5.18.3 Diagram




### 38.4.5.18.4 Register reset values

| Register | Reset |
|---|---|
| DCHPRI3 | 03h |
| DCHPRI2 | 02h |
| DCHPRI1 | 01h |
| DCHPRI0 | 00h |
| DCHPRI7 | 07h |
| DCHPRI6 | 06h |
| DCHPRI5 | 05h |
| DCHPRI4 | 04h |
| DCHPRI11 | 0Bh |
| DCHPRI10 | 0Ah |
| DCHPRI9 | 09h |
| DCHPRI8 | 08h |
| DCHPRI15 | 0Fh |
| DCHPRI14 | 0Eh |
| DCHPRI13 | 0Dh |
| DCHPRI12 | 0Ch |

#### 38.4.5.18.5 Fields

| Field | Function |
|---|---|
| 7<br>ECP | Enable Channel Preemption. This field resets to 0.<br>0b - Channel n cannot be suspended by a higher priority channel's service request.<br>1b - Channel n can be temporarily suspended by the service request of a higher priority channel. |
| 6<br>DPA | Disable Preempt Ability. This field resets to 0.<br>0b - Channel n can suspend a lower priority channel.<br>1b - Channel n cannot suspend any channel, regardless of channel priority. |
| 5-4<br>— | Reserved |
| 3-0<br>CHPRI | Channel n Arbitration Priority<br>Channel priority when fixed-priority arbitration is enabled |

### 38.4.5.19 TCD Source Address (TCD0_SADDR - TCD15_SADDR)

#### 38.4.5.19.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_SADDR | 1000h + (n × 20h) |

#### 38.4.5.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 38.4.5.19.3 Fields

| Field | Function |
|---|---|
| 31-0 | Source Address |

| Field | Function |
|---|---|
| SADDR | Memory address pointing to the source data. |

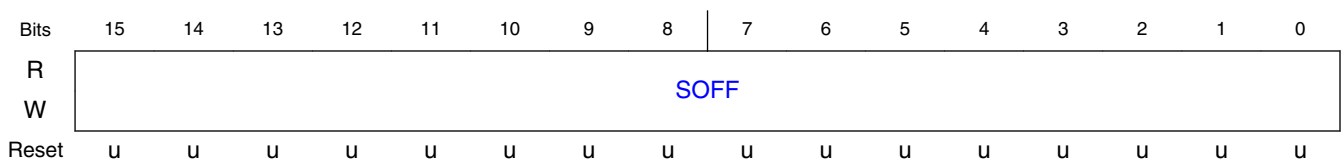
## 38.4.5.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)

### 38.4.5.20.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_SOFF | 1004h + (n × 20h) |

### 38.4.5.20.2 Diagram




### 38.4.5.20.3 Fields

| Field | Function |
|---|---|
| 15-0<br>SOFF | Source address signed offset<br>Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

## 38.4.5.21 TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)

### 38.4.5.21.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_ATTR | 1006h + (n × 20h) |

### 38.4.5.21.2 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SMOD | | | | | SSIZE | | | DMOD | | | | | DSIZE | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.5.21.3 Fields

| Field | Function |
|---|---|
| 15-11<br>SMOD | Source Address Modulo<br>00000b - Source address modulo feature is disabled<br>00001-11111b - This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. |
| 10-8<br>SSIZE | Source data transfer size<br>**NOTE:** Using a Reserved value causes a configuration error.<br>**NOTE:** The eDMA defaults to privileged data access for all transactions.<br>000b - 8-bit<br>001b - 16-bit<br>010b - 32-bit<br>011b - Reserved<br>100b - 16-byte burst<br>101b - 32-byte burst<br>110b - Reserved<br>111b - Reserved |
| 7-3<br>DMOD | Destination Address Modulo<br>See the SMOD definition |
| 2-0<br>DSIZE | Destination data transfer size<br>See the SSIZE definition |

## 38.4.5.22 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)

### 38.4.5.22.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLOFFNO | 1008h + (n × 20h) |

### 38.4.5.22.2 Function

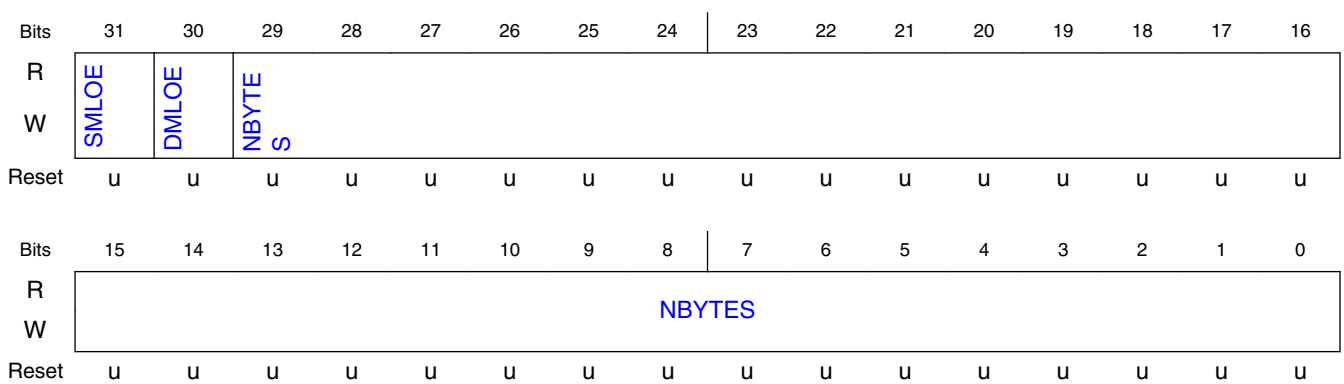
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

### 38.4.5.22.3 Diagram




### 38.4.5.22.4 Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - The minor loop offset is not applied to the SADDR<br>1b - The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - The minor loop offset is not applied to the DADDR<br>1b - The minor loop offset is applied to the DADDR |
| 29-0<br>NBYTES | Minor Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 38.4.5.23 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)

### 38.4.5.23.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLOFFYES | 1008h + (n × 20h) |

### 38.4.5.23.2 Function

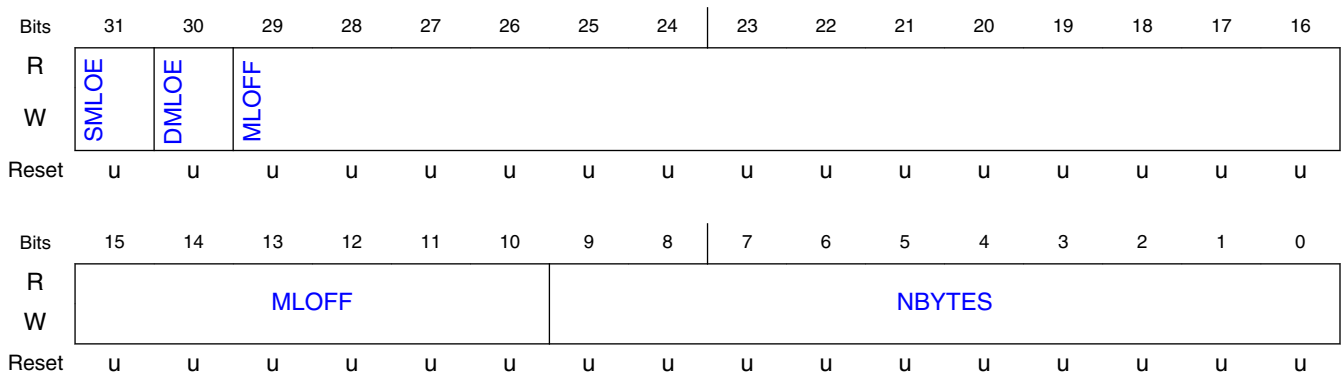
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

### 38.4.5.23.3 Diagram




### 38.4.5.23.4 Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br>0b - The minor loop offset is not applied to the SADDR<br>1b - The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - The minor loop offset is not applied to the DADDR<br>1b - The minor loop offset is applied to the DADDR |
| 29-10<br>MLOFF | If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes. |
| 9-0<br>NBYTES | Minor Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 38.4.5.24 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD15_NBYTES_MLNO)

### 38.4.5.24.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLNO | 1008h + (n × 20h) |

### 38.4.5.24.2 Function

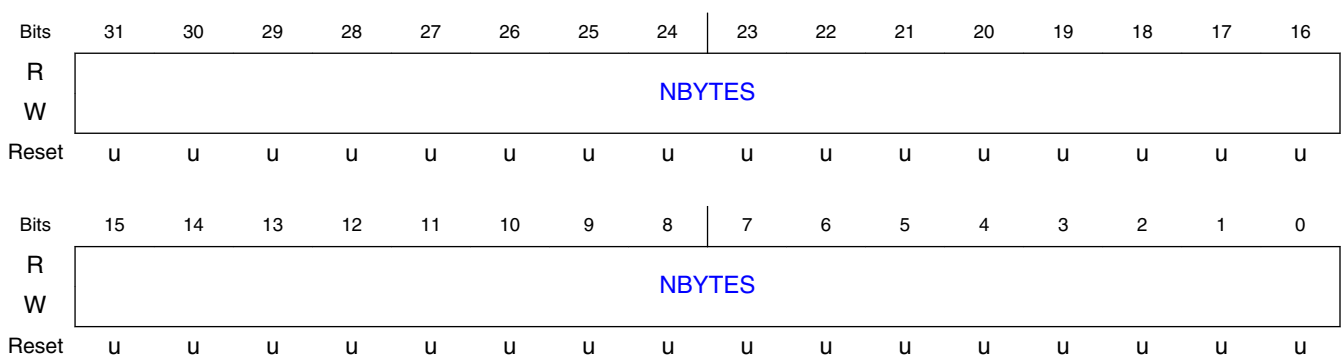
This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

### 38.4.5.24.3 Diagram




### 38.4.5.24.4 Fields

| Field | Function |
|---|---|
| 31-0<br>NBYTES | Minor Byte Transfer Count |

| Field | Function |
|---|---|
| | Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.<br>**NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer. |

## 38.4.5.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD15_SLAST)

### 38.4.5.25.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_SLAST | 100Ch + (n × 20h) |

### 38.4.5.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.5.25.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SLAST | Last Source Address Adjustment<br>Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.<br>This register uses two's complement notation; the overflow bit is discarded. |

## 38.4.5.26 TCD Destination Address (TCD0_DADDR - TCD15_DADDR)

### 38.4.5.26.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_DADDR | 1010h + (n × 20h) |

### 38.4.5.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.5.26.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DADDR | Destination Address<br>Memory address pointing to the destination data. |

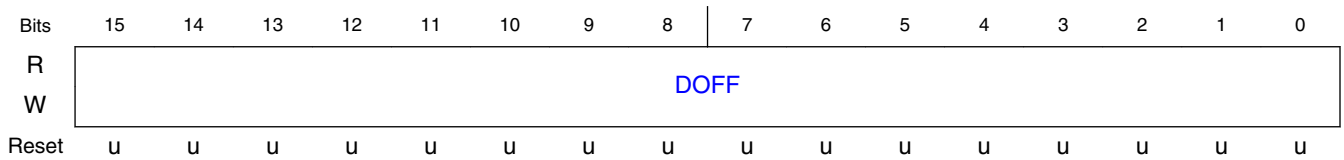
## 38.4.5.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)

### 38.4.5.27.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_DOFF | 1014h + (n × 20h) |

### 38.4.5.27.2 Diagram




### 38.4.5.27.3 Fields

| Field | Function |
|---|---|
| 15-0<br>DOFF | Destination Address Signed Offset<br>Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed. |

## 38.4.5.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)

### 38.4.5.28.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_CITER_ELINKNO | 1016h + (n × 20h) |

### 38.4.5.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is cleared, this register is defined as follows.

### 38.4.5.28.3 Diagram

### 38.4.5.28.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-0<br>CITER | Current Major Iteration Count<br>This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br>**NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)

### 38.4.5.29.1 Offset

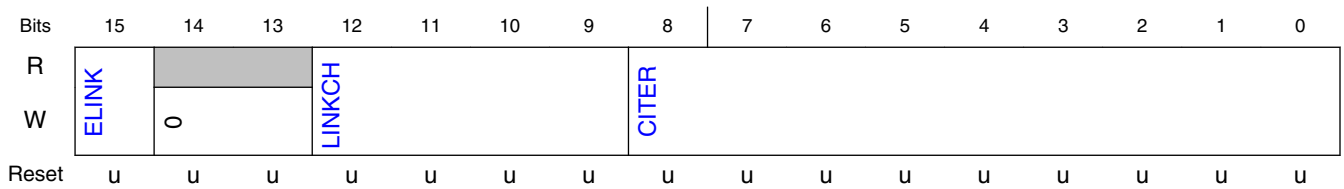
For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_CITER_ELINKYES | 1016h + (n × 20h) |

### 38.4.5.29.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is set, this register is defined as follows.

### 38.4.5.29.3 Diagram




### 38.4.5.29.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-13<br>— | Reserved |
| 12-9<br>LINKCH | Minor Loop Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit. |
| 8-0<br>CITER | Current Major Iteration Count<br>This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. |

| Field | Function |
|---|---|
| | **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD15_DLASTSGA)

### 38.4.5.30.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_DLASTSGA | 1018h + (n × 20h) |

### 38.4.5.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLASTSGA | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLASTSGA | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.5.30.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DLASTSGA | DLASTSGA<br>Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).<br>If (TCDn_CSR[ESG] = 0) then:<br>• Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.<br>• This field uses two's complement notation for the final destination address adjustment.<br>Otherwise: |

| Field | Function |
|---|---|
| | • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported. |

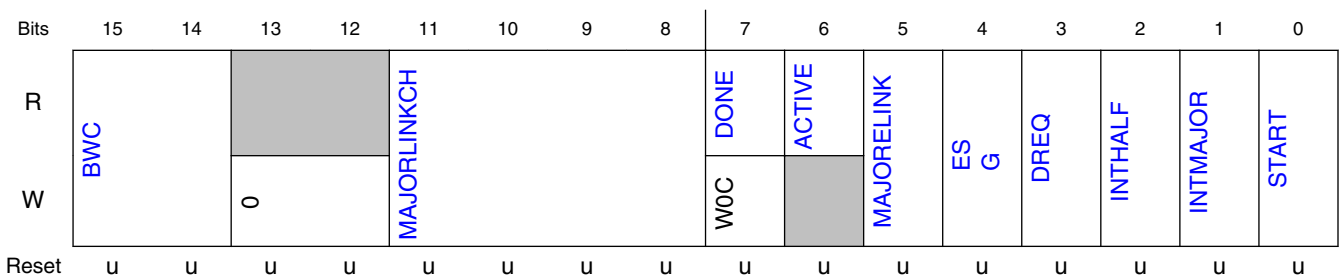
## 38.4.5.31 TCD Control and Status (TCD0_CSR - TCD15_CSR)

### 38.4.5.31.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_CSR | 101Ch + (n × 20h) |

### 38.4.5.31.2 Diagram




### 38.4.5.31.3 Fields

| Field | Function |
|---|---|
| 15-14 | Bandwidth Control |
| BWC | Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch.<br>**NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.<br>00b - No eDMA engine stalls.<br>01b - Reserved<br>10b - eDMA engine stalls for 4 cycles after each R/W.<br>11b - eDMA engine stalls for 8 cycles after each R/W. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13-12<br>— | Reserved |
| 11-8<br>MAJORLINKCH | Major Loop Link Channel Number<br>If (MAJORELINK = 0) then:<br>• No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted.<br>Otherwise:<br>• After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. |
| 7<br>DONE | Channel Done<br>This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated.<br>**NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits. |
| 6<br>ACTIVE | Channel Active<br>This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected. |
| 5<br>MAJORELINK | Enable channel-to-channel linking on major loop complete<br>As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>**NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.<br>0b - The channel-to-channel linking is disabled.<br>1b - The channel-to-channel linking is enabled. |
| 4<br>ESG | Enable Scatter/Gather Processing<br>As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory.<br>**NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.<br>0b - The current channel's TCD is normal format.<br>1b - The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution. |
| 3<br>DREQ | Disable Request<br>If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero.<br>0b - The channel's ERQ bit is not affected.<br>1b - The channel's ERQ bit is cleared when the major loop is complete. |
| 2<br>INTHALF | Enable an interrupt when major counter is half complete.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress.<br>**NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead.<br>0b - The half-point interrupt is disabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - The half-point interrupt is enabled. |
| 1<br>INTMAJOR | Enable an interrupt when major iteration count completes.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero.<br>0b - The end-of-major loop interrupt is disabled.<br>1b - The end-of-major loop interrupt is enabled. |
| 0<br>START | Channel Start<br>If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.<br>0b - The channel is not explicitly started.<br>1b - The channel is explicitly started via a software initiated service request. |

## 38.4.5.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)

### 38.4.5.32.1 Offset

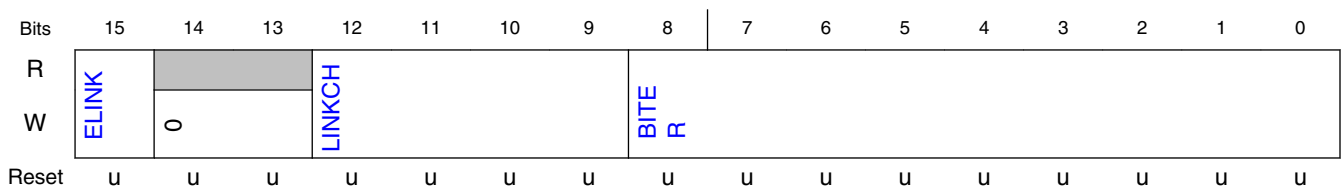
For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_BITER_ELINKYES | 101Eh + (n × 20h) |

### 38.4.5.32.2 Function

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

### 38.4.5.32.3 Diagram



### 38.4.5.32.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-13<br>— | Reserved |
| 12-9<br>LINKCH | Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. |
| 8-0<br>BITER | Starting major iteration count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.5.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)

### 38.4.5.33.1 Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| TCDn_BITER_ELINKNO | 101Eh + (n × 20h) |

### 38.4.5.33.2 Function

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

### 38.4.5.33.3 Diagram

### 38.4.5.33.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-0<br>BITER | Starting Major Iteration Count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.6 DMA1 register descriptions

### 38.4.6.1 DMA1 Memory map

DMA1 base address: 4100_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control Register (CR) | 32 | RW | See description. |
| 4h | Error Status Register (ES) | 32 | RO | 0000_0000h |
| Ch | Enable Request Register (ERQ) | 32 | RW | 0000_0000h |
| 14h | Enable Error Interrupt Register (EEI) | 32 | RW | 0000_0000h |
| 18h | Clear Enable Error Interrupt Register (CEEI) | 8 | WORZ | 00h |
| 19h | Set Enable Error Interrupt Register (SEEI) | 8 | WORZ | 00h |
| 1Ah | Clear Enable Request Register (CERQ) | 8 | WORZ | 00h |
| 1Bh | Set Enable Request Register (SERQ) | 8 | WORZ | 00h |
| 1Ch | Clear DONE Status Bit Register (CDNE) | 8 | WORZ | 00h |
| 1Dh | Set START Bit Register (SSRT) | 8 | WORZ | 00h |
| 1Eh | Clear Error Register (CERR) | 8 | WORZ | 00h |
| 1Fh | Clear Interrupt Request Register (CINT) | 8 | WORZ | 00h |
| 24h | Interrupt Request Register (INT) | 32 | W1C | 0000_0000h |
| 2Ch | Error Register (ERR) | 32 | W1C | 0000_0000h |
| 34h | Hardware Request Status Register (HRS) | 32 | RO | 0000_0000h |
| 44h | Enable Asynchronous Request in Stop Register (EARS) | 32 | RW | 0000_0000h |
| 100h | Channel Priority Register (DCHPRI3) | 8 | RW | 03h |
| 101h | Channel Priority Register (DCHPRI2) | 8 | RW | 02h |
| 102h | Channel Priority Register (DCHPRI1) | 8 | RW | 01h |
| 103h | Channel Priority Register (DCHPRI0) | 8 | RW | 00h |
| 104h | Channel Priority Register (DCHPRI7) | 8 | RW | 07h |
| 105h | Channel Priority Register (DCHPRI6) | 8 | RW | 06h |
| 106h | Channel Priority Register (DCHPRI5) | 8 | RW | 05h |
| 107h | Channel Priority Register (DCHPRI4) | 8 | RW | 04h |
| 1000h | TCD Source Address (TCD0_SADDR) | 32 | RW | See description. |
| 1004h | TCD Signed Source Address Offset (TCD0_SOFF) | 16 | RW | See description. |
| 1006h | TCD Transfer Attributes (TCD0_ATTR) | 16 | RW | See description. |
| 1008h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO) | 32 | RW | See description. |
| 1008h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1008h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 100Ch | TCD Last Source Address Adjustment (TCD0_SLAST) | 32 | RW | See description. |
| 1010h | TCD Destination Address (TCD0_DADDR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1014h | TCD Signed Destination Address Offset (TCD0_DOFF) | 16 | RW | See description. |
| 1016h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO) | 16 | RW | See description. |
| 1016h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES) | 16 | RW | See description. |
| 1018h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA) | 32 | RW | See description. |
| 101Ch | TCD Control and Status (TCD0_CSR) | 16 | RW | See description. |
| 101Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO) | 16 | RW | See description. |
| 101Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES) | 16 | RW | See description. |
| 1020h | TCD Source Address (TCD1_SADDR) | 32 | RW | See description. |
| 1024h | TCD Signed Source Address Offset (TCD1_SOFF) | 16 | RW | See description. |
| 1026h | TCD Transfer Attributes (TCD1_ATTR) | 16 | RW | See description. |
| 1028h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD1_NBYTES_MLNO) | 32 | RW | See description. |
| 1028h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD1_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1028h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD1_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 102Ch | TCD Last Source Address Adjustment (TCD1_SLAST) | 32 | RW | See description. |
| 1030h | TCD Destination Address (TCD1_DADDR) | 32 | RW | See description. |
| 1034h | TCD Signed Destination Address Offset (TCD1_DOFF) | 16 | RW | See description. |
| 1036h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_CITER_ELINKNO) | 16 | RW | See description. |
| 1036h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_CITER_ELINKYES) | 16 | RW | See description. |
| 1038h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD1_DLASTSGA) | 32 | RW | See description. |
| 103Ch | TCD Control and Status (TCD1_CSR) | 16 | RW | See description. |
| 103Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_BITER_ELINKNO) | 16 | RW | See description. |
| 103Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_BITER_ELINKYES) | 16 | RW | See description. |
| 1040h | TCD Source Address (TCD2_SADDR) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1044h | TCD Signed Source Address Offset (TCD2_SOFF) | 16 | RW | See description. |
| 1046h | TCD Transfer Attributes (TCD2_ATTR) | 16 | RW | See description. |
| 1048h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD2_NBYTES_MLNO) | 32 | RW | See description. |
| 1048h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD2_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1048h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD2_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 104Ch | TCD Last Source Address Adjustment (TCD2_SLAST) | 32 | RW | See description. |
| 1050h | TCD Destination Address (TCD2_DADDR) | 32 | RW | See description. |
| 1054h | TCD Signed Destination Address Offset (TCD2_DOFF) | 16 | RW | See description. |
| 1056h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_CITER_ELINKNO) | 16 | RW | See description. |
| 1056h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_CITER_ELINKYES) | 16 | RW | See description. |
| 1058h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD2_DLASTSGA) | 32 | RW | See description. |
| 105Ch | TCD Control and Status (TCD2_CSR) | 16 | RW | See description. |
| 105Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_BITER_ELINKNO) | 16 | RW | See description. |
| 105Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_BITER_ELINKYES) | 16 | RW | See description. |
| 1060h | TCD Source Address (TCD3_SADDR) | 32 | RW | See description. |
| 1064h | TCD Signed Source Address Offset (TCD3_SOFF) | 16 | RW | See description. |
| 1066h | TCD Transfer Attributes (TCD3_ATTR) | 16 | RW | See description. |
| 1068h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD3_NBYTES_MLNO) | 32 | RW | See description. |
| 1068h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD3_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1068h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD3_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 106Ch | TCD Last Source Address Adjustment (TCD3_SLAST) | 32 | RW | See description. |
| 1070h | TCD Destination Address (TCD3_DADDR) | 32 | RW | See description. |
| 1074h | TCD Signed Destination Address Offset (TCD3_DOFF) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1076h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_CITER_ELINKNO) | 16 | RW | See description. |
| 1076h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_CITER_ELINKYES) | 16 | RW | See description. |
| 1078h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD3_DLASTSGA) | 32 | RW | See description. |
| 107Ch | TCD Control and Status (TCD3_CSR) | 16 | RW | See description. |
| 107Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_BITER_ELINKNO) | 16 | RW | See description. |
| 107Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_BITER_ELINKYES) | 16 | RW | See description. |
| 1080h | TCD Source Address (TCD4_SADDR) | 32 | RW | See description. |
| 1084h | TCD Signed Source Address Offset (TCD4_SOFF) | 16 | RW | See description. |
| 1086h | TCD Transfer Attributes (TCD4_ATTR) | 16 | RW | See description. |
| 1088h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD4_NBYTES_MLNO) | 32 | RW | See description. |
| 1088h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD4_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 1088h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD4_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 108Ch | TCD Last Source Address Adjustment (TCD4_SLAST) | 32 | RW | See description. |
| 1090h | TCD Destination Address (TCD4_DADDR) | 32 | RW | See description. |
| 1094h | TCD Signed Destination Address Offset (TCD4_DOFF) | 16 | RW | See description. |
| 1096h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_CITER_ELINKNO) | 16 | RW | See description. |
| 1096h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_CITER_ELINKYES) | 16 | RW | See description. |
| 1098h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD4_DLASTSGA) | 32 | RW | See description. |
| 109Ch | TCD Control and Status (TCD4_CSR) | 16 | RW | See description. |
| 109Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_BITER_ELINKNO) | 16 | RW | See description. |
| 109Eh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_BITER_ELINKYES) | 16 | RW | See description. |
| 10A0h | TCD Source Address (TCD5_SADDR) | 32 | RW | See description. |
| 10A4h | TCD Signed Source Address Offset (TCD5_SOFF) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10A6h | TCD Transfer Attributes (TCD5_ATTR) | 16 | RW | See description. |
| 10A8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD5_NBYTES_MLNO) | 32 | RW | See description. |
| 10A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD5_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10A8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD5_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10ACh | TCD Last Source Address Adjustment (TCD5_SLAST) | 32 | RW | See description. |
| 10B0h | TCD Destination Address (TCD5_DADDR) | 32 | RW | See description. |
| 10B4h | TCD Signed Destination Address Offset (TCD5_DOFF) | 16 | RW | See description. |
| 10B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_CITER_ELINKNO) | 16 | RW | See description. |
| 10B6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_CITER_ELINKYES) | 16 | RW | See description. |
| 10B8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD5_DLASTSGA) | 32 | RW | See description. |
| 10BCh | TCD Control and Status (TCD5_CSR) | 16 | RW | See description. |
| 10BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_BITER_ELINKNO) | 16 | RW | See description. |
| 10BEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_BITER_ELINKYES) | 16 | RW | See description. |
| 10C0h | TCD Source Address (TCD6_SADDR) | 32 | RW | See description. |
| 10C4h | TCD Signed Source Address Offset (TCD6_SOFF) | 16 | RW | See description. |
| 10C6h | TCD Transfer Attributes (TCD6_ATTR) | 16 | RW | See description. |
| 10C8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD6_NBYTES_MLNO) | 32 | RW | See description. |
| 10C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD6_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10C8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD6_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10CCh | TCD Last Source Address Adjustment (TCD6_SLAST) | 32 | RW | See description. |
| 10D0h | TCD Destination Address (TCD6_DADDR) | 32 | RW | See description. |
| 10D4h | TCD Signed Destination Address Offset (TCD6_DOFF) | 16 | RW | See description. |
| 10D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_CITER_ELINKNO) | 16 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10D6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_CITER_ELINKYES) | 16 | RW | See description. |
| 10D8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD6_DLASTSGA) | 32 | RW | See description. |
| 10DCh | TCD Control and Status (TCD6_CSR) | 16 | RW | See description. |
| 10DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_BITER_ELINKNO) | 16 | RW | See description. |
| 10DEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_BITER_ELINKYES) | 16 | RW | See description. |
| 10E0h | TCD Source Address (TCD7_SADDR) | 32 | RW | See description. |
| 10E4h | TCD Signed Source Address Offset (TCD7_SOFF) | 16 | RW | See description. |
| 10E6h | TCD Transfer Attributes (TCD7_ATTR) | 16 | RW | See description. |
| 10E8h | TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD7_NBYTES_MLNO) | 32 | RW | See description. |
| 10E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD7_NBYTES_MLOFFNO) | 32 | RW | See description. |
| 10E8h | TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD7_NBYTES_MLOFFYES) | 32 | RW | See description. |
| 10ECh | TCD Last Source Address Adjustment (TCD7_SLAST) | 32 | RW | See description. |
| 10F0h | TCD Destination Address (TCD7_DADDR) | 32 | RW | See description. |
| 10F4h | TCD Signed Destination Address Offset (TCD7_DOFF) | 16 | RW | See description. |
| 10F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_CITER_ELINKNO) | 16 | RW | See description. |
| 10F6h | TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_CITER_ELINKYES) | 16 | RW | See description. |
| 10F8h | TCD Last Destination Address Adjustment/Scatter Gather Address (TCD7_DLASTSGA) | 32 | RW | See description. |
| 10FCh | TCD Control and Status (TCD7_CSR) | 16 | RW | See description. |
| 10FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_BITER_ELINKNO) | 16 | RW | See description. |
| 10FEh | TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_BITER_ELINKYES) | 16 | RW | See description. |

## 38.4.6.2 Control Register (CR)

#### 38.4.6.2.1 Offset

| Register | Offset |
|---|---|
| CR | 0h |

#### 38.4.6.2.2 Function

The CR defines the basic operating configuration of the DMA.

Arbitration can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels are cycled through (from high to low channel number) without regard to priority.

**NOTE**

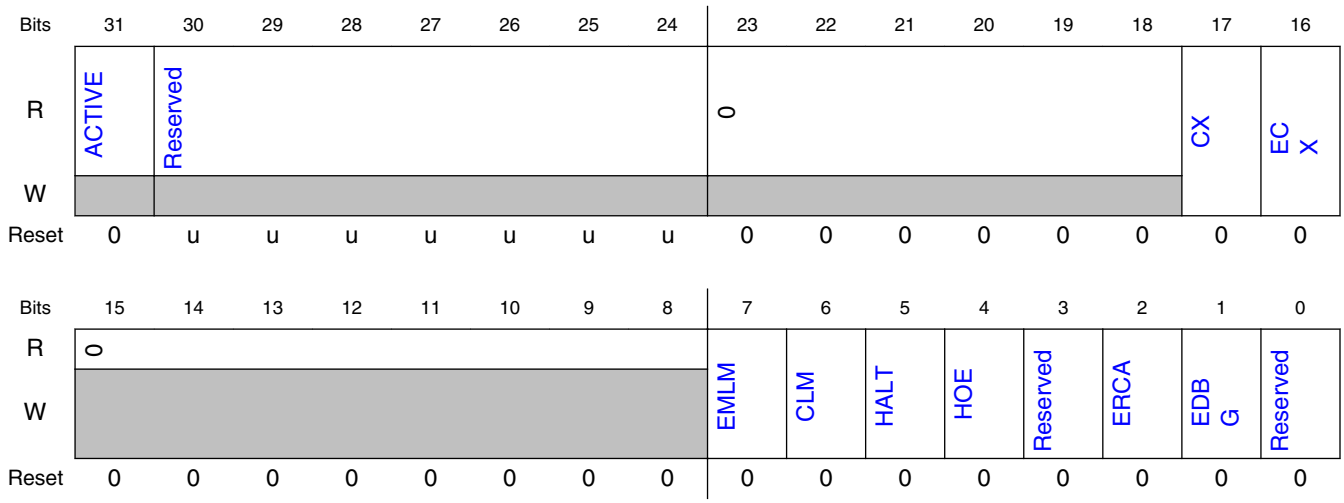
For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

### 38.4.6.2.3 Diagram




### 38.4.6.2.4 Fields

| Field | Function |
|---|---|
| 31<br>ACTIVE | DMA Active Status<br>0b - eDMA is idle.<br>1b - eDMA is executing a channel. |
| 30-24<br>— | eDMA version number<br>Reserved |
| 23-18<br>— | Reserved |
| 17<br>CX | Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed. |
| 16<br>ECX | Error Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt. |
| 15-8<br>— | Reserved |
| 7<br>EMLM | Enable Minor Loop Mapping<br>0b - Disabled. TCDn.word2 is defined as a 32-bit NBYTES field.<br>1b - Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | address, the destination address, or both. The NBYTES field is reduced when either offset is enabled. |
| 6<br>CLM | Continuous Link Mode<br>**NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing.<br>0b - A minor loop channel link made to itself goes through channel arbitration before being activated again.<br>1b - A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. |
| 5<br>HALT | Halt DMA Operations<br>0b - Normal operation<br>1b - Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared. |
| 4<br>HOE | Halt On Error<br>0b - Normal operation<br>1b - Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared. |
| 3<br>— | Reserved<br>Reserved |
| 2<br>ERCA | Enable Round Robin Channel Arbitration<br>0b - Fixed priority arbitration is used for channel selection .<br>1b - Round robin arbitration is used for channel selection . |
| 1<br>EDBG | Enable Debug<br>0b - When in debug mode, the DMA continues to operate.<br>1b - When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared. |
| 0<br>— | Reserved<br>Reserved |

## 38.4.6.3 Error Status Register (ES)

### 38.4.6.3.1 Offset

| Register | Offset |
|---|---|
| ES | 4h |

### 38.4.6.3.2 Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

### 38.4.6.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VLD | 0 | | | | | | | | | | | | | | ECX |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CPE | 0 | | | ERRCHN | | | SAE | SOE | DAE | DOE | NCE | SGE | SBE | DBE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.3.4 Fields

| Field | Function |
|---|---|
| 31<br>VLD | VLD<br>Logical OR of all ERR status bits<br>0b - No ERR bits are set.<br>1b - At least one ERR bit is set indicating a valid error exists that has not been cleared. |
| 30-17<br>— | Reserved |
| 16<br>ECX | Transfer Canceled<br>0b - No canceled transfers<br>1b - The last recorded entry was a canceled transfer by the error cancel transfer input |
| 15<br>— | Reserved |
| 14<br>CPE | Channel Priority Error<br>0b - No channel priority error |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique. |
| 13-11<br>— | Reserved |
| 10-8<br>ERRCHN | Error Channel Number or Canceled Channel Number<br>The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer. |
| 7<br>SAE | Source Address Error<br>0b - No source address configuration error.<br>1b - The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE]. |
| 6<br>SOE | Source Offset Error<br>0b - No source offset configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE]. |
| 5<br>DAE | Destination Address Error<br>0b - No destination address configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE]. |
| 4<br>DOE | Destination Offset Error<br>0b - No destination offset configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. |
| 3<br>NCE | NBYTES/CITER Configuration Error<br>0b - No NBYTES/CITER configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], orTCDn_CITER[CITER] is equal to zero, orTCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK] |
| 2<br>SGE | Scatter/Gather Configuration Error<br>0b - No scatter/gather configuration error<br>1b - The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary. |
| 1<br>SBE | Source Bus Error<br>0b - No source bus error<br>1b - The last recorded error was a bus error on a source read |
| 0<br>DBE | Destination Bus Error<br>0b - No destination bus error<br>1b - The last recorded error was a bus error on a destination write |

## 38.4.6.4 Enable Request Register (ERQ)

### 38.4.6.4.1 Offset

| Register | Offset |
|---|---|
| ERQ | Ch |

### 38.4.6.4.2 Function

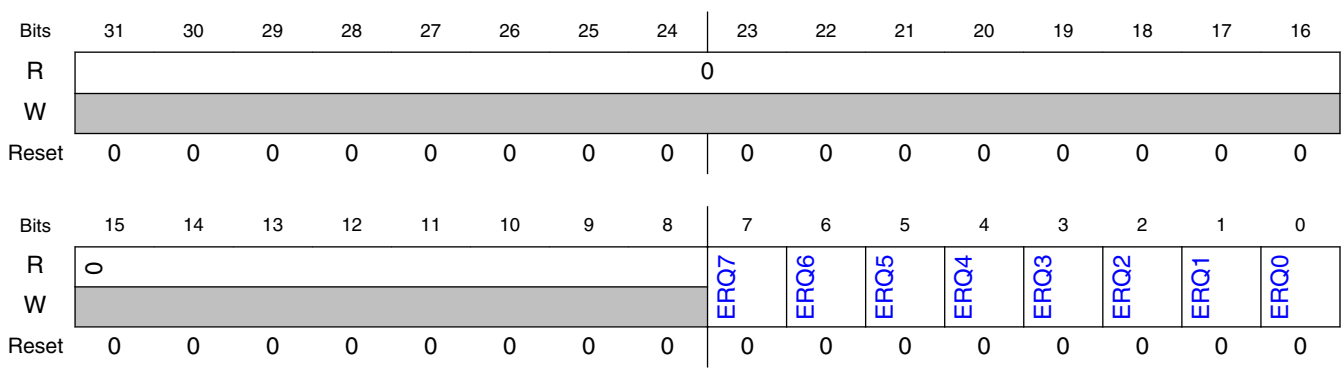
The ERQ register provides a bit map for the 8 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

**NOTE**

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

### 38.4.6.4.3 Diagram




### 38.4.6.4.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7 | Enable DMA Request 7 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ERQ7 | 0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 6<br>ERQ6 | Enable DMA Request 6<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 5<br>ERQ5 | Enable DMA Request 5<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 4<br>ERQ4 | Enable DMA Request 4<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 3<br>ERQ3 | Enable DMA Request 3<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 2<br>ERQ2 | Enable DMA Request 2<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 1<br>ERQ1 | Enable DMA Request 1<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |
| 0<br>ERQ0 | Enable DMA Request 0<br>0b - The DMA request signal for the corresponding channel is disabled<br>1b - The DMA request signal for the corresponding channel is enabled |

## 38.4.6.5 Enable Error Interrupt Register (EEI)

### 38.4.6.5.1 Offset

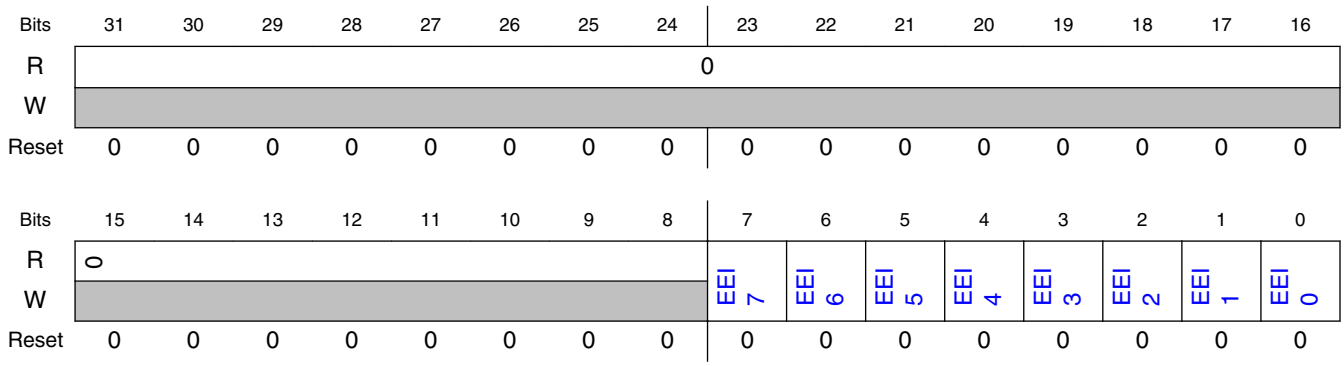
| Register | Offset |
|---|---|
| EEI | 14h |

### 38.4.6.5.2 Function

The EEI register provides a bit map for the 8 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

### 38.4.6.5.3 Diagram




### 38.4.6.5.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>EEI7 | Enable Error Interrupt 7<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 6<br>EEI6 | Enable Error Interrupt 6<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 5<br>EEI5 | Enable Error Interrupt 5<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 4<br>EEI4 | Enable Error Interrupt 4<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 3<br>EEI3 | Enable Error Interrupt 3<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 2<br>EEI2 | Enable Error Interrupt 2<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 1<br>EEI1 | Enable Error Interrupt 1<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |
| 0<br>EEI0 | Enable Error Interrupt 0<br>0b - The error signal for corresponding channel does not generate an error interrupt<br>1b - The assertion of the error signal for corresponding channel generates an error interrupt request |

## 38.4.6.6 Clear Enable Error Interrupt Register (CEEI)

### 38.4.6.6.1 Offset

| Register | Offset |
|---|---|
| CEEI | 18h |

### 38.4.6.6.2 Function

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.6.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | CAEE | 0 | | | CEEI | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.6.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAEE | Clear All Enable Error Interrupts<br>0b - Clear only the EEI bit specified in the CEEI field<br>1b - Clear all bits in EEI |
| 5-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>CEEI | Clear Enable Error Interrupt<br>Clears the corresponding bit in EEI |

## 38.4.6.7 Set Enable Error Interrupt Register (SEEI)

### 38.4.6.7.1 Offset

| Register | Offset |
|---|---|
| SEEI | 19h |

### 38.4.6.7.2 Function

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.7.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | SAEE | 0 | | | SEEI | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.7.4 Fields

| Field | Function |
|---|---|
| 7 | No Op enable<br>0b - Normal operation |

*Table continues on the next page...*

| Field | Function |
|---|---|
| NOP | 1b - No operation, ignore the other bits in this register |
| 6<br>SAEE | Sets All Enable Error Interrupts<br>0b - Set only the EEI bit specified in the SEEI field.<br>1b - Sets all bits in EEI |
| 5-3<br>— | Reserved |
| 2-0<br>SEEI | Set Enable Error Interrupt<br>Sets the corresponding bit in EEI |

## 38.4.6.8 Clear Enable Request Register (CERQ)

### 38.4.6.8.1 Offset

| Register | Offset |
|---|---|
| CERQ | 1Ah |

### 38.4.6.8.2 Function

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

**NOTE**

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

### 38.4.6.8.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | CAER | 0 | | | CERQ | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.8.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAER | Clear All Enable Requests<br>0b - Clear only the ERQ bit specified in the CERQ field<br>1b - Clear all bits in ERQ |
| 5-3<br>— | Reserved |
| 2-0<br>CERQ | Clear Enable Request<br>Clears the corresponding bit in ERQ. |

## 38.4.6.9 Set Enable Request Register (SERQ)

### 38.4.6.9.1 Offset

| Register | Offset |
|---|---|
| SERQ | 1Bh |

### 38.4.6.9.2 Function

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.9.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | SAER | 0 | | | SERQ | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.9.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>SAER | Set All Enable Requests<br>0b - Set only the ERQ bit specified in the SERQ field<br>1b - Set all bits in ERQ |
| 5-3<br>— | Reserved |
| 2-0<br>SERQ | Set Enable Request<br>Sets the corresponding bit in ERQ. |

## 38.4.6.10 Clear DONE Status Bit Register (CDNE)

### 38.4.6.10.1 Offset

| Register | Offset |
|---|---|
| CDNE | 1Ch |

### 38.4.6.10.2 Function

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.10.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | CADN | 0 | | | CDNE | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.10.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CADN | Clears All DONE Bits<br>0b - Clears only the TCDn_CSR[DONE] bit specified in the CDNE field<br>1b - Clears all bits in TCDn_CSR[DONE] |
| 5-3<br>— | Reserved |
| 2-0<br>CDNE | Clear DONE Bit<br>Clears the corresponding bit in TCDn_CSR[DONE] |

## 38.4.6.11 Set START Bit Register (SSRT)

### 38.4.6.11.1 Offset

| Register | Offset |
|---|---|
| SSRT | 1Dh |

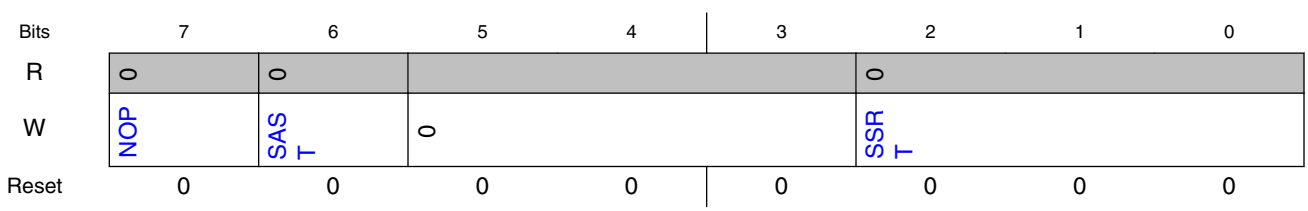
### 38.4.6.11.2 Function

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.11.3 Diagram




### 38.4.6.11.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>SAST | Set All START Bits (activates all channels)<br>0b - Set only the TCDn_CSR[START] bit specified in the SSRT field<br>1b - Set all bits in TCDn_CSR[START] |
| 5-3<br>— | Reserved |
| 2-0<br>SSRT | Set START Bit<br>Sets the corresponding bit in TCDn_CSR[START] |

## 38.4.6.12 Clear Error Register (CERR)

### 38.4.6.12.1 Offset

| Register | Offset |
|---|---|
| CERR | 1Eh |

### 38.4.6.12.2 Function

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

### 38.4.6.12.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | | |
| W | NOP | CAEI | 0 | | | CERR | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.12.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAEI | Clear All Error Indicators<br>0b - Clear only the ERR bit specified in the CERR field<br>1b - Clear all bits in ERR |
| 5-3<br>— | Reserved |
| 2-0<br>CERR | Clear Error Indicator<br>Clears the corresponding bit in ERR |

## 38.4.6.13 Clear Interrupt Request Register (CINT)

### 38.4.6.13.1 Offset

| Register | Offset |
|---|---|
| CINT | 1Fh |

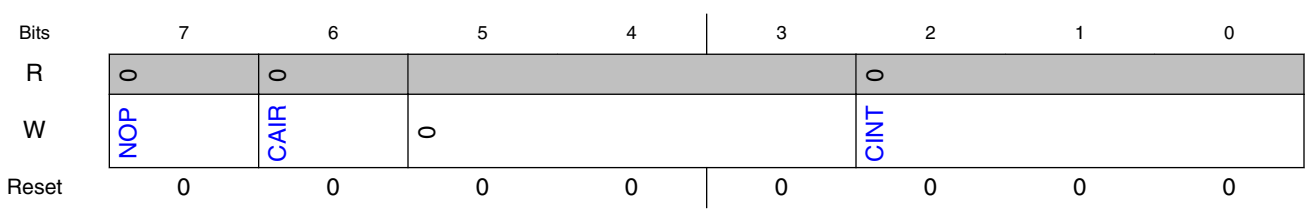
### 38.4.6.13.2 Function

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

### 38.4.6.13.3 Diagram




### 38.4.6.13.4 Fields

| Field | Function |
|---|---|
| 7<br>NOP | No Op enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| 6<br>CAIR | Clear All Interrupt Requests<br>0b - Clear only the INT bit specified in the CINT field<br>1b - Clear all bits in INT |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5-3<br>— | Reserved |
| 2-0<br>CINT | Clear Interrupt Request<br>Clears the corresponding bit in INT |

## 38.4.6.14 Interrupt Request Register (INT)

### 38.4.6.14.1 Offset

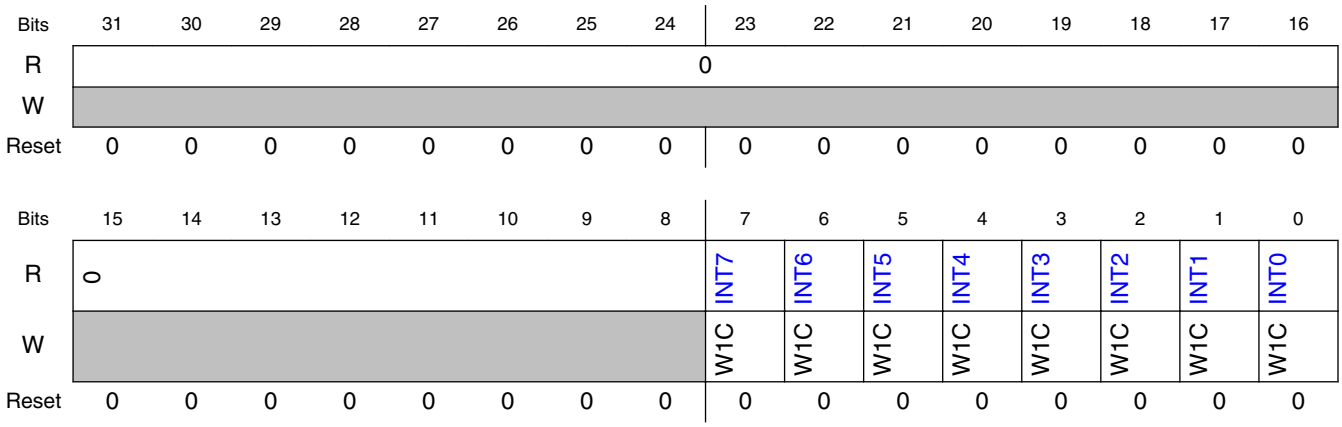
| Register | Offset |
|---|---|
| INT | 24h |

### 38.4.6.14.2 Function

The INT register provides a bit map for the 8 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

### 38.4.6.14.3 Diagram




### 38.4.6.14.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>INT7 | Interrupt Request 7<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 6<br>INT6 | Interrupt Request 6<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 5<br>INT5 | Interrupt Request 5<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 4<br>INT4 | Interrupt Request 4<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 3<br>INT3 | Interrupt Request 3<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 2<br>INT2 | Interrupt Request 2<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 1<br>INT1 | Interrupt Request 1<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |
| 0<br>INT0 | Interrupt Request 0<br>0b - The interrupt request for corresponding channel is cleared<br>1b - The interrupt request for corresponding channel is active |

## 38.4.6.15 Error Register (ERR)

### 38.4.6.15.1 Offset

| Register | Offset |
|---|---|
| ERR | 2Ch |

### 38.4.6.15.2 Function

The ERR register provides a bit map for the 8 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI register, and then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

### 38.4.6.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | ERR 7 | ERR 6 | ERR 5 | ERR 4 | ERR 3 | ERR 2 | ERR 1 | ERR 0 |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.15.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>ERR7 | Error In Channel 7<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 6<br>ERR6 | Error In Channel 6<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 5<br>ERR5 | Error In Channel 5<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 4<br>ERR4 | Error In Channel 4<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 3<br>ERR3 | Error In Channel 3<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 2<br>ERR2 | Error In Channel 2<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 1<br>ERR1 | Error In Channel 1<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 0<br>ERR0 | Error In Channel 0<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |

## 38.4.6.16 Hardware Request Status Register (HRS)

### 38.4.6.16.1 Offset

| Register | Offset |
|---|---|
| HRS | 34h |

### 38.4.6.16.2 Function

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

**NOTE**

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

### 38.4.6.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | HRS 7 | HRS 6 | HRS 5 | HRS 4 | HRS 3 | HRS 2 | HRS 1 | HRS 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.16.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>HRS7 | Hardware Request Status Channel 7<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 7 is not present<br>1b - A hardware service request for channel 7 is present |
| 6<br>HRS6 | Hardware Request Status Channel 6<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 6 is not present<br>1b - A hardware service request for channel 6 is present |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5<br>HRS5 | Hardware Request Status Channel 5<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 5 is not present<br>1b - A hardware service request for channel 5 is present |
| 4<br>HRS4 | Hardware Request Status Channel 4<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 4 is not present<br>1b - A hardware service request for channel 4 is present |
| 3<br>HRS3 | Hardware Request Status Channel 3<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 3 is not present<br>1b - A hardware service request for channel 3 is present |
| 2<br>HRS2 | Hardware Request Status Channel 2<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 2 is not present<br>1b - A hardware service request for channel 2 is present |
| 1<br>HRS1 | Hardware Request Status Channel 1<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 1 is not present<br>1b - A hardware service request for channel 1 is present |
| 0<br>HRS0 | Hardware Request Status Channel 0<br>The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b - A hardware service request for channel 0 is not present<br>1b - A hardware service request for channel 0 is present |

## 38.4.6.17 Enable Asynchronous Request in Stop Register (EARS)

### 38.4.6.17.1 Offset

| Register | Offset |
|---|---|
| EARS | 44h |

### 38.4.6.17.2 Function

The EARS register is used to enable or disable the DMA requests in Enable Request Register (ERQ) by AND'ing the bits of these two registers.

### 38.4.6.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | EDREQ_7 | EDREQ_6 | EDREQ_5 | EDREQ_4 | EDREQ_3 | EDREQ_2 | EDREQ_1 | EDREQ_0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 38.4.6.17.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>EDREQ_7 | Enable asynchronous DMA request in stop mode for channel 7<br>0b - Disable asynchronous DMA request for channel 7.<br>1b - Enable asynchronous DMA request for channel 7. |
| 6<br>EDREQ_6 | Enable asynchronous DMA request in stop mode for channel 6<br>0b - Disable asynchronous DMA request for channel 6.<br>1b - Enable asynchronous DMA request for channel 6. |
| 5<br>EDREQ_5 | Enable asynchronous DMA request in stop mode for channel 5<br>0b - Disable asynchronous DMA request for channel 5.<br>1b - Enable asynchronous DMA request for channel 5. |
| 4<br>EDREQ_4 | Enable asynchronous DMA request in stop mode for channel 4<br>0b - Disable asynchronous DMA request for channel 4.<br>1b - Enable asynchronous DMA request for channel 4. |
| 3<br>EDREQ_3 | Enable asynchronous DMA request in stop mode for channel 3.<br>0b - Disable asynchronous DMA request for channel 3.<br>1b - Enable asynchronous DMA request for channel 3. |
| 2<br>EDREQ_2 | Enable asynchronous DMA request in stop mode for channel 2.<br>0b - Disable asynchronous DMA request for channel 2.<br>1b - Enable asynchronous DMA request for channel 2. |
| 1<br>EDREQ_1 | Enable asynchronous DMA request in stop mode for channel 1.<br>0b - Disable asynchronous DMA request for channel 1<br>1b - Enable asynchronous DMA request for channel 1. |
| 0<br>EDREQ_0 | Enable asynchronous DMA request in stop mode for channel 0.<br>0b - Disable asynchronous DMA request for channel 0.<br>1b - Enable asynchronous DMA request for channel 0. |

## 38.4.6.18 Channel Priority Register (DCHPRI3 - DCHPRI4)

### 38.4.6.18.1 Offset

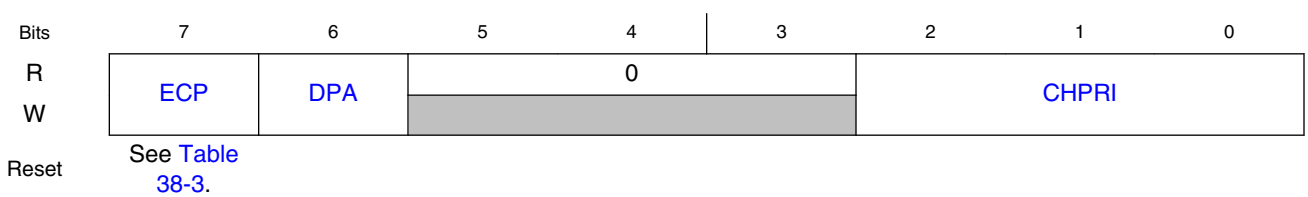
For n = 0 to 7:

| Register | Offset |
|---|---|
| DCHPRIn | FDh + (n × 1h) |

### 38.4.6.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 7.

### 38.4.6.18.3 Diagram




### 38.4.6.18.4 Register reset values

| Register | Reset |
|---|---|
| DCHPRI3 | 03h |
| DCHPRI2 | 02h |
| DCHPRI1 | 01h |
| DCHPRI0 | 00h |
| DCHPRI7 | 07h |
| DCHPRI6 | 06h |
| DCHPRI5 | 05h |
| DCHPRI4 | 04h |

### 38.4.6.18.5 Fields

| Field | Function |
|---|---|
| 7<br>ECP | Enable Channel Preemption. This field resets to 0.<br>0b - Channel n cannot be suspended by a higher priority channel's service request.<br>1b - Channel n can be temporarily suspended by the service request of a higher priority channel. |
| 6<br>DPA | Disable Preempt Ability. This field resets to 0.<br>0b - Channel n can suspend a lower priority channel.<br>1b - Channel n cannot suspend any channel, regardless of channel priority. |
| 5-3<br>— | Reserved |
| 2-0<br>CHPRI | Channel n Arbitration Priority<br>Channel priority when fixed-priority arbitration is enabled |

## 38.4.6.19 TCD Source Address (TCD0_SADDR - TCD7_SADDR)

### 38.4.6.19.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_SADDR | 1000h + (n × 20h) |

### 38.4.6.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 38.4.6.19.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SADDR | Source Address<br>Memory address pointing to the source data. |

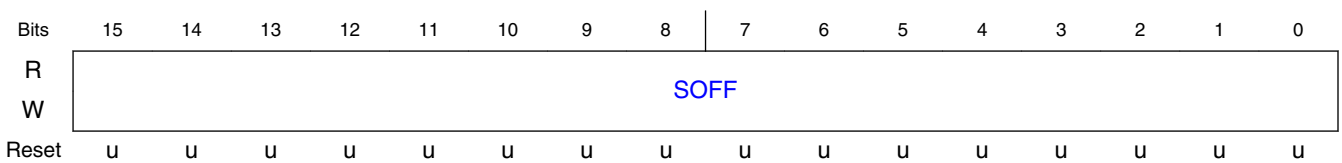
### 38.4.6.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD7_SOFF)

#### 38.4.6.20.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_SOFF | 1004h + (n × 20h) |

#### 38.4.6.20.2 Diagram




#### 38.4.6.20.3 Fields

| Field | Function |
|---|---|
| 15-0<br>SOFF | Source address signed offset<br>Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

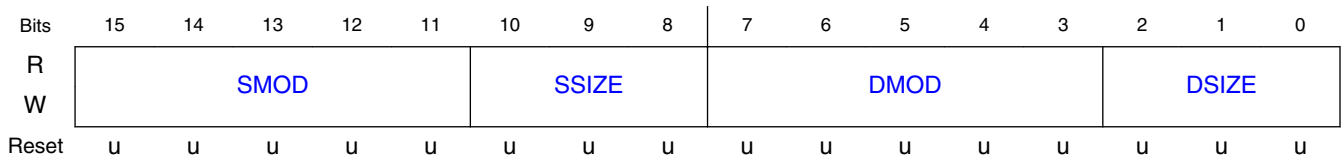
### 38.4.6.21 TCD Transfer Attributes (TCD0_ATTR - TCD7_ATTR)

#### 38.4.6.21.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_ATTR | 1006h + (n × 20h) |

## 38.4.6.21.2 Diagram




## 38.4.6.21.3 Fields

| Field | Function |
|---|---|
| 15-11<br>SMOD | Source Address Modulo<br>00000b - Source address modulo feature is disabled<br>00001-11111b - This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. |
| 10-8<br>SSIZE | Source data transfer size<br>**NOTE:** Using a Reserved value causes a configuration error.<br>**NOTE:** The eDMA defaults to privileged data access for all transactions.<br>000b - 8-bit<br>001b - 16-bit<br>010b - 32-bit<br>011b - Reserved<br>100b - 16-byte<br>101b - 32-byte<br>110b - Reserved<br>111b - Reserved |
| 7-3<br>DMOD | Destination Address Modulo<br>See the SMOD definition |
| 2-0<br>DSIZE | Destination data transfer size<br>See the SSIZE definition |

## 38.4.6.22 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD7_NBYTES_MLOFFNO)

### 38.4.6.22.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLOFFNO | 1008h + (n × 20h) |

### 38.4.6.22.2 Function

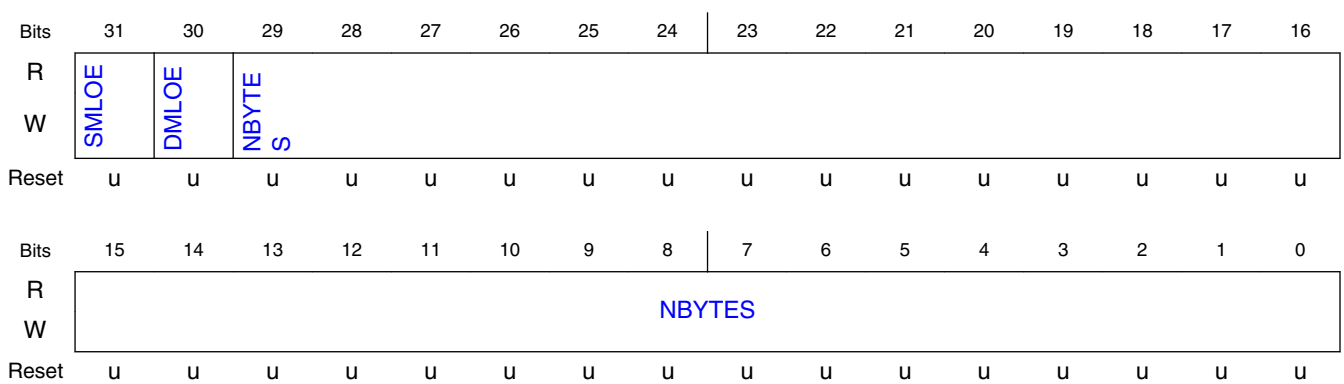
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

### 38.4.6.22.3 Diagram



### 38.4.6.22.4 Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br>0b - The minor loop offset is not applied to the SADDR<br>1b - The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - The minor loop offset is not applied to the DADDR<br>1b - The minor loop offset is applied to the DADDR |
| 29-0<br>NBYTES | Minor Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 38.4.6.23 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD7_NBYTES_MLOFFYES)

### 38.4.6.23.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLOFFYES | 1008h + (n × 20h) |

### 38.4.6.23.2 Function

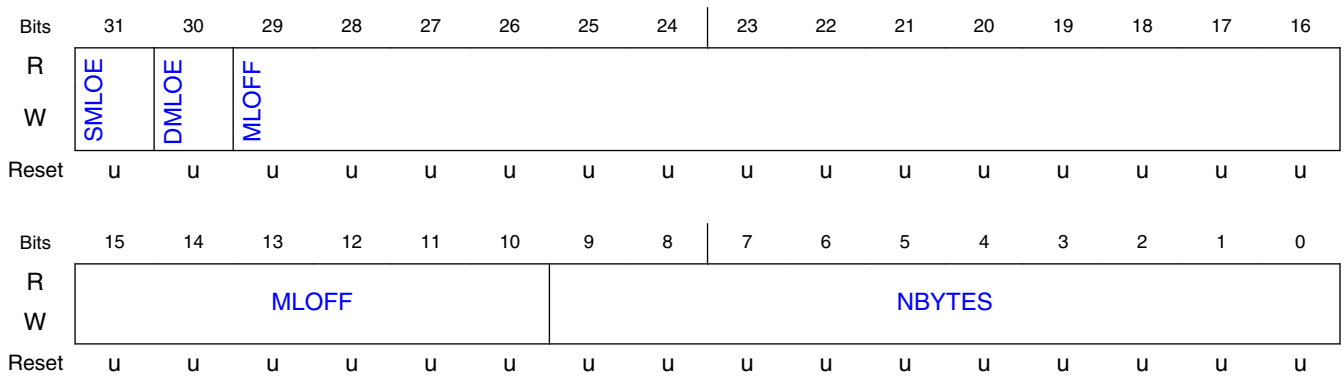
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

### 38.4.6.23.3 Diagram




### 38.4.6.23.4 Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br>0b - The minor loop offset is not applied to the SADDR<br>1b - The minor loop offset is applied to the SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - The minor loop offset is not applied to the DADDR<br>1b - The minor loop offset is applied to the DADDR |
| 29-10<br>MLOFF | If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes. |
| 9-0<br>NBYTES | Minor Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. |

## 38.4.6.24 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD7_NBYTES_MLNO)

### 38.4.6.24.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_NBYTES_MLNO | 1008h + (n × 20h) |

### 38.4.6.24.2 Function

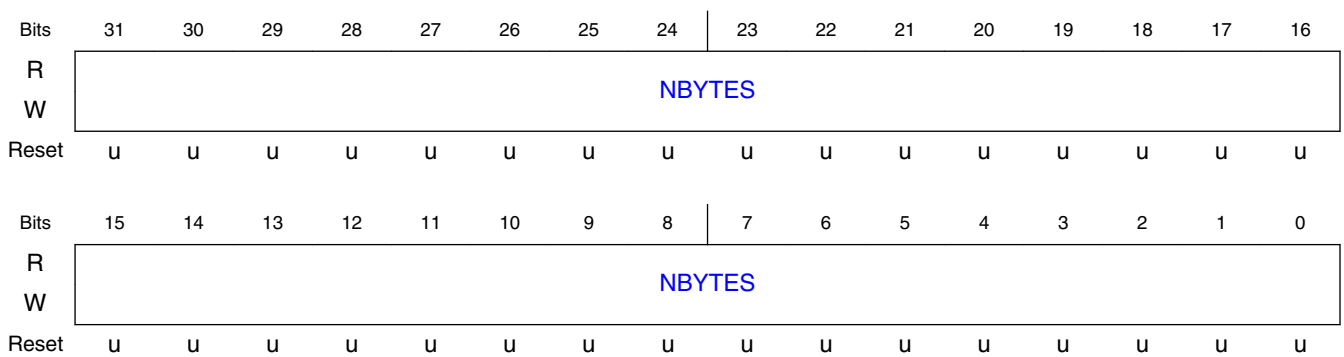
This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

### 38.4.6.24.3 Diagram




### 38.4.6.24.4 Fields

| Field | Function |
|---|---|
| 31-0<br>NBYTES | Minor Byte Transfer Count |

| Field | Function |
|---|---|
| | Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.<br>**NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer. |

## 38.4.6.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD7_SLAST)

### 38.4.6.25.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_SLAST | 100Ch + (n × 20h) |

### 38.4.6.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.6.25.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SLAST | Last Source Address Adjustment<br>Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.<br>This register uses two's complement notation; the overflow bit is discarded. |

## 38.4.6.26 TCD Destination Address (TCD0_DADDR - TCD7_DADDR)

### 38.4.6.26.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_DADDR | 1010h + (n × 20h) |

### 38.4.6.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.6.26.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DADDR | Destination Address<br>Memory address pointing to the destination data. |

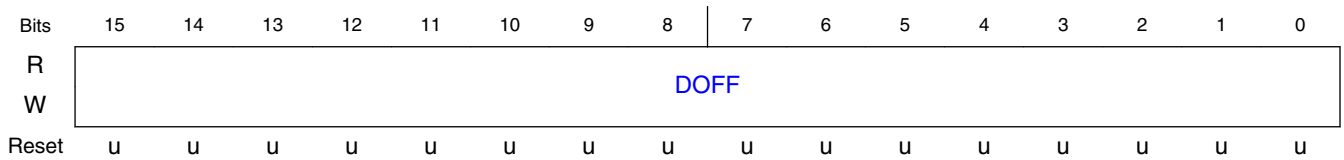
## 38.4.6.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD7_DOFF)

### 38.4.6.27.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_DOFF | 1014h + (n × 20h) |

### 38.4.6.27.2 Diagram




### 38.4.6.27.3 Fields

| Field | Function |
|---|---|
| 15-0<br>DOFF | Destination Address Signed Offset<br>Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed. |

## 38.4.6.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD7_CITER_ELINKNO)

### 38.4.6.28.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_CITER_ELINKNO | 1016h + (n × 20h) |

### 38.4.6.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD7_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is cleared, this register is defined as follows.

### 38.4.6.28.3 Diagram

### 38.4.6.28.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-0<br>CITER | Current Major Iteration Count<br>This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br>**NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.6.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD7_CITER_ELINKYES)

### 38.4.6.29.1 Offset

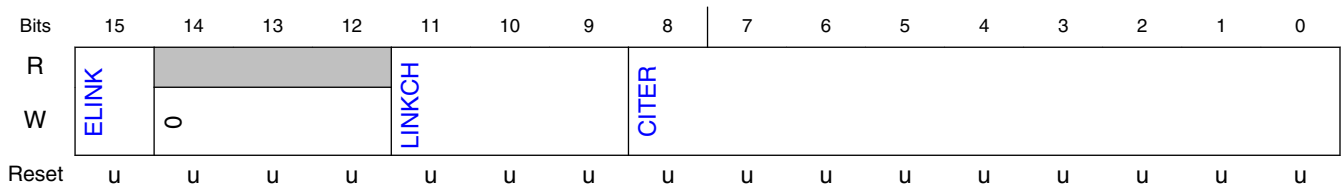
For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_CITER_ELINKYES | 1016h + (n × 20h) |

### 38.4.6.29.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD7_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is set, this register is defined as follows.

### 38.4.6.29.3 Diagram




### 38.4.6.29.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable channel-to-channel linking on minor-loop complete<br>As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-12<br>— | Reserved |
| 11-9<br>LINKCH | Minor Loop Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit. |
| 8-0<br>CITER | Current Major Iteration Count<br>This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. |

| Field | Function |
|---|---|
| | **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.6.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD7_DLASTSGA)

### 38.4.6.30.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_DLASTSGA | 1018h + (n × 20h) |

### 38.4.6.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLASTSGA | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLASTSGA | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### 38.4.6.30.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DLASTSGA | DLASTSGA<br>Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).<br>If (TCDn_CSR[ESG] = 0) then:<br>• Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.<br>• This field uses two's complement notation for the final destination address adjustment.<br>Otherwise: |

| Field | Function |
|---|---|
| | • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported. |

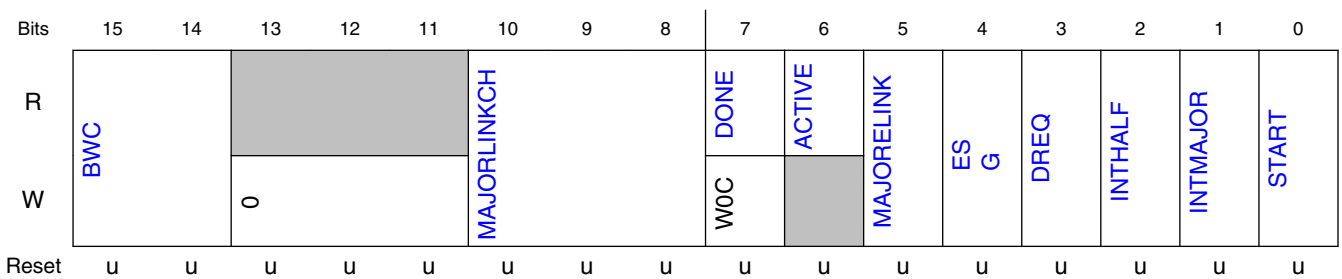
## 38.4.6.31 TCD Control and Status (TCD0_CSR - TCD7_CSR)

### 38.4.6.31.1 Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_CSR | 101Ch + (n × 20h) |

### 38.4.6.31.2 Diagram




### 38.4.6.31.3 Fields

| Field | Function |
|---|---|
| 15-14 | Bandwidth Control |
| BWC | Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch.<br>**NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.<br>00b - No eDMA engine stalls.<br>01b - Reserved<br>10b - eDMA engine stalls for 4 cycles after each R/W.<br>11b - eDMA engine stalls for 8 cycles after each R/W. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13-11<br>— | Reserved |
| 10-8<br>MAJORLINKCH | Major Loop Link Channel Number<br>If (MAJORELINK = 0) then:<br>• No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted.<br>Otherwise:<br>• After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. |
| 7<br>DONE | Channel Done<br>This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated.<br>**NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits. |
| 6<br>ACTIVE | Channel Active<br>This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected. |
| 5<br>MAJORELINK | Enable channel-to-channel linking on major loop complete<br>As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel.<br>**NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.<br>0b - The channel-to-channel linking is disabled.<br>1b - The channel-to-channel linking is enabled. |
| 4<br>ESG | Enable Scatter/Gather Processing<br>As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory.<br>**NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.<br>0b - The current channel's TCD is normal format.<br>1b - The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution. |
| 3<br>DREQ | Disable Request<br>If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero.<br>0b - The channel's ERQ bit is not affected.<br>1b - The channel's ERQ bit is cleared when the major loop is complete. |
| 2<br>INTHALF | Enable an interrupt when major counter is half complete.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress.<br>**NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead.<br>0b - The half-point interrupt is disabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - The half-point interrupt is enabled. |
| 1<br>INTMAJOR | Enable an interrupt when major iteration count completes.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero.<br>0b - The end-of-major loop interrupt is disabled.<br>1b - The end-of-major loop interrupt is enabled. |
| 0<br>START | Channel Start<br>If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.<br>0b - The channel is not explicitly started.<br>1b - The channel is explicitly started via a software initiated service request. |

## 38.4.6.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD7_BITER_ELINKYES)

### 38.4.6.32.1 Offset

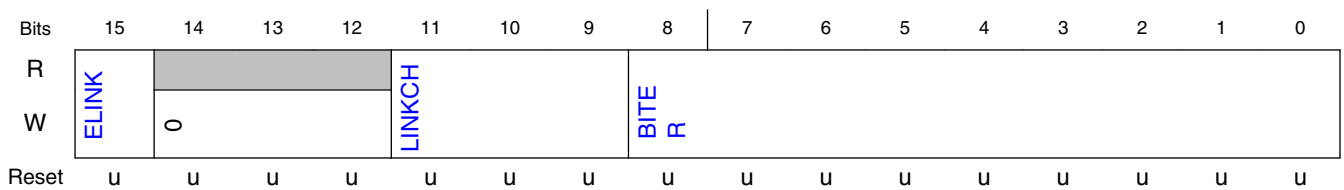
For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_BITER_ELINKYES | 101Eh + (n × 20h) |

### 38.4.6.32.2 Function

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

### 38.4.6.32.3 Diagram



### 38.4.6.32.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-12<br>— | Reserved |
| 11-9<br>LINKCH | Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. |
| 8-0<br>BITER | Starting major iteration count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.4.6.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD7 _BITER_ELINKNO)

### 38.4.6.33.1 Offset

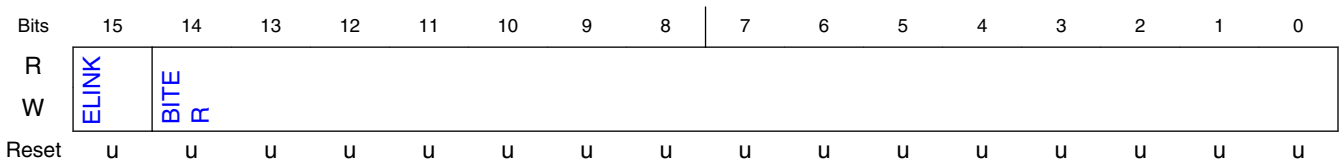
For n = 0 to 7:

| Register | Offset |
|---|---|
| TCDn_BITER_ELINKNO | 101Eh + (n × 20h) |

#### 38.4.6.33.2 Function

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

#### 38.4.6.33.3 Diagram




#### 38.4.6.33.4 Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enables channel-to-channel linking on minor loop complete<br>As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>0b - The channel-to-channel linking is disabled<br>1b - The channel-to-channel linking is enabled |
| 14-0<br>BITER | Starting Major Iteration Count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.<br>**NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

## 38.5 Functional description

The operation of the eDMA is described in the following subsections.

### 38.5.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

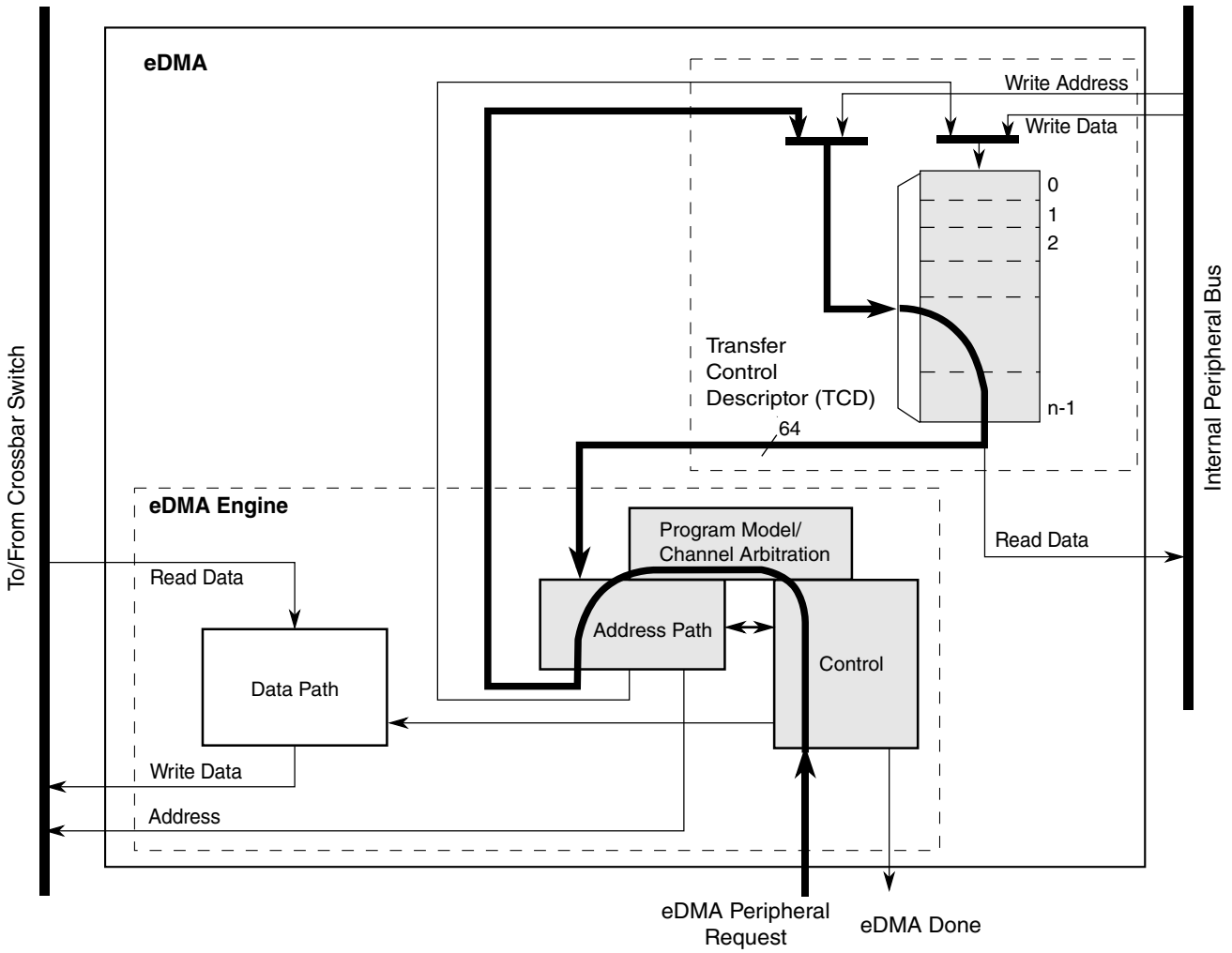



**Figure 38-2. eDMA operation, part 1**

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

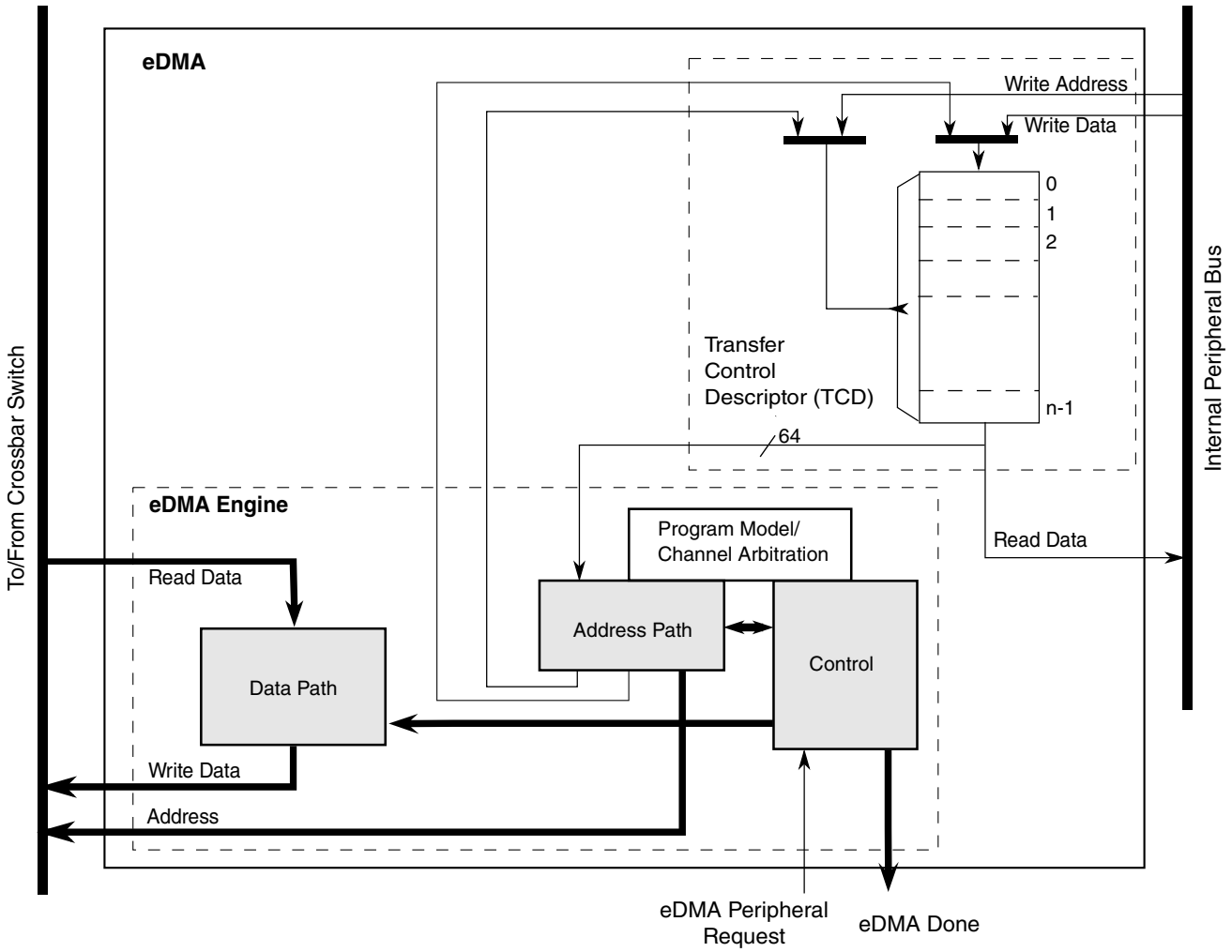



**Figure 38-3. eDMA operation, part 2**

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

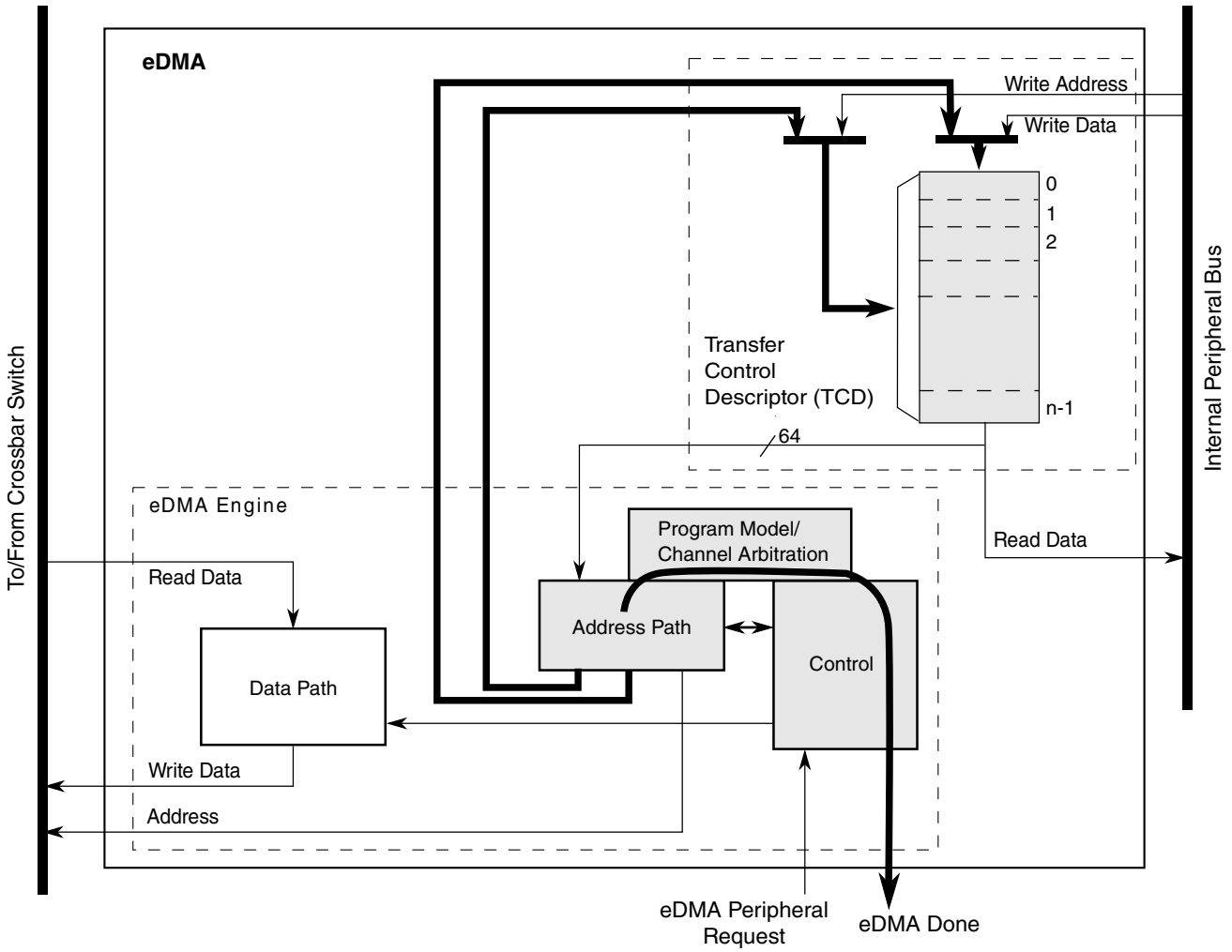



**Figure 38-4. eDMA operation, part 3**

## 38.5.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.

- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

**NOTE**

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration in undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error

occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

**NOTE**

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

### 38.5.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

### 38.5.4 Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

#### 38.5.4.1 Peak transfer rates

The peak transfer rates for several different source and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32-bits in size

**NOTE**

All architectures will not meet the assumptions listed above. See the SRAM configuration section for more information.

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

**Table 38-4. eDMA peak transfer rates (Mbytes/sec)**

| System Speed, Width | Internal SRAM-to-Internal SRAM | 32 bit internal peripheral bus-to-Internal SRAM | Internal SRAM-to-32 bit internal peripheral bus |
|---|---|---|---|
| 48.0 MHz, 32 bit | 96.0 | 48.0 | 38.4 |
| 66.7 MHz, 32 bit | 133.3 | 66.7 | 53.3 |
| 83.3 MHz, 32 bit | 166.7 | 83.3 | 66.7 |
| 100.0 MHz, 32 bit | 200.0 | 100.0 | 80.0 |
| 133.3 MHz, 32 bit | 266.7 | 133.3 | 106.7 |
| 150.0 MHz, 32 bit | 300.0 | 150.0 | 120.0 |

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

### 38.5.4.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

**Table 38-5. Hardware service request process**

| Cycle | | Description |
|---|---|---|
| With internal peripheral bus read and internal SRAM write | With SRAM read and internal peripheral bus write | |
| 1 | | eDMA peripheral request is asserted. |
| 2 | | The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD*n*_CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* word 7. |
| 3 | | Channel arbitration begins. |
| 4 | | Channel arbitration completes. The transfer control descriptor local memory read is initiated. |
| 5–6 | | The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles |
| 7 | | The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here. |
| 8–11 | 8–12 | The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write. |
| 12 | 13 | This cycle represents the data phase of the last destination write. |
| 13 | 14 | The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* word 7 is read and checked for channel linking or scatter/gather requests. |
| 14 | 15 | The appropriate fields in the first part of the TCD*n* are written back into the local memory. |
| 15 | 16 | The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start. |
| 16 | 17 | The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request. |

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can

be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle x +5. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

**Table 38-6. eDMA peak request rate (MReq/sec)**

| System frequency (MHz) | Request rate with zero wait states | Request rate with wait states |
|---|---|---|
| 48.0 | 5.3 | 4.2 |
| 66.6 | 7.4 | 5.8 |
| 83.3 | 9.2 | 7.2 |
| 100.0 | 11.1 | 8.7 |
| 133.3 | 14.8 | 11.6 |
| 150.0 | 16.6 | 13.0 |

A general formula to compute the peak request rate with overlapping requests is:

PEAKreq = freq / [ entry + (1 + read_ws) + (1 + write_ws) + exit ]

where:

**Table 38-7. Peak request formula operands**

| Operand | Description |
|---|---|
| PEAKreq | Peak request rate |
| freq | System frequency |
| entry | Channel startup (4 cycles) |
| read_ws | Wait states seen during the system bus read data phase |
| write_ws | Wait states seen during the system bus write data phase |
| exit | Channel shutdown (3 cycles) |

### 38.5.4.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer,

PEAKreq = 150 MHz / [ 4 + (1 + 1) + (1 + 3) + 3 ] cycles = 11.5 Mreq/sec

For an internal peripheral bus to SRAM transfer,

PEAKreq = 150 MHz / [ 4 + (1 + 2) + (1 + 1) + 3 ] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a TCD*n*_CSR[START] bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

**Note**

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

## 38.6 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

### 38.6.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.

3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

**Table 38-8. TCD Control and Status fields**

| TCD*n*_CSR field name | Description |
|---|---|
| START | Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware) |
| ACTIVE | Status bit indicating the channel is currently in execution |
| DONE | Status bit indicating major loop completion (cleared by software when using a software initiated DMA service) |
| D_REQ | Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service |
| BWC | Control bits for throttling bandwidth control of a channel |
| E_SG | Control bit to enable scatter-gather feature |
| INT_HALF | Control bit to enable interrupt when major loop is half complete |
| INT_MAJ | Control bit to enable interrupt when major loop completes |

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

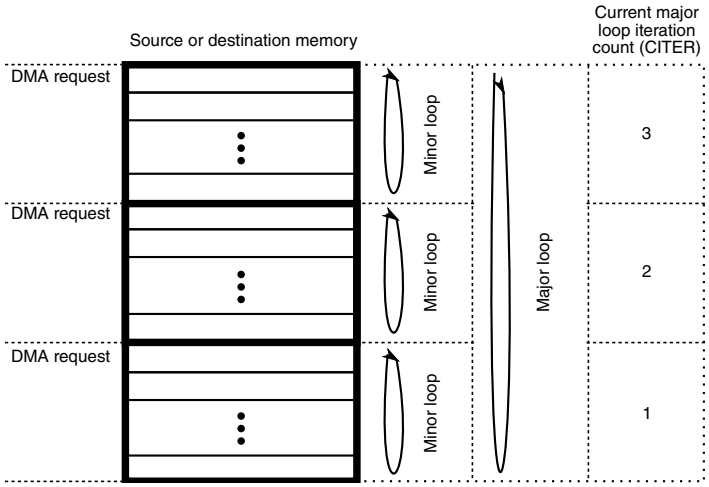



**Figure 38-5. Example of multiple loop iterations**

The following figure lists the memory array terms and how the TCD settings interrelate.

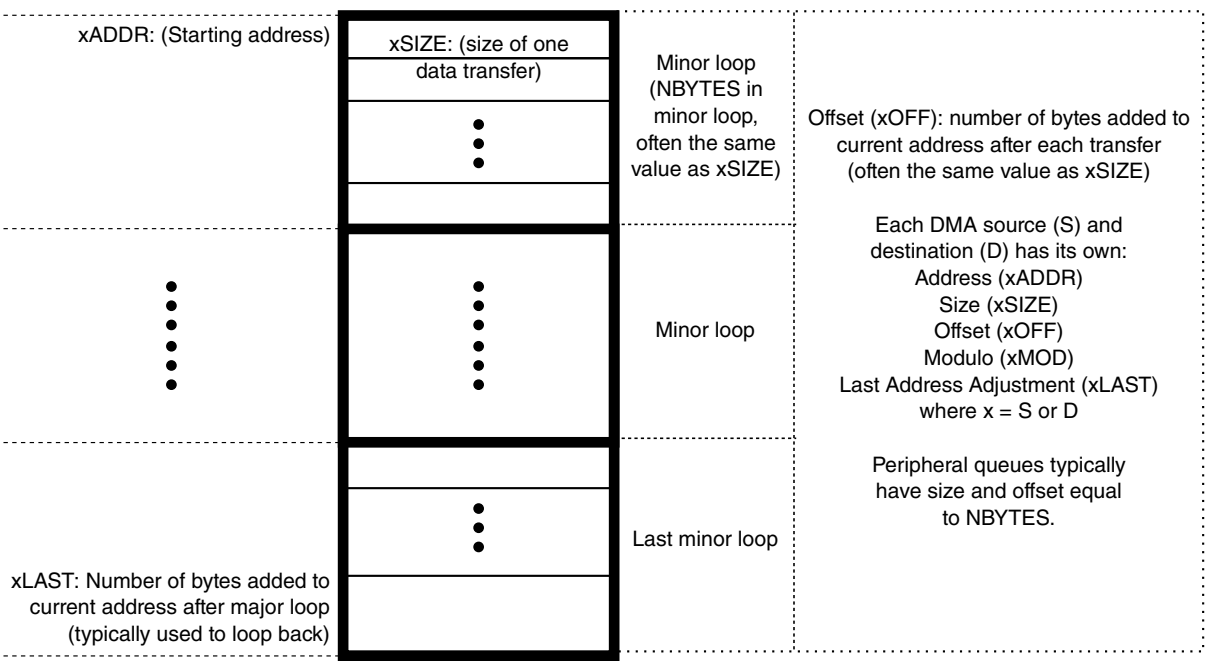



**Figure 38-6. Memory array terms**

## 38.6.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than channel priority error, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

## 38.6.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

### 38.6.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

### 38.6.3.2 Round-robin channel arbitration

Channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels.

## 38.6.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

### 38.6.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
    b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
    c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
    d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
    e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
    f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
    g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

### 38.6.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

    b. Write 32-bits to location 0x2010 → first iteration of the minor loop.

    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

    d. Write 32-bits to location 0x2014 → second iteration of the minor loop.

    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

    f. Write 32-bits to location 0x2018 → third iteration of the minor loop.

    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

    h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).

15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

#### 38.6.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a $2^4$ byte (16-byte) size queue.

**Table 38-9. Modulo example**

| Transfer Number | Address |
|---|---|
| 1 | 0x12345670 |
| 2 | 0x12345674 |
| 3 | 0x12345678 |
| 4 | 0x1234567C |
| 5 | 0x12345670 |
| 6 | 0x12345674 |

## 38.6.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

#### 38.6.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is

to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

| Stage | TCD*n*_CSR bits | | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 1 | 0 | 0 | Channel service request via software |
| 2 | 0 | 1 | 0 | Channel is executing |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle |

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

| Stage | TCD*n*_CSR bits | | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 0 | 0 | 0 | Channel service request via hardware (peripheral request asserted) |
| 2 | 0 | 1 | 0 | Channel is executing |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle |

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

### 38.6.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and

DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

#### 38.6.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected as the channel arbitration mode. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

### 38.6.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit

2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

**Note**

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

**Table 38-10. Channel Linking Parameters**

| Desired Link Behavior | TCD Control Field Name | Description |
|---|---|---|
| Link at end of Minor Loop | CITER[E_LINK] | Enable channel-to-channel linking on minor loop completion (current iteration) |
| | CITER[LINKCH] | Link channel number when linking at end of minor loop (current iteration) |
| Link at end of Major Loop | CSR[MAJOR_E_LINK] | Enable channel-to-channel linking on major loop completion |
| | CSR[MAJOR_LINKCH] | Link channel number when linking at end of major loop |

## 38.6.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

### 38.6.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

### 38.6.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCDn_CSR[MAJORELINK] bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCDn_CSR[MAJORELINK] bit at the same time the eDMA engine is retiring the channel. The TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCDn_CSR[MAJORELINK] bit.
2. Read back the TCDn_CSR[MAJORELINK] bit.
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCDn_CSR[MAJORELINK] bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

**NOTE**

The user must clear the The TCDn_CSR[DONE] bit before writing the TCDn_CSR[MAJORELINK] bit. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

### 38.6.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] bit at the same time the eDMA engine is retiring the channel. The ESG bit would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the MAJORLINKCH field and the ESG bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD MAJOR.E_LINK and E_SG bits to zero on any writes to a channel's TCD word 7 if that channel's TCD.DONE bit is set indicating the major loop is complete.

**NOTE**

The user must clear the TCDn_CSR[DONE] bit before writing the MAJORELINK or ESG bits. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

#### 38.6.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCDn_CSR[MAJORELINK] bit is zero, the TCDn_CSR[MAJORLINKCH] field is not used by the eDMA. In this case, the MAJORLINKCH field may be used for other purposes. This method uses the MAJORLINKCH field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCDn_CSR[MAJORLINKCH] field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCDn_CSR[DREQ] bit.

Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write 1b to the TCDn_CSR[ESG] bit.
5. Read back the 16 bit TCD control/status field.
6. Test the ESG request status and MAJORLINKCH value in the TCDn_CSR register:

   If ESG = 1b, the dynamic link attempt was successful.

   If ESG = 0b and the MAJORLINKCH (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0b and the MAJORLINKCH (ID) changed, the dynamic link attempt was successful (the new TCD's E_SG value cleared the ESG bit).

#### 38.6.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.DLAST_SGA field as a TCD identification (ID).

1. Write 1b to the TCDn_CSR[DREQ] bit.

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

2. Write the TCDn_DLASTSGA register with the scatter/gather address.
3. Write 1b to the TCDn_CSR[ESG] bit.
4. Read back the ESG bit.
5. Test the ESG request status:

   If ESG = 1b, the dynamic link attempt was successful.

   If ESG = 0b, read the 32 bit TCDn_DLASTSGA field.

   If ESG = 0b and the TCDn_DLASTSGA did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If ESG = 0b and the TCDn_DLASTSGA changed, the dynamic link attempt was successful (the new TCD's E_SG value cleared the ESG bit).

## 38.6.8 Suspend/resume a DMA channel with active hardware service requests

The DMA allows the user to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. Once the DMA and peripherals have been configured and are active, it is rare to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, a specific procedure must be followed. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (SPI), Sigma Delta Analog to Digital Convertor (SDADC), or other module.

### 38.6.8.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status Register (DMA_HRSn) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ bit on appropriate DMA channel.

### 38.6.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the its ERQ bit.
2. Enable the DMA service request at the peripheral.

For example, assume the SPI is set as a master for transmitting data via a DMA service request when the SPI_TXFIFO has an empty slot. The DMA will transfer the next command and data to the TXFIFO upon the request. If the user needs to suspend the DMA/SPI transfer loop, perform the following steps:

1. Disable the DMA service request at the source by writing 0 to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is 0.

2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is 0 for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ bit. If a service request is present, wait until the request has been processed and the HRS bit reads zero.

# Chapter 39
# Direct Memory Access Multiplexer (DMAMUX)

## 39.1 Chip-specific DMAMUX information

### 39.1.1 ARM Cortex-M0+ DMAMUX configuration

#### 39.1.1.1 DMAMUX request sources

This device includes a DMA request MUX that allows up to 32 DMA request signals to be mapped to any of the 8 DMA channels. Because of the mux there is not a correlation between any of the DMA request sources and a specific DMA channel.

Some of the modules support Asynchronous DMA operation as indicated by the last column in the following DMA source assignment table.

**Table 39-1. DMA request sources - DMA Channels 0-7**

| Source number | Source module | Source description | Asynch DMA capable |
|---|---|---|---|
| 0 | Low-Leakage Wake-up Unit 1 (LLWU1) | Wake-Up | Yes |
| 1 | Cryptographic Acceleration Unit version 3 (CAU3) | Data Request | No |
| 2 | Low-Power Timer 2 (LPTMR2) | Match | Yes |
| 3 | Timer / Pulse Width Modulation Module 3 (TPM3) | Channel 0 | Yes |
| 4 | Timer / Pulse Width Modulation Module 3 (TPM3) | Channel 1 | Yes |
| 5 | Timer / Pulse Width Modulation Module 3 (TPM3) | Overflow | Yes |
| 6 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) | Master / Slave Receive | Yes |
| 7 | Low-Power Inter-Integrated Circuit 3 (LPI2C3) | Master / Slave Transmit | Yes |
| 8 | 2.4GHz Radio 0 (RF0) | Receive | No |
| 9 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | Receive | Yes |
| 10 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | Transmit | Yes |

*Table continues on the next page...*

**Table 39-1. DMA request sources - DMA Channels 0-7 (continued)**

| Source number | Source module | Source description | Asynch DMA capable |
|---|---|---|---|
| 11 | Low-Power Universal Asynchronous Receive / Transmit 3 (LPUART3) | Receive | Yes |
| 12 | Low-Power Universal Asynchronous Receive / Transmit 3 (LPUART3) | Transmit | Yes |
| 13 | PORTE | Pin Request | Yes |
| 14 | Low-Power Comparator 1 (LPCMP1) | Comparator 1 Trigger | Yes |
| 15 | Low-Power Timer 1 (LPTMR1) | LPTMR1 Trigger | Yes |
| 16 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 0 | Yes |
| 17 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 1 | Yes |
| 18 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 2 | Yes |
| 19 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 3 | Yes |
| 20 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 4 | Yes |
| 21 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 5 | Yes |
| 22 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 6 | Yes |
| 23 | Flexible Inout / Ouput 0 (FlexIO0) | Shifter 7 | Yes |
| 24 | Serial Audio Interface 0 (SAI0) | Receive | Yes |
| 25 | Serial Audio Interface 0 (SAI0) | Transmit | Yes |
| 26 | PORTA | Pin Request | Yes |
| 27 | PORTB | Pin Request | Yes |
| 28 | PORTC | Pin Request | Yes |
| 29 | PORTD | Pin Request | Yes |
| 30 | Low-Power Analog-to-Digital Converter 0 (LPADC0) | Conversion Complete | Yes |
| 31 | Digital-to-Analog Converter 0 (DAC0) | Conversion Complete | Yes |

### 39.1.1.2 DMA transfers via TRGMUX1 DMAMUX1

The TRGMUX1_DMAMUX1 can trigger a DMA transfer on the first four DMA channels. The assignments are as follows.

**Table 39-2. TRGMUX1 DMAMUX1 channel assignments for periodic DMA triggering**

| DMA channel number | TRGMUX1_DMAMUX1 register fields |
|---|---|
| 0 | SEL0 |
| 1 | SEL1 |
| 2 | SEL2 |
| 3 | SEL3 |

## 39.1.2 ARM Cortex-M4 DMAMUX configuration

### 39.1.2.1 DMAMUX request sources

This device includes a DMA request MUX that allows up to 64 DMA request signals to be mapped to any of the 16 DMA channels. Because of the mux there is not a correlation between any of the DMA request sources and a specific DMA channel.

Some of the modules support Asynchronous DMA operation as indicated by the last column in the following DMA source assignment table.

**Table 39-3. DMA request sources - DMA Channels 0-15**

| Source number | Source module | Source description | Asynch DMA capable |
|---|---|---|---|
| 0 | Low-Leakage Wake-Up Unit 0 (LLWU0) | Wake-up Event | Yes |
| 1 | Low-Power Timer 0 (LPTMR0) | LPTMR0 Match | Yes |
| 2 | Low-Power Timer 1 (LPTMR1) | LPTMR1 Match | Yes |
| 3 | Timer/PWM Module 0 (TPM0) | Channel 0 | Yes |
| 4 | Timer/PWM Module 0 (TPM0) | Channel 1 | Yes |
| 5 | Timer/PWM Module 0 (TPM0) | Channel 2 | Yes |
| 6 | Timer/PWM Module 0 (TPM0) | Channel 3 | Yes |
| 7 | Timer/PWM Module 0 (TPM0) | Channel 4 | Yes |
| 8 | Timer/PWM Module 0 (TPM0) | Channel 5 | Yes |
| 9 | Timer/PWM Module 0 (TPM0) | Overflow | Yes |
| 10 | Timer/PWM Module 1 (TPM1) | Channel 0 | Yes |
| 11 | Timer/PWM Module 1 (TPM1) | Channel 1 | Yes |
| 12 | Timer/PWM Module 1 (TPM1) | Overflow | Yes |
| 13 | Timer/PWM Module 2 (TPM2) | Channel 0 | Yes |
| 14 | Timer/PWM Module 2 (TPM2) | Channel 1 | Yes |
| 15 | Timer/PWM Module 2 (TPM2) | Channel 2 | Yes |
| 16 | Timer/PWM Module 2 (TPM2) | Channel 3 | Yes |
| 17 | Timer/PWM Module 2 (TPM2) | Channel 4 | Yes |
| 18 | Timer/PWM Module 2 (TPM2) | Channel 5 | Yes |
| 19 | Timer/PWM Module 2 (TPM2) | Overflow | Yes |
| 20 | Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) | Receive | No |
| 21 | Euro Mastercard Visa Secure Identity Module 0 (EMVSIM0) | Transmit | No |
| 22 | Flexible Input/Output 0 (FlexIO0) | Shifter 0 | Yes |
| 23 | Flexible Input/Output 0 (FlexIO0) | Shifter 1 | Yes |

*Table continues on the next page...*

## Table 39-3. DMA request sources - DMA Channels 0-15 (continued)

| Source number | Source module | Source description | Asynch DMA capable |
|---|---|---|---|
| 24 | Flexible Input/Output 0 (FlexIO0) | Shifter 2 | Yes |
| 25 | Flexible Input/Output 0 (FlexIO0) | Shifter 3 | Yes |
| 26 | Flexible Input/Output 0 (FlexIO0) | Shifter 4 | Yes |
| 27 | Flexible Input/Output 0 (FlexIO0) | Shifter 5 | Yes |
| 28 | Flexible Input/Output 0 (FlexIO0) | Shifter 6 | Yes |
| 29 | Flexible Input/Output 0 (FlexIO0) | Shifter 7 | Yes |
| 30 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) | Master / Slave Receive | Yes |
| 31 | Low-Power Inter-Integrated Circuit 0 (LPI2C0) | Master / Slave Transmit | Yes |
| 32 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) | Master / Slave Receive | Yes |
| 33 | Low-Power Inter-Integrated Circuit 1 (LPI2C1) | Master / Salve Transmit | Yes |
| 34 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) | Master / Slave Receive | Yes |
| 35 | Low-Power Inter-Integrated Circuit 2 (LPI2C2) | Master / Slave Transmit | Yes |
| 36 | Serial Audio Interface 0 (SAI0) | Receive | Yes |
| 37 | Serial Audio Interface 0 (SAI0) | Transmit | Yes |
| 38 | Low-Power Serial Peripheral Interface 0 (LPSPI0) | Receive | Yes |
| 39 | Low-Power Serial Peripheral Interface 0 (LPSPI0) | Transmit | Yes |
| 40 | Low-Power Serial Peripheral Interface 1 (LPSPI1) | Receive | Yes |
| 41 | Low-Power Serial Peripheral Interface 1 (LPSPI1) | Transmit | Yes |
| 42 | Low-Power Serial Peripheral Interface 2 (LPSPI2) | Receive | Yes |
| 43 | Low-Power Serial Peripheral Interface 2 (LPSPI2) | Transmit | Yes |
| 44 | Low-Power Universal Asynchronous Receive/ Transmit 0 (LPUART0) | Receive | Yes |
| 45 | Low-Power Universal Asynchronous Receive/ Transmit 0 (LPUART0) | Transmit | Yes |
| 46 | Low-Power Universal Asynchronous Receive/ Transmit 1 (LPUART1) | Receive | Yes |
| 47 | Low-Power Universal Asynchronous Receive/ Transmit 1 (LPUART1) | Transmit | Yes |
| 48 | Low-Power Universal Asynchronous Receive/ Transmit 2 (LPUART2) | Receive | Yes |
| 49 | Low-Power Universal Asynchronous Receive/ Transmit 2 (LPUART2) | Transmit | Yes |
| 50 | PORTA | Pin Request | Yes |
| 51 | PORTB | Pin Request | Yes |
| 52 | PORTC | Pin Request | Yes |
| 53 | PORTD | Pin Request | Yes |

*Table continues on the next page...*

**Table 39-3. DMA request sources - DMA Channels 0-15 (continued)**

| Source number | Source module | Source description | Asynch DMA capable |
|---|---|---|---|
| 54 | Low-Power Analog-to-Digital Converter 0 (LPADC0) | Conversion Complete | Yes |
| 55 | Low-Power Comparator 0 (LPCMP0) | Comparator Trigger | Yes |
| 56 | Digital-to-Analog Converter 0 (DAC0) | Conversion Complete | Yes |
| 57 | Cryptographic Acceleration Unit version 3 (CAU3) | Data Request | No |
| 58 | Low-Power Timer 2 (LPTMR2) | LPTMR2 Trigger | Yes |
| 59 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | Receive | Yes |
| 60 | Low-Power Serial Peripheral Interface 3 (LPSPI3) | Transmit | Yes |
| 61 | Low-Power Universal Asynchronous Receive / Transmit 3 (LPUART3) | Receive | Yes |
| 62 | Low-Power Universal Asynchronous Receive / Transmit 3 (LPUART3) | Transmit | Yes |
| 63 | PORTE | Pin Request | Yes |

#### 39.1.2.2 DMA transfers via TRGMUX1 DMAMUX1

The TRGMUX0_DMAMUX0 can trigger a DMA transfer on the first four DMA channels. The assignments are as follows.

**Table 39-4. TRGMUX0 DMAMUX0 channel assignments for periodic DMA triggering**

| DMA channel number | TRGMUX0_DMAMUX0 register fields |
|---|---|
| 0 | SEL0 |
| 1 | SEL1 |
| 2 | SEL2 |
| 3 | SEL3 |

## 39.2 Introduction

### 39.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

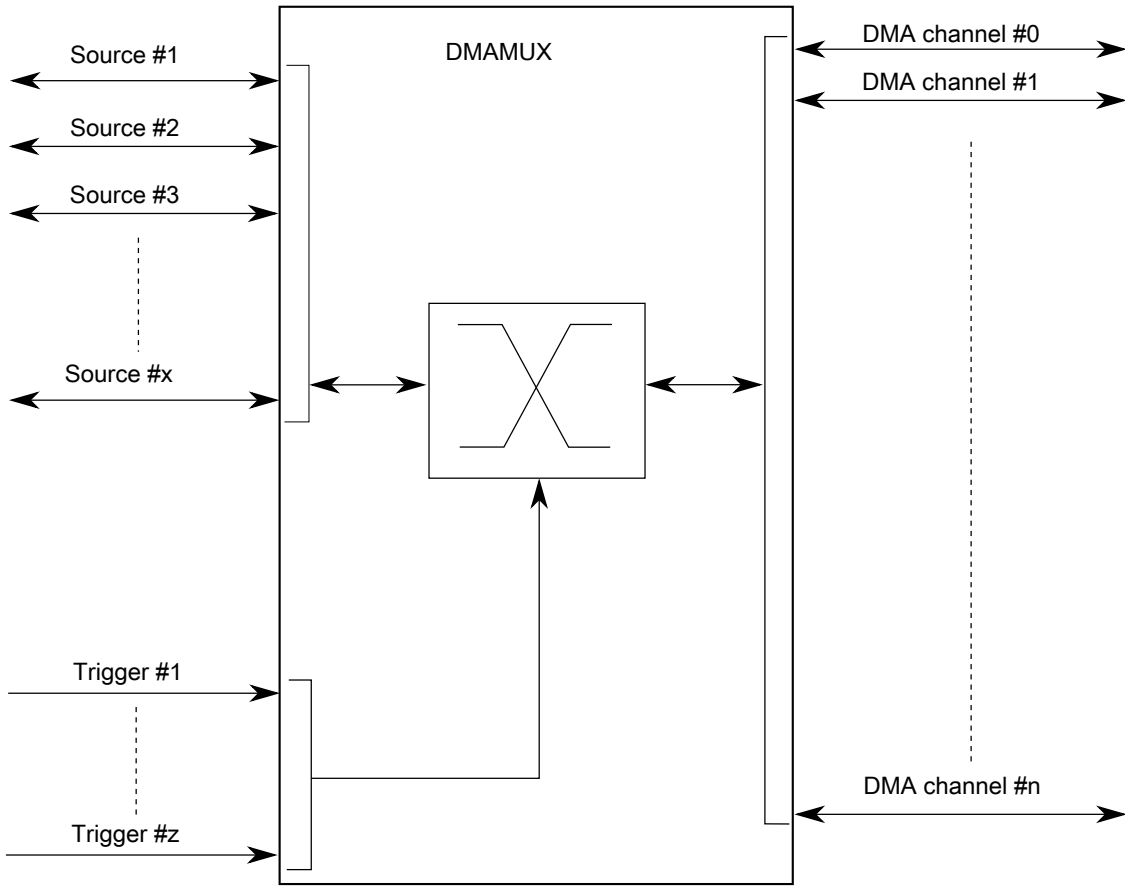



**Figure 39-1. DMAMUX block diagram**

## 39.2.2 Features

The DMAMUX module provides these features:

- Up to 64 peripheral slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - Each channel output can be individually configured to be Always On and not depend on any of the peripheral slots.
  - The first 4 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots.
- On every memory map configuration change for a any channel, this module signals to the DMA Controller to reset the internal state machine for that channel and it can accept a new request based on the new configuration.

### 39.2.3 Modes of operation

The following operating modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (LPIT). This mode is available only for channels 0 to 3.

## 39.3 External signal description

The DMAMUX has no external pins.

## 39.4 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

### 39.4.1 DMAMUX0 register descriptions

#### 39.4.1.1 DMAMUX0 Memory map

DMAMUX0 base address: 4002_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 3Ch | Channel a Configuration Register (CHCFG0 - CHCFG15) | 32 | RW | 0000_0000h |

## 39.4.1.2 Channel a Configuration Register (CHCFG0 - CHCFG15)

### 39.4.1.2.1 Offset

For a = 0 to 15:

| Register | Offset |
|---|---|
| CHCFGa | 0h + (a × 4h) |

### 39.4.1.2.2 Function

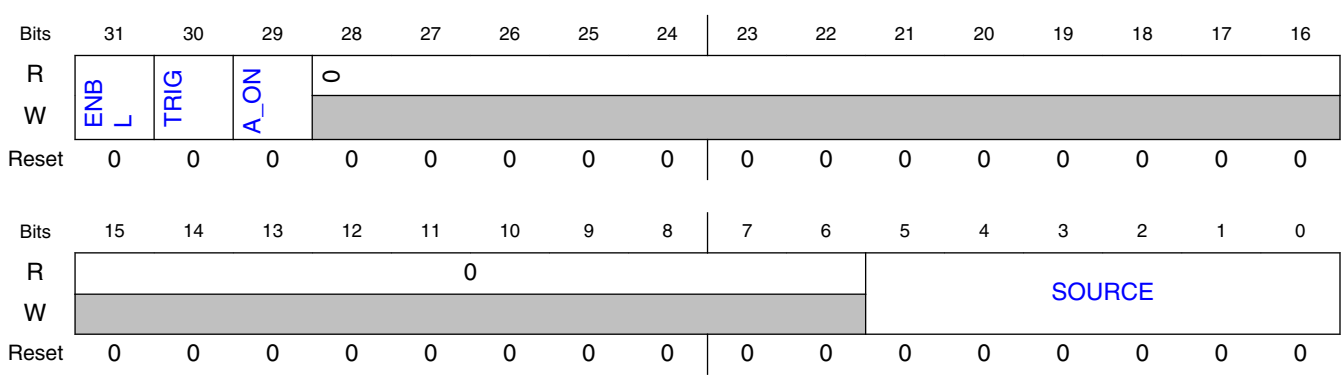
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

**NOTE**

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

### 39.4.1.2.3 Diagram



#### 39.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>ENBL | DMA Mux Channel Enable<br>Enables the channel for DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel.<br>0b - DMA Mux channel is disabled<br>1b - DMA Mux channel is enabled |
| 30<br>TRIG | DMA Channel Trigger Enable<br>Enables the periodic trigger capability for the triggered DMA channel.<br>0b - Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Triggering is enabled. If triggering is enabled and ENBL is set, the DMA_CH_MUX is in Periodic Trigger mode. |
| 29<br>A_ON | DMA Channel Always Enable<br>Enables the DMA Channel to be always ON. If TRIG bit is set, the module will assert request on every trigger.<br>0b - DMA Channel Always ON function is disabled<br>1b - DMA Channel Always ON function is enabled |
| 28-6<br>— | Reserved field |
| 5-0<br>SOURCE | DMA Channel Source (Slot Number)<br>Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMA_CH_MUX information for details about the peripherals and their slot numbers. |

## 39.4.2 DMAMUX1 register descriptions

### 39.4.2.1 DMAMUX1 Memory map

DMAMUX1 base address: 4102_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 1Ch | Channel a Configuration Register (CHCFG0 - CHCFG7) | 32 | RW | 0000_0000h |

### 39.4.2.2 Channel a Configuration Register (CHCFG0 - CHCFG7)

#### 39.4.2.2.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| CHCFGa | 0h + (a × 4h) |

#### 39.4.2.2.2 Function

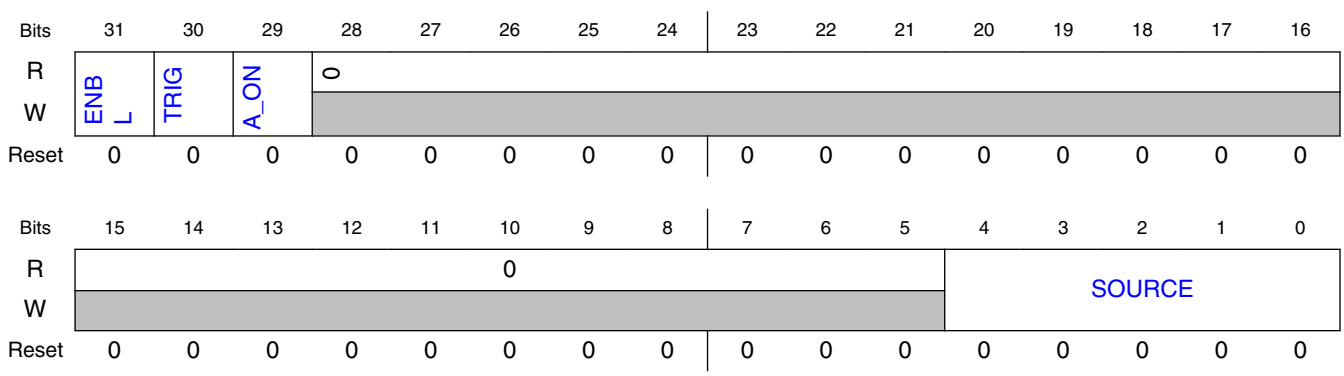
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

**NOTE**

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

#### 39.4.2.2.3 Diagram




#### 39.4.2.2.4 Fields

| Field | Function |
|---|---|
| 31<br>ENBL | DMA Mux Channel Enable<br>Enables the channel for DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel.<br>0b - DMA Mux channel is disabled<br>1b - DMA Mux channel is enabled |
| 30 | DMA Channel Trigger Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TRIG | Enables the periodic trigger capability for the triggered DMA channel.<br>0b - Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Triggering is enabled. If triggering is enabled and ENBL is set, the DMA_CH_MUX is in Periodic Trigger mode. |
| 29<br>A_ON | DMA Channel Always Enable<br>Enables the DMA Channel to be always ON. If TRIG bit is set, the module will assert request on every trigger.<br>0b - DMA Channel Always ON function is disabled<br>1b - DMA Channel Always ON function is enabled |
| 28-5<br>— | Reserved field |
| 4-0<br>SOURCE | DMA Channel Source (Slot Number)<br>Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMA_CH_MUX information for details about the peripherals and their slot numbers. |

## 39.5 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

### 39.5.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (LPIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the LPIT. See the section on periodic interrupt timer for more information on this topic.

### Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

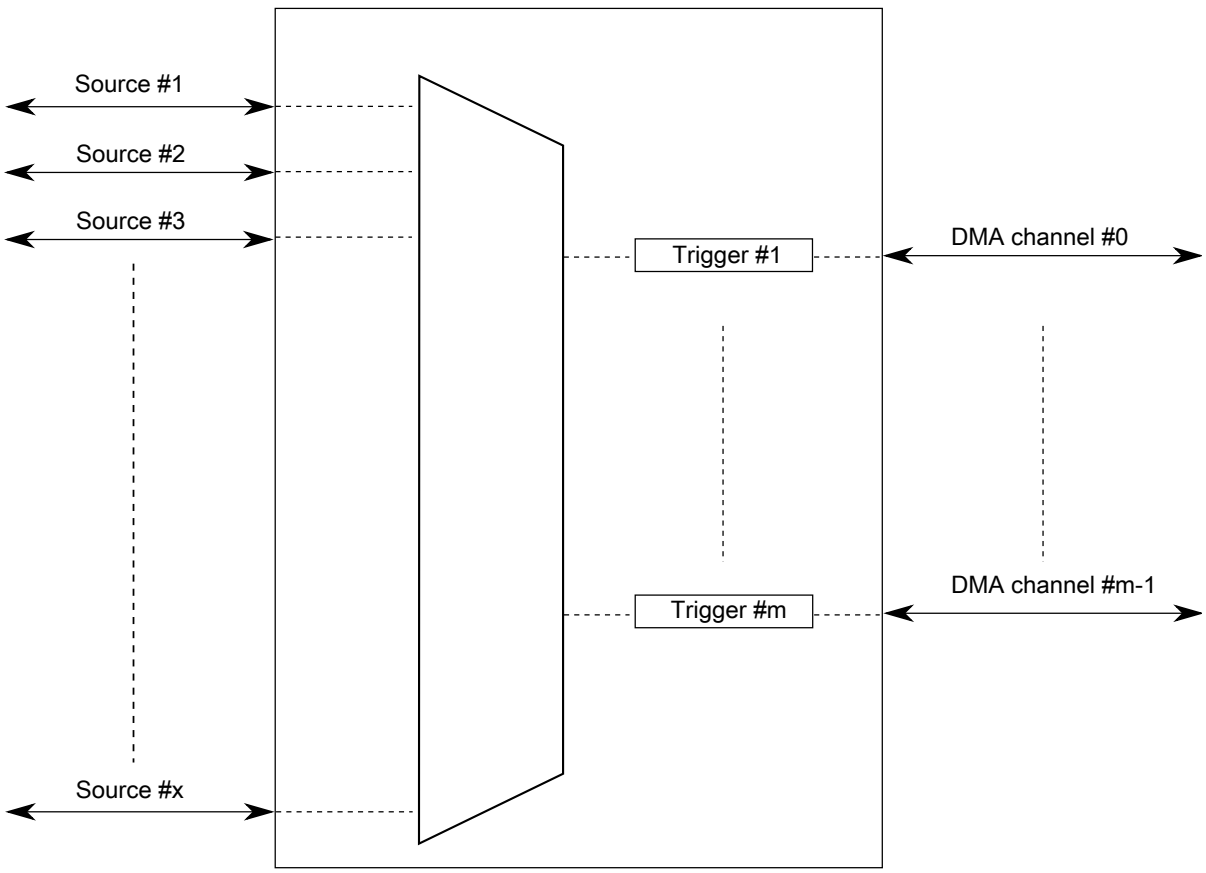



**Figure 39-2. DMAMUX triggered channels**

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

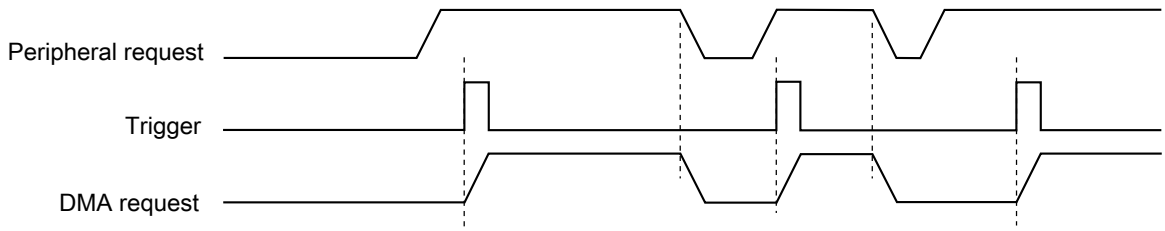



**Figure 39-3. DMAMUX channel triggering: normal operation**

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

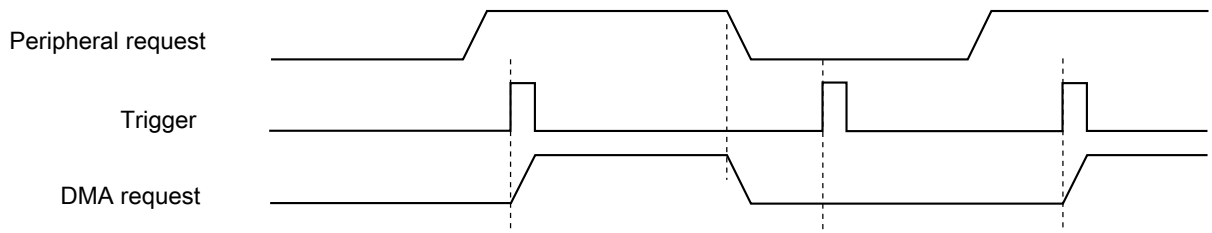



**Figure 39-4. DMAMUX channel triggering: ignored trigger**

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

### 39.5.2 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, each DMA Channel can be individually configured to function as an Always Enabled source. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the always enabled channel provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA channel can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and configured as "always enabled" channel. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

**NOTE**

When a channel is configured as "Always Enabled", then the peripheral DMA sources for that channel are ignored; i.e. SOURCE field has no effect.

## 39.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

### 39.6.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

### 39.6.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.

5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

**NOTE**

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00000000 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC0000005 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is setwhile CHCFG[TRIG] is cleared.

**NOTE**

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1 with no periodic triggering capability :

1. Write 0x00000000 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x80000005 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes long is 32-bits */
volatile unsigned long *CHCFG0 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned long *CHCFG1 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned long *CHCFG2 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned long *CHCFG3 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned long *CHCFG4 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0010);
volatile unsigned long *CHCFG5 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0014);
volatile unsigned long *CHCFG6 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0018);
volatile unsigned long *CHCFG7 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x001C);
volatile unsigned long *CHCFG8 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0020);
volatile unsigned long *CHCFG9 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0024);
volatile unsigned long *CHCFG10= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0028);
volatile unsigned long *CHCFG11= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x002C);
volatile unsigned long *CHCFG12= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0030);
volatile unsigned long *CHCFG13= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0034);
volatile unsigned long *CHCFG14= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0038);
volatile unsigned long *CHCFG15= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x003C);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00000000;
*CHCFG1 = 0x80000005;
```

### To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

### To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00000000 to CHCFG8.
3. Write 0x80000007 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

**Initialization/application information**

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes long is 32-bits */
volatile unsigned long *CHCFG0 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned long *CHCFG1 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned long *CHCFG2 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned long *CHCFG3 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned long *CHCFG4 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0010);
volatile unsigned long *CHCFG5 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0014);
volatile unsigned long *CHCFG6 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0018);
volatile unsigned long *CHCFG7 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x001C);
volatile unsigned long *CHCFG8 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0020);
volatile unsigned long *CHCFG9 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0024);
volatile unsigned long *CHCFG10= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0028);
volatile unsigned long *CHCFG11= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x002C);
volatile unsigned long *CHCFG12= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0030);
volatile unsigned long *CHCFG13= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0034);
volatile unsigned long *CHCFG14= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0038);
volatile unsigned long *CHCFG15= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x003C);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00000000;
*CHCFG8 = 0x80000007;
```

#### 39.6.2.1 Configuration options

**Table 39-5. Channel Configuration Options**

| ENBL | TRIG | A_ON | Function | Mode |
|---|---|---|---|---|
| 0 | X | X | DMA channel is disabled | Disabled Mode |
| 1 | 0 | 0 | DMA channel is enabled with no triggering (transparent) | Normal Mode |
| 1 | 1 | 0 | DMA channel is enabled with triggering | Periodic Trigger Mode |
| 1 | 0 | 1 | DMA channel is always enabled | Always On Mode |
| 1 | 1 | 1 | DMA channel is always enabled with triggering | Always On Trigger Mode |

# Chapter 40
# Messaging Unit (MU)

## 40.1 Overview

The Messaging Unit module enables two processors within the SoC to communicate and coordinate by passing messages (e.g. data, status and control) through the MU interface. The MU also provides the ability for one processor to signal the other processor using interrupts.

The MU module contains two sets of registers (Processor A-facing, Processor B-facing). They work in synchronized clock domains.

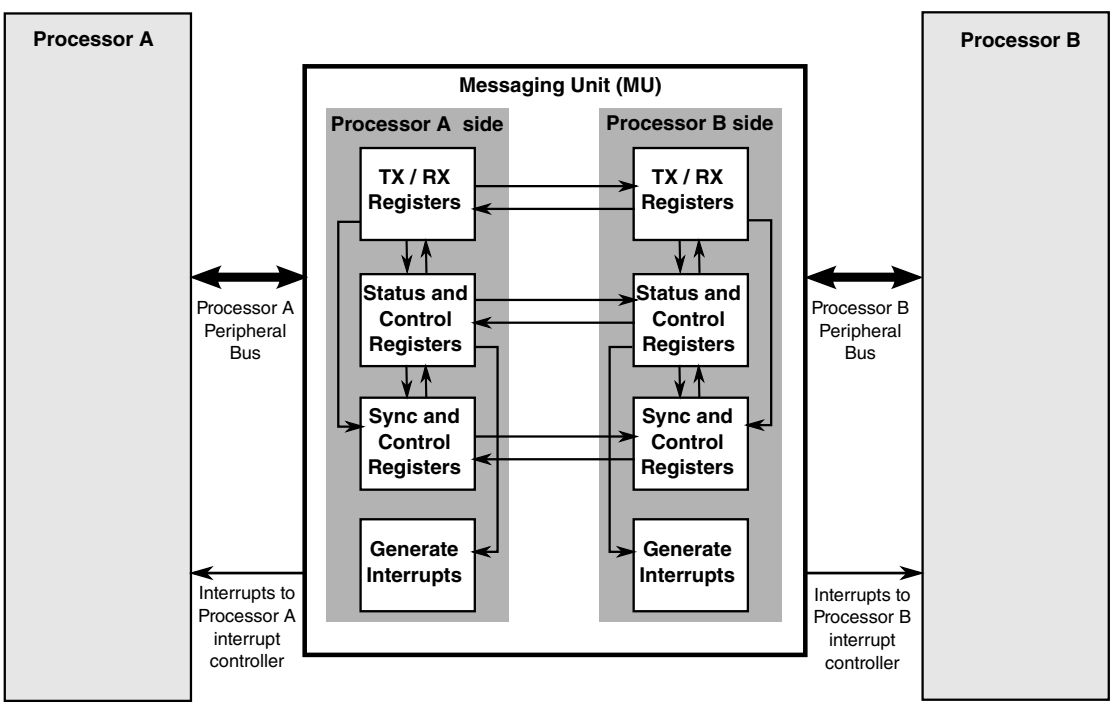



**Figure 40-1. MU Block Diagram**

### 40.1.1 Features

The MU includes the following features:

- Messaging control by interrupts or by polling
- Symmetrical processor interfaces with each side supporting the following:
  - Three general-purpose flags reflected to the other side
  - Four general-purpose interrupt requests reflected to the other side
  - Four receive registers with maskable interrupt
  - Four transmit registers with maskable interrupt
- The Processor B can take the Processor A out of low-power modes by asserting one of the interrupts to the Processor A and vice versa

**NOTE**

MU chapter references to Processor A may also correspond to Core 0 and references to Processor B may also correspond to Core 1 in other parts of this document.

### 40.1.2 Modes of Operation

The MU supports the modes described in the indicated sections:

- Low Power Modes

## 40.2 Register Definition

The MU provides transmit and receive data registers for the communication between Processor A and Processor B. Some control and status registers to the Processor A and Processor B sides for control operations (such as interrupts and reset), and for status checking of the other MU-side. The following diagram shows the MU registers schematic.

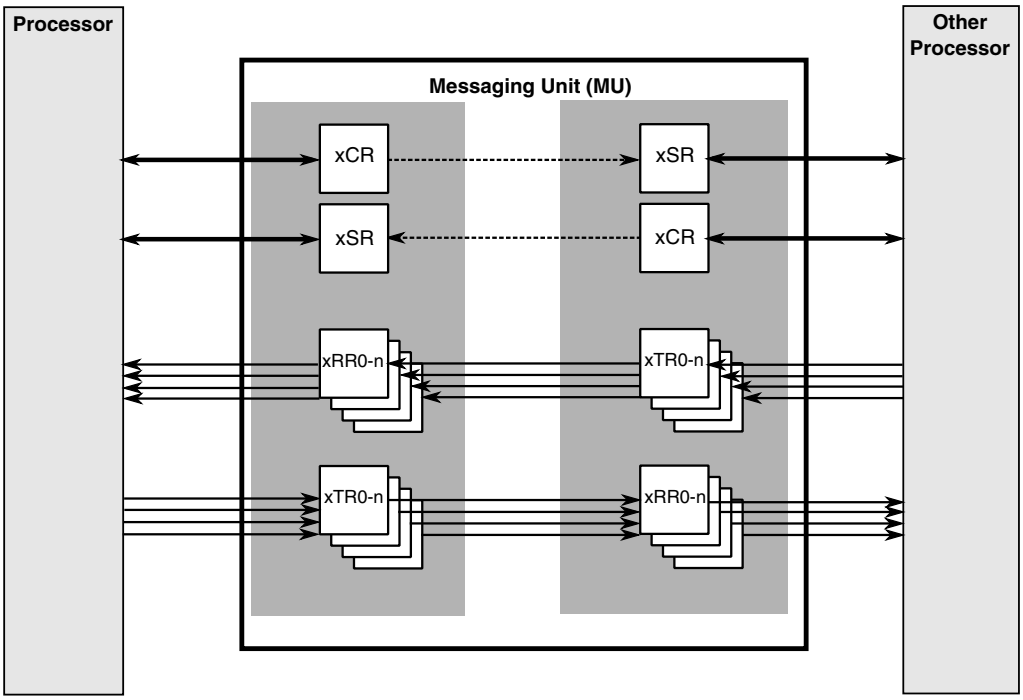



**Figure 40-2. MU Registers**

The detailed MU Register Definition can be found below.

**NOTE**

Writing to an RRn register can cause a transfer error.

## 40.2.1 MUA register descriptions

This section contains the detailed register descriptions for the MUA registers.

### 40.2.1.1 MU Memory map

MUA base address: 4002_5000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VER) | 32 | RO | 0101_0006h |
| 4h | Parameter Register (PAR) | 32 | RO | 0000_0000h |
| 20h - 2Ch | Transmit Register (TR0 - TR3) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 40h - 4Ch | Receive Register (RR0 - RR3) | 32 | RO | 0000_0000h |
| 60h | Status Register (SR) | 32 | W1C | 00F0_0000h |
| 64h | Control Register (CR) | 32 | RW | 0000_0000h |
| 68h | Core Control Register (CCR) | 32 | RW | See description. |

## 40.2.1.2 Version ID Register (VER)

### 40.2.1.2.1 Offset

| Register | Offset |
|---|---|
| VER | 0h |

### 40.2.1.2.2 Function

Use Version ID register to determine the version ID and feature set number of MUA.

### 40.2.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

### 40.2.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | This read only field returns the major version number for MUA. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for MUA. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number for MUA.<br>000000000000x1xxb - Core Control and Status Registers are implemented in both MUA and MUB.<br>000000000000xx1xb - RAIP/RAIE register bits are implemented.<br>000000000000xxx0b - Standard features implemented |

## 40.2.1.3 Parameter Register (PAR)

### 40.2.1.3.1 Offset

| Register | Offset |
|---|---|
| PAR | 4h |

### 40.2.1.3.2 Function

Use Parameter register to determine the parameter settings of MUA.

### 40.2.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PARAMETER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PARAMETER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 40.2.1.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PARAMETER | This bitfield contains the parameter settings of MUA. |

## 40.2.1.4 Transmit Register (TR0 - TR3)

### 40.2.1.4.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TRn | 20h + (n × 4h) |

### 40.2.1.4.2 Function

Use MUA Transmit Register (TRn, 32-bit, write-only) to transmit a message or data to MUB.

- You can only write to the TRn register when the TEn bit in SR register is set to "1".
- Reading the TRn register returns all zeros.

### 40.2.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 40.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | DATA<br>MUA Transmit Register Data. (Write-only)<br>• Data written to the TRn register is reflected in the MUB Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register.<br>• A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUA Status Register (SR) on the transmitter side, and sets a "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUA side).<br>• Any write to the TRn register will update all status information. |

## 40.2.1.5 Receive Register (RR0 - RR3)

### 40.2.1.5.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| RRn | 40h + (n × 4h) |

### 40.2.1.5.2 Function

Use MUA Receive Register (RRn, 32-bit, read-only) to receive a message or data from the MUB.

- Data written to the MUB TRn register is immediately reflected in the MUA RRn register.
- You can only read the RRn register when the RFn bit in the SR register is set to "1".
- Writing to the RRn register generates an error response to the MUA.

### 40.2.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 40.2.1.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | DATA<br>MUA Receive Register. (Read-only)<br>• Reflects the data written to MUB Transmit Register 0 (TRn).<br>• Reading the RRn register clears the "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUB Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUB side).<br>• Any read of the RRn register will update all status information. |

## 40.2.1.6 Status Register (SR)

### 40.2.1.6.1 Offset

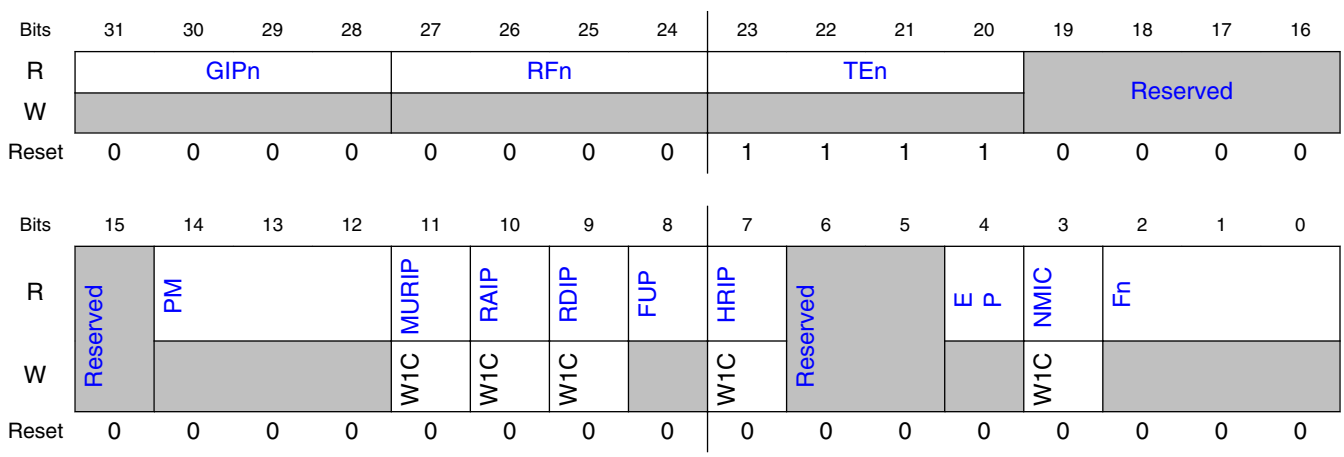
| Register | Offset |
|---|---|
| SR | 60h |

### 40.2.1.6.2 Function

Use the Processor A Status Register (SR, 32-bit, read-write) to show interrupt status from the Processor B, general purpose flags, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor A-side programmer
- Other dual-purpose bits are set by the Processor A-side programmer, and cleared by the MU logic.

### 40.2.1.6.3 Diagram




### 40.2.1.6.4 Fields

| Field | Function |
|---|---|
| 31-28<br>GIPn | GIPn<br>For n = {0, 1, 2, 3} MUA General Interrupt Request n Pending. (Read-Write)<br>• GIPn bit signals the MUA that the GIRn bit in the CR register on the MUB side was set from "0" to "1". If the GIEn bit in the MUA CR register is set to "1", a General Interrupt n request is issued. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • The GIPn bit is cleared by writing it back as "1". Writing "0", or writing "1" when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUA register, set the desired bit, and write it to the MUA SR register, thus clearing the GIPn bit.<br>• GIPn bit is cleared when the MU is reset.<br><br>0000b - MUA general purpose interrupt n is not pending. (default)<br>0001b - MUA general purpose interrupt n is pending. |
| 27-24<br>RFn | RFn<br>For n = {0, 1, 2, 3} MUA Receive Register n Full. (Read-only)<br>• The RFn bit is set to "1" when the MUB TRn register is written on the MUB side.<br>• After the RFn bit is set to "1", the RFn bit signals the MUA side that new data is ready to be read by the MUA in the MUA RRn register, and a Receive n interrupt is issued on the MUA side (if the RIEn bit in the MUA CR register has been set to "1").<br>• RFn bit is cleared when the MUA RRn register is read, and when the MU is reset.<br><br>0000b - MUA RRn register is not full (default).<br>0001b - MUA RRn register has received data from MUB TRn register and is ready to be read by the MUA. |
| 23-20<br>TEn | TEn<br>For n = {0, 1, 2, 3} MUA Transmit Register n Empty. (Read-only)<br>• The TEn bit is set to "1" after the MUB RRn register is read on the MUB side.<br>• After the TEn bit is set to "1", the TEn bit signals the MUA side that the MUA TRn register is ready to be written on the MUA side, and a Transmit n interrupt is issued on the MUA side (if the TEn bit in the MUA CR register is set to "1").<br>• TEn bit is cleared after the MUA TRn register is written on the MUA side.<br>• TEn bit is set to "1" when the MU is reset.<br><br>0000b - MUA TRn register is not empty.<br>0001b - MUA TRn register is empty (default). |
| 19-15<br>— | Reserved |
| 14-12<br>PM | PM<br>Processor B-side Power Mode. (Read-only)<br>• PM[2:0] bits indicate the Processor B-side power mode.<br><br>**NOTE:** The Processor B Power Mode is platform-specific.<br><br>000b - The MUB processor is in Run Mode.<br>001b - The MUB processor is in COO Mode.<br>010b - The MUB processor is in WAIT Mode.<br>011b - The MUB processor is in STOP/VLPS Mode.<br>100b - The MUB processor is in LLS/VLLS Mode. |
| 11<br>MURIP | MURIP<br>MU Reset Interrupt Pending. (Read-Write)<br>• MURIP bit signals Processor A that Processor B initiated a MU reset by setting MUR bit in MUB_CR register.<br>• MURIP bit is set to "1" after Processor B initiated a MU reset. The interrupt generated by a Processor B MU reset is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B set MUR bit in MUB_CR, if the interrupt is enabled by the MURIE bit.<br>• To clear the MURIP bit, write "1", which also clears general purpose interrupt 3.<br>• Only system reset can reset MURIP. MU reset cannot reset this bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Processor B did not issue MU reset<br>1b - Processor B issued MU reset |
| 10<br>RAIP | RAIP<br>Processor B Reset Asserted Interrupt Pending. (Read-Write)<br>• RAIP bit signals the Processor A that the Processor B has entered reset.<br>• RAIP bit is set to "1" after the Processor B enters reset. The interrupt generated by a Processor B reset assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B enters reset, if the interrupt is enabled by the RAIE bit.<br>• To clear the RAIP bit, write "1", which also clears general purpose interrupt 3.<br>• When Processor A comes out of reset, the RAIP bit has value "0" (default). Processor A will then monitor the reset of Processor B and will assert RAIP if its reset asserts.<br><br>0b - Processor B did not enter reset<br>1b - Processor B entered reset |
| 9<br>RDIP | RDIP<br>Processor B Reset De-asserted Interrupt Pending. (Read-Write)<br>• RDIP bit signals the Processor A that the Processor B has come out of reset.<br>• RDIP bit is set to "1" after the Processor B comes out of reset. The interrupt generated by a Processor B reset de-assertion is ORed with the Processor A general purpose interrupt 3. The Processor A general purpose interrupt 3 is issued when the Processor B comes out of reset if the interrupt is enabled by the RDIE bit.<br>• To clear the RDIP bit, write "1", which also clears general purpose interrupt 3.<br>• When Processor A comes out of reset, the RDIP bit has value "0"(default). Processor A then monitors the reset of Processor B and will assert RDIP when reset deasserts.<br><br>0b - Processor B did not exit reset<br>1b - Processor B exited from reset |
| 8<br>FUP | FUP<br>MUA Flags Update Pending. (Read-only)<br>• FUP bit is set to "1" when the MUA side sends a Flags Update request to the MUB side.<br>• A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUA CR register. No flag update changes are allowed while the FUP bit is set to "1". Any write to the Fn[2:0] bits of the MUA CR register, while the FUP bit is set to "1", will not generate a Flags Update event, and the Fn[2:0] bits will stay unchanged.<br>• FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU MUB side, and during MU reset.<br><br>0b - No flags updated, initiated by the MUA, in progress (default)<br>1b - MUA initiated flags update, processing |
| 7<br>HRIP | HRIP<br>Processor A Hardware reset interrupt pending<br>• HRIP bit indicates if Processor A had been reset by HR bit in MUB_CCR<br>• HRIP is set to "1" if Processor B set HR in MUB_CCR to "1" and HRM in MUA_CCR is cleared.<br>• Write "1" to clear HRIP.<br><br>0b - MUB didn't issue hardware reset to Processor A<br>1b - MUB had initiated a hardware reset to Processor A through HR bit. |
| 6-5<br>— | Reserved |
| 4<br>EP | EP<br>MUA Side Event Pending. (Read-only) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • EP bit is set to "1" when the MUA side mechanism sends an event update request to the MUB side.<br>• EP bit is cleared when the event update acknowledge is received. An "event" is any hardware message that is reflected in the MUB SR register on the MUB side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically.<br>• To ensure events have been posted to MUB before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to "1", you should wait and continue to poll it (EP bit) before entering STOP mode.<br>• Reading the MUA SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions.<br>• The EP bit is cleared when the MU resets.<br><br>0b - The MUA side event is not pending (default).<br>1b - The MUA side event is pending. |
| 3<br>NMIC | NMIC<br>Processor A Non-Maskable-Interrupt Clear. (Write-only)<br>• NMIC bit is used by the Processor A-side MU Non-Maskable Interrupt service routine to clear the non-maskable interrupt.<br>• Writing NMIC bit as "1" signals the MU to clear the NMI bit in the MUB CR register, thus de-asserting the interrupts and enabling the NMI bit to receive another interrupt.<br>• The NMIC bit is always read as "0"; therefore it (NMIC bit) cannot be polled. The NMIC bit can only be used as part of the interrupt service routine, in which the NMIC bit should only be written as "1" once.<br>• The NMIC bit is cleared when the MU resets.<br><br>0b - Default<br>1b - Writing "1" clears the NMI bit in the MUB CR register. |
| 2-0<br>Fn | Fn<br>For n = {0, 1, 2} MUA Side Flag n. (Read-only)<br>• Fn bit is the MUA side flag that reflects the values written to the Fn bit in the MUB control register.<br>• Every time that the Fn bit in the MUB CR register is written, the Fn bit in the MUB CR register write event updates the Fn bit in the MUA SR register after the event update latency, which is measured in terms of the number of clocks of the MUB and the MUA.<br><br>000b - Fn bit in the MUB CR register is written 0 (default).<br>001b - Fn bit in the MUB CR register is written 1. |

## 40.2.1.7 Control Register (CR)

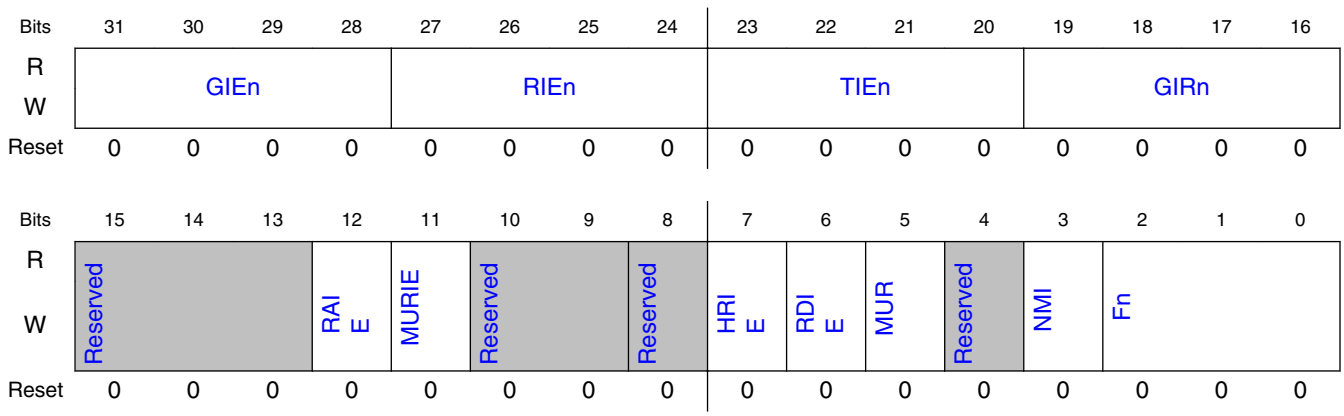
### 40.2.1.7.1 Offset

| Register | Offset |
|---|---|
| CR | 64h |

### 40.2.1.7.2 Function

Use the Control Register (CR, 32-bit, read-write) to enable the MU interrupts on the MUA-side, and trigger events and interrupts on the MUB-side (general purpose interrupt, flag update).

### 40.2.1.7.3 Diagram




### 40.2.1.7.4 Fields

| Field | Function |
|---|---|
| 31-28<br>GIEn | GIEn<br>For n = {0, 1, 2, 3} MUA General Purpose Interrupt Enable n. (Read-Write)<br>• GIEn bit enables MUA General Interrupt n.<br>• If GIEn bit is set to "1" (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUA SR register is set to "1".<br>• If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued.<br>• GIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUA General Interrupt n. (default)<br>0001b - Enables MUA General Interrupt n. |
| 27-24<br>RIEn | RIEn<br>For n = {0, 1, 2, 3} MUA Receive Interrupt Enable n. (Read-Write)<br>• RIEn bit enables MUA Receive Interrupt n.<br>• If RIEn bit is set to "1" (enabled), then an MUA Receive Interrupt n request is issued when the RFn bit in the MUA SR register is set to "1".<br>• If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUA Receive Interrupt n request will be issued.<br>• RIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUA Receive Interrupt n. (default)<br>0001b - Enables MUA Receive Interrupt n. |
| 23-20<br>TIEn | TIEn<br>For n = {0, 1, 2, 3} MUA Transmit Interrupt Enable n. (Read-Write) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • TIEn bit enables MUA Transmit Interrupt n.<br>• If TIEn bit is set to "1" (enabled), then an MUA Transmit Interrupt n request is issued when the TEn bit in the MUA SR register is set to "1".<br>• If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUA Transmit Interrupt n request will be issued.<br>• TIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUA Transmit Interrupt n. (default)<br>0001b - Enables MUA Transmit Interrupt n. |
| 19-16<br>GIRn | GIRn<br>For n = {0, 1, 2, 3} MUA General Purpose Interrupt Request n. (Read-Write)<br>• Writing "1" to the GIRn bit sets the GIPn bit in the MUB SR register on the Processor B-side. If the GIEn bit in the MUB CR register is set to "1" on the MUB side, a General Purpose Interrupt n request is triggered.<br>• The GIRn bit is cleared if the GIPn bit (in the MUB SR register on the MUB side) is cleared by writing it (GIPn bit) as "1", thereby signalling the MUA that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as "0" on the MUA side.<br>• To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit).<br>• GIRn bit is cleared when the MU resets.<br><br>0000b - MUA General Interrupt n is not requested to the MUB (default).<br>0001b - MUA General Interrupt n is requested to the MUB. |
| 15-13<br>— | Reserved |
| 12<br>RAIE | RAIE<br>Processor B Reset Assertion Interrupt Enable. (Read-Write)<br>• RAIE bit enables Processor A General Interrupt 3.<br>• If RAIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the RAIP bit in the MUA SR register is set to "1".<br>• If RAIE is cleared, then the value of the RAIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The RAIE bit is cleared when the MU resets.<br><br>0b - Disables Processor A-side General Purpose Interrupt 3 request due to Processor B reset assertion.<br>1b - Enables Processor A-side General Purpose Interrupt 3 request due to Processor B reset assertion. |
| 11<br>MURIE | MURIE<br>MU reset Interrupt Enable. (Read-Write)<br>• MURIE bit enables Processor A General Interrupt 3.<br>• If MURIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the MURIP bit in the MUA SR register is set to "1".<br>• If MURIE is cleared, then the value of the MURIP bit is ignored and no General Interrupt 3 request will be issued.<br>• Only system reset can reset MURIE, MU reset cannot reset this bit.<br><br>0b - Disables Processor A-side General Purpose Interrupt 3 request due to MU reset issued by MUB.<br>1b - Enables Processor A-side General Purpose Interrupt 3 request due to MU reset issued by MUB. |
| 10-9<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>— | Reserved |
| 7<br>HRIE | Processor A hardware reset interrupt enable<br>Processor A Hardware Reset Interrupt Enable. (Read-Write)<br>• HRIE bit enables Processor A General Interrupt 3.<br>• If HRIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the HRIP bit in the MUA SR register is set to "1".<br>• If HRIE is cleared, then the value of the HRIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The HRIE bit is cleared when the MU resets.<br><br>0b - Disables Processor A General Purpose Interrupt 3 request due to Processor B issued HR to Processor A.<br>1b - Enables Processor A General Purpose Interrupt 3 request due to Processor B issued HR to Processor A. |
| 6<br>RDIE | RDIE<br>Processor B Reset De-assertion Interrupt Enable. (Read-Write)<br>• RDIE bit enables Processor A General Interrupt 3.<br>• If RDIE bit is set to "1", then General Interrupt 3 request is issued to the Processor A when the RDIP bit in the MUA SR register is set to "1".<br>• If RDIE is cleared, then the value of the RDIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The RDIE bit is cleared when the MU resets.<br><br>0b - Disables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion.<br>1b - Enables Processor A General Purpose Interrupt 3 request due to Processor B reset de-assertion. |
| 5<br>MUR | MUR<br>MU Reset.<br>• Setting MUR bit to "1" resets both the Processor A and the Processor B sides of the MU module, forcing all control and status registers to return to their default values and all internal states to be cleared.<br>• The BOOT and RSTH fields will not be affected by the MUR bit.<br>• Before setting the MUR bit to "1", it is advisable to interrupt the Processor B , because setting the MUR bit may affect the ongoing Processor B program.<br>• MUR bit can only be written as "1".<br>• MUR bit is always read as "0".<br>• MUR bit is cleared during the MU reset sequence.<br><br>0b - N/A. Self clearing bit (default).<br>1b - Asserts the MU reset. |
| 4<br>— | Reserved |
| 3<br>NMI | NMI<br>MUB Non-maskable Interrupt. (Read-Write)<br>• When NMI bit is set to "1", it initiates a Non-Maskable Interrupt to the Processor B.<br>• NMI bit is cleared by the MU after the MUB asserts the NMIC bit in the MUB SR register.<br>• After the NMI bit is cleared, the MUA can initiate another non-maskable interrupt to the MUB.<br>• The NMI bit is cleared when the MU resets.<br><br>0b - Non-maskable interrupt is not issued to the Processor B by the Processor A (default). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Non-maskable interrupt is issued to the Processor B by the Processor A. |
| 2-0<br>Fn | Fn<br>For n = {0, 1, 2} MUA to MUB Flag n. (Read-Write)<br>• Fn bit is a read-write flag that is reflected in Fn bit in the MUB SR register on the MUB side.<br>• Fn bit is cleared when the MU resets.<br><br>000b - Clears the Fn bit in the SR register.<br>001b - Sets the Fn bit in the SR register. |

## 40.2.1.8 Core Control Register (CCR)

### 40.2.1.8.1 Offset

| Register | Offset |
|---|---|
| CCR | 68h |

### 40.2.1.8.2 Function

Use the Core Control Register (CCR, 32-bit, read-write) to enable the MUA to control the Processor on MUB-side.

### 40.2.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | BOOT | | CLKE | RST H | HRM | HR |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | 0 | 0 |

1. Reset value loaded during Processor A reset from Flash IFR.

## 40.2.1.8.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-4<br>BOOT | Slave Processor B Boot Config.<br>Configures the boot source for Processor B when Processor B is slave core.<br>00b - Boot from Dflash base<br>01b - Reserved<br>10b - Boot from CM0+ RAM base<br>11b - Reserved |
| 3<br>CLKE | MUB clock enable<br>When set, forces MUB platform clock to remain enabled even after MUB-side has entered a stop mode. When clear, allows MUB to gate its platform clock whenever desired after entering a stop mode. Note that if MUA also enters a stop mode, then this bit is overidden to allow MUB to gate its platform clock.<br>0b - MUB platform clock gated when MUB-side enters a stop mode.<br>1b - MUB platform clock kept running after MUB-side enters a stop mode, until MUA also enters a stop mode. |
| 2<br>RSTH | Processor B Reset Hold<br>When set, will cause the Processor B to be held in reset following any reset event. When cleared, allows the Processor B to be released from reset.<br>Setting this bit will not force the Processor B into reset. It will only hold Processor B if it resets itself.<br>0b - Release Processor B from reset<br>1b - Hold Processor B in reset |
| 1<br>HRM | When set, HR bit in MUB CCR has no effect<br>Processor A Hardware Reset Mask. (Read-Write)<br>• The Processor B can give a hardware reset to the Processor A by setting the HR bit in the MUB CCR Register to “1”.<br>• When the HRM bit is set to “1” by the Processor A, the HR reset issued by the Processor B is masked (disabled by the Processor A).<br>• When the HRM bit is cleared, the HR reset issued by the Processor B to the Processor A is not masked (enabled by the Processor A).<br>0b - HR bit in MUB CCR is not masked, enables the hardware reset to the Processor A (default after hardware reset).<br>1b - HR bit in MUB CCR is masked, disables the hardware reset request to the Processor A. |
| 0<br>HR | HR<br>Processor B Hardware Reset. (Read-Write)<br>• HR bit asserts and de-asserts the hardware reset of the Processor B.<br>• Set HR bit to "1" to start a hardware reset of the Processor B.<br>• Clear the HR bit to de-assert the Processor B hardware reset input.<br>• Assert the HR bit for a minimum of 3 clock cycles. The RAIP bit in MUA SR register indicates the state of the Processor on MUB-side. As soon as the Processor B goes into Reset (MUA RAIP bit is set to "1"), the HR bit can be de-asserted.<br>• Strobe-setting the HR bit will not cause an internal MU reset but will be routed outside MU to Processor B reset logic<br>• The HR reset issued by the Processor A to the Processor B is maskable by the Processor B (according to the settings of the HRM bit in the MUB CCR register). If the HRM bit (in the MUB CCR register) is set to "1", then the HR reset is masked; if the HRM bit (in the MUB CCR register) is cleared (default), then the HR reset is enabled.<br>• The HR bit does not return to the reset value during the software (MUR) reset. |

| Field | Function |
|---|---|
| | 0b - De-assert Hardware reset to the Processor B. (default)<br>1b - Assert Hardware reset to the Processor B. |

## 40.2.2 MUB register descriptions

This section contains the detailed register descriptions for the MUB registers.

### 40.2.2.1 MU Memory map

MUB base address: 4102_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VER) | 32 | RO | 0101_0006h |
| 4h | Parameter Register (PAR) | 32 | RO | 0000_0000h |
| 20h - 2Ch | Transmit Register (TR0 - TR3) | 32 | RW | 0000_0000h |
| 40h - 4Ch | Receive Register (RR0 - RR3) | 32 | RO | 0000_0000h |
| 60h | Status Register (SR) | 32 | W1C | 00F0_0000h |
| 64h | Control Register (CR) | 32 | RW | 0000_0000h |
| 68h | Core Control Register (CCR) | 32 | RW | See description. |

### 40.2.2.2 Version ID Register (VER)

#### 40.2.2.2.1 Offset

| Register | Offset |
|---|---|
| VER | 0h |

#### 40.2.2.2.2 Function

Use Version ID register to determine the version ID and feature set number of MUB.

#### 40.2.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

#### 40.2.2.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for MUB. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for MUB. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number for MUB.<br>000000000000x1xxb - Core Control and Status Registers are implemented in both MUA and MUB.<br>000000000000xx1xb - RAIP/RAIE register bits are implemented.<br>000000000000xxx0b - Standard features implemented |

### 40.2.2.3 Parameter Register (PAR)

#### 40.2.2.3.1 Offset

| Register | Offset |
|---|---|
| PAR | 4h |

#### 40.2.2.3.2 Function

Use Parameter register to determine the parameter settings of MUB.

#### 40.2.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PARAMETER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PARAMETER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 40.2.2.3.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PARAMETER | This bitfield contains the parameter settings of MUB. |

### 40.2.2.4 Transmit Register (TR0 - TR3)

#### 40.2.2.4.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TRn | 20h + (n × 4h) |

#### 40.2.2.4.2 Function

Use MUB Transmit Register (TRn, 32-bit, write-only) to transmit a message or data to MUA.

- You can only write to the TRn register when the TEn bit in SR register is set to "1".
- Reading the TRn register returns all zeros.

### 40.2.2.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 40.2.2.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | DATA<br>MUB Transmit Register Data. (Write-only)<br>• Data written to the TRn register is reflected in the MUA Receive Register n (RRn). The TRn and RRn registers are not double-buffered, a write to the TRn register overrides the data readable at the RRn register.<br>• A write to the transmit register clears a "transmitter empty" bit (TEn) in the MUB Status Register (SR) on the transmitter side, and sets a "receiver full" bit (RFn) in the MUA Status Register (SR) on the receiver side (optionally triggering an interrupt 0 on the MUB side).<br>• Any write to the TRn register will update all status information. |

## 40.2.2.5 Receive Register (RR0 - RR3)

### 40.2.2.5.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| RRn | 40h + (n × 4h) |

### 40.2.2.5.2 Function

Use MUB Receive Register (RRn, 32-bit, read-only) to receive a message or data from the MUA.

- Data written to the MUA TRn register is immediately reflected in the MUB RRn register.

- You can only read the RRn register when the RFn bit in the SR register is set to "1".
- Writing to the RRn register generates an error response to the MUB.

#### 40.2.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 40.2.2.5.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | DATA<br>MUB Receive Register. (Read-only)<br>• Reflects the data written to MUA Transmit Register 0 (TRn).<br>• Reading the RRn register clears the "receiver full" bit (RFn) in the MUB Status Register (SR) on the receiver side, and sets the "transmitter empty" bit (TEn) in the MUA Status Register on the transmitter side (optionally triggering a transmit interrupt 0 on the MUA side).<br>• Any read of the RRn register will update all status information. |

### 40.2.2.6 Status Register (SR)

#### 40.2.2.6.1 Offset

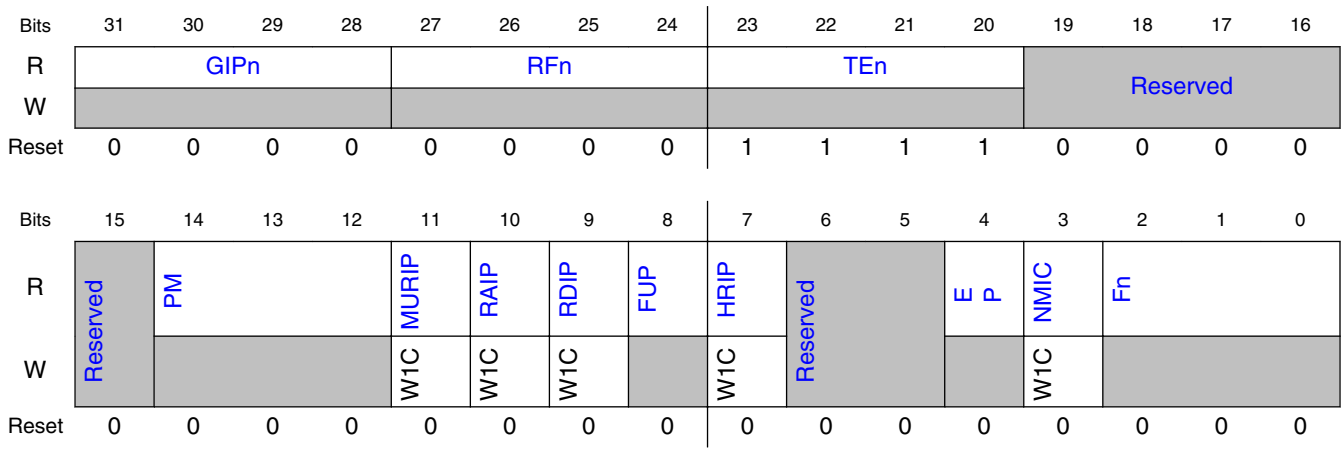
| Register | Offset |
|---|---|
| SR | 60h |

#### 40.2.2.6.2 Function

Use the Processor B Status Register (SR, 32-bit, read-write) to show interrupt status from the Processor A, general purpose flags, and to set dual function control-status bits.

- Some dual-purpose bits are set by the MU logic, and cleared by the Processor B-side programmer
- Other dual-purpose bits are set by the Processor B-side programmer, and cleared by the MU logic.

### 40.2.2.6.3 Diagram




### 40.2.2.6.4 Fields

| Field | Function |
|---|---|
| 31-28<br>GIPn | GIPn<br>For n = {0, 1, 2, 3} MUB General Interrupt Request n Pending. (Read-Write)<br>• GIPn bit signals the MUB that the GIRn bit in the CR register on the MUA side was set from "0" to "1". If the GIEn bit in the MUB CR register is set to "1", a General Interrupt n request is issued.<br>• The GIPn bit is cleared by writing it back as "1". Writing "0", or writing "1" when the GIPn bit is cleared is ignored. Use this feature in the interrupt routine, where the GIPn bit is cleared in order to de-assert the interrupt request source at the interrupt controller. An example of a proper bit clearing sequence is: clear MUB register, set the desired bit, and write it to the MUB SR register, thus clearing the GIPn bit.<br>• GIPn bit is cleared when the MU is reset.<br><br>0000b - MUB general purpose interrupt n is not pending. (default)<br>0001b - MUB general purpose interrupt n is pending. |
| 27-24<br>RFn | RFn<br>For n = {0, 1, 2, 3} MUB Receive Register n Full. (Read-only)<br>• The RFn bit is set to "1" when the MUA TRn register is written on the MUA side.<br>• After the RFn bit is set to "1", the RFn bit signals the MUB side that new data is ready to be read by the MUB in the MUB RRn register, and a Receive n interrupt is issued on the MUB side (if the RIEn bit in the MUB CR register has been set to "1").<br>• RFn bit is cleared when the MUB RRn register is read, and when the MU is reset.<br><br>0000b - MUB RRn register is not full (default).<br>0001b - MUB RRn register has received data from MUA TRn register and is ready to be read by the MUB. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-20<br>TEn | TEn<br>For n = {0, 1, 2, 3} MUB Transmit Register n Empty. (Read-only)<br>• The TEn bit is set to "1" after the MUA RRn register is read on the MUA side.<br>• After the TEn bit is set to "1", the TEn bit signals the MUB side that the MUB TRn register is ready to be written on the MUB side, and a Transmit n interrupt is issued on the MUB side (if the TEn bit in the MUB CR register is set to "1").<br>• TEn bit is cleared after the MUB TRn register is written on the MUB side.<br>• TEn bit is set to "1" when the MU is reset.<br><br>0000b - MUB TRn register is not empty.<br>0001b - MUB TRn register is empty (default). |
| 19-15<br>— | Reserved |
| 14-12<br>PM | PM<br>Processor A-side Power Mode. (Read-only)<br>• PM[2:0] bits indicate the Processor A-side power mode.<br><br>**NOTE:** The Processor A Power Mode is platform-specific.<br><br>000b - The MUA processor is in Run Mode.<br>001b - The MUA processor is in COO Mode.<br>010b - The MUA processor is in WAIT Mode.<br>011b - The MUA processor is in STOP/VLPS Mode.<br>100b - The MUA processor is in LLS/VLLS Mode. |
| 11<br>MURIP | MURIP<br>MU Reset Interrupt Pending. (Read-Write)<br>• MURIP bit signals Processor B that Processor A initiated a MU reset by setting MUR bit in MUA_CR register.<br>• MURIP bit is set to "1" after Processor A initiated a MU reset. The interrupt generated by a Processor A MU reset is ORed with the Processor B general purpose interrupt 3. The Processor B general purpose interrupt 3 is issued when the Processor A set MUR bit in MUA_CR, if the interrupt is enabled by the MURIE bit.<br>• To clear the MURIP bit, write "1", which also clears general purpose interrupt 3.<br>• Only system reset can reset MURIP. MU reset cannot reset this bit.<br><br>0b - Processor A did not issue MU reset<br>1b - Processor A issued MU reset |
| 10<br>RAIP | RAIP<br>Processor A Reset Asserted Interrupt Pending. (Read-Write)<br>• RAIP bit signals the Processor B that the Processor A has entered reset.<br>• RAIP bit is set to "1" after the Processor A enters reset. The interrupt generated by a Processor A reset assertion is ORed with the Processor B general purpose interrupt 3. The Processor B general purpose interrupt 3 is issued when the Processor A enters reset, if the interrupt is enabled by the RAIE bit.<br>• To clear the RAIP bit, write "1", which also clears general purpose interrupt 3.<br>• When Processor B comes out of reset, the RAIP bit has value "0" (default). Processor B will then monitor the reset of Processor A and will assert RAIP if its reset asserts.<br><br>0b - Processor A did not enter reset<br>1b - Processor A entered reset |
| 9<br>RDIP | RDIP<br>Processor A Reset De-asserted Interrupt Pending. (Read-Write)<br>• RDIP bit signals the Processor B that the Processor A has come out of reset. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • RDIP bit is set to "1" after the Processor A comes out of reset. The interrupt generated by a Processor A reset de-assertion is ORed with the Processor B general purpose interrupt 3. The Processor B general purpose interrupt 3 is issued when the Processor A comes out of reset if the interrupt is enabled by the RDIE bit.<br>• To clear the RDIP bit, write "1", which also clears general purpose interrupt 3.<br>• When Processor B comes out of reset, the RDIP bit has value "0"(default). Processor B then monitors the reset of Processor A and will assert RDIP when reset deasserts.<br><br>0b - Processor A did not exit reset<br>1b - Processor A exited from reset |
| 8<br>FUP | FUP<br>MUB Flags Update Pending. (Read-only)<br>• FUP bit is set to "1" when the MUB side sends a Flags Update request to the MUA side.<br>• A Flags Update request is generated when there is a change to the Fn[2:0] bits of the MUB CR register. No flag update changes are allowed while the FUP bit is set to "1". Any write to the Fn[2:0] bits of the MUB CR register, while the FUP bit is set to "1", will not generate a Flags Update event, and the Fn[2:0] bits will stay unchanged.<br>• FUP bit is cleared when this Flags Update request is internally acknowledged (that the flag is updated) from the MU MUA side, and during MU reset.<br><br>0b - No flags updated, initiated by the MUB, in progress (default)<br>1b - MUB initiated flags update, processing |
| 7<br>HRIP | HRIP<br>Processor B Hardware reset interrupt pending<br>• HRIP bit indicates if Processor B had been reset by HR bit in MUA_CCR<br>• HRIP is set to "1" if Processor A set HR in MUA_CCR to "1" and HRM in MUB_CCR is cleared.<br>• Write "1" to clear HRIP.<br><br>0b - MUA didn't issue hardware reset to Processor B<br>1b - MUA had initiated a hardware reset to Processor B through HR bit. |
| 6-5<br>— | Reserved |
| 4<br>EP | EP<br>MUB Side Event Pending. (Read-only)<br>• EP bit is set to "1" when the MUB side mechanism sends an event update request to the MUA side.<br>• EP bit is cleared when the event update acknowledge is received. An "event" is any hardware message that is reflected in the MUA SR register on the MUA side (for example, "transmit register 0 written"). During normal operations, you do not have to deal with the state of the EP bit because the event update mechanism works automatically.<br>• To ensure events have been posted to MUA before entering STOP mode, you should verify that the EP bit is cleared. If EP bit is set to "1", you should wait and continue to poll it (EP bit) before entering STOP mode.<br>• Reading the MUB SR register (to check the EP bit) should be the last access to the MU that should be performed before entering STOP or WAIT modes; otherwise, the EP bit may be set by subsequent additional actions.<br>• The EP bit is cleared when the MU resets.<br><br>0b - The MUB side event is not pending (default).<br>1b - The MUB side event is pending. |
| 3<br>NMIC | NMIC<br>Processor B Non-Maskable-Interrupt Clear. (Write-only)<br>• NMIC bit is used by the Processor B-side MU Non-Maskable Interrupt service routine to clear the non-maskable interrupt. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • Writing NMIC bit as "1" signals the MU to clear the NMI bit in the MUA CR register, thus de-asserting the interrupts and enabling the NMI bit to receive another interrupt.<br>• The NMIC bit is always read as "0"; therefore it (NMIC bit) cannot be polled. The NMIC bit can only be used as part of the interrupt service routine, in which the NMIC bit should only be written as "1" once.<br>• The NMIC bit is cleared when the MU resets.<br><br>0b - Default<br>1b - Writing "1" clears the NMI bit in the MUA CR register. |
| 2-0<br>Fn | Fn<br><br>For n = {0, 1, 2} MUB Side Flag n. (Read-only)<br>• Fn bit is the MUB side flag that reflects the values written to the Fn bit in the MUA control register.<br>• Every time that the Fn bit in the MUA CR register is written, the Fn bit in the MUA CR register write event updates the Fn bit in the MUB SR register after the event update latency, which is measured in terms of the number of clocks of the MUA and the MUB.<br><br>000b - Fn bit in the MUA CR register is written 0 (default).<br>001b - Fn bit in the MUA CR register is written 1. |

## 40.2.2.7 Control Register (CR)

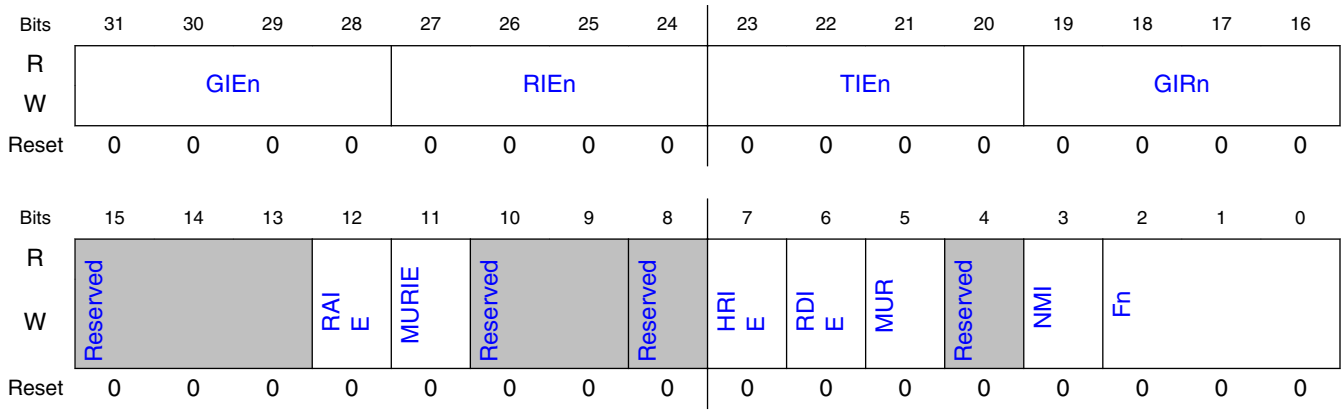
### 40.2.2.7.1 Offset

| Register | Offset |
|---|---|
| CR | 64h |

### 40.2.2.7.2 Function

Use the Control Register (CR, 32-bit, read-write) to enable the MU interrupts on the MUB-side, and trigger events and interrupts on the MUA-side (general purpose interrupt, flag update).

## 40.2.2.7.3 Diagram




## 40.2.2.7.4 Fields

| Field | Function |
|---|---|
| 31-28<br>GIEn | GIEn<br>For n = {0, 1, 2, 3} MUB General Purpose Interrupt Enable n. (Read-Write)<br>• GIEn bit enables MUB General Interrupt n.<br>• If GIEn bit is set to "1" (enabled), then a General Interrupt n request is issued when the GIPn bit in the MUB SR register is set to "1".<br>• If GIEn is cleared (disabled), then the value of the GIPn bit is ignored and no General Interrupt n request will be issued.<br>• GIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUB General Interrupt n. (default)<br>0001b - Enables MUB General Interrupt n. |
| 27-24<br>RIEn | RIEn<br>For n = {0, 1, 2, 3} MUB Receive Interrupt Enable n. (Read-Write)<br>• RIEn bit enables MUB Receive Interrupt n.<br>• If RIEn bit is set to "1" (enabled), then an MUB Receive Interrupt n request is issued when the RFn bit in the MUB SR register is set to "1".<br>• If RIEn bit is cleared (disabled), then the value of the RFn bit is ignored and no MUB Receive Interrupt n request will be issued.<br>• RIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUB Receive Interrupt n. (default)<br>0001b - Enables MUB Receive Interrupt n. |
| 23-20<br>TIEn | TIEn<br>For n = {0, 1, 2, 3} MUB Transmit Interrupt Enable n. (Read-Write)<br>• TIEn bit enables MUB Transmit Interrupt n.<br>• If TIEn bit is set to "1" (enabled), then an MUB Transmit Interrupt n request is issued when the TEn bit in the MUB SR register is set to "1".<br>• If TIEn bit is cleared (disabled), then the value of the TEn bit is ignored and no MUB Transmit Interrupt n request will be issued.<br>• TIEn bit is cleared when the MU resets.<br><br>0000b - Disables MUB Transmit Interrupt n. (default)<br>0001b - Enables MUB Transmit Interrupt n. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19-16<br>GIRn | GIRn<br>For n = {0, 1, 2, 3} MUB General Purpose Interrupt Request n. (Read-Write)<br>• Writing "1" to the GIRn bit sets the GIPn bit in the MUA SR register on the Processor A-side. If the GIEn bit in the MUA CR register is set to "1" on the MUA side, a General Purpose Interrupt n request is triggered.<br>• The GIRn bit is cleared if the GIPn bit (in the MUA SR register on the MUA side) is cleared by writing it (GIPn bit) as "1", thereby signalling the MUB that the interrupt was accepted (cleared by the software). The GIPn bit cannot be written as "0" on the MUB side.<br>• To ensure proper operations, you must verify that the GIRn bit is cleared (meaning that there is no pending interrupt) before setting it (GIRn bit).<br>• GIRn bit is cleared when the MU resets.<br><br>0000b - MUB General Interrupt n is not requested to the MUA (default).<br>0001b - MUB General Interrupt n is requested to the MUA. |
| 15-13<br>— | Reserved |
| 12<br>RAIE | RAIE<br>Processor A Reset Assertion Interrupt Enable. (Read-Write)<br>• RAIE bit enables Processor B General Interrupt 3.<br>• If RAIE bit is set to "1", then General Interrupt 3 request is issued to the Processor B when the RAIP bit in the MUB SR register is set to "1".<br>• If RAIE is cleared, then the value of the RAIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The RAIE bit is cleared when the MU resets.<br><br>0b - Disables Processor B-side General Purpose Interrupt 3 request due to Processor A reset assertion.<br>1b - Enables Processor B-side General Purpose Interrupt 3 request due to Processor A reset assertion. |
| 11<br>MURIE | MURIE<br>MU reset Interrupt Enable. (Read-Write)<br>• MURIE bit enables Processor B General Interrupt 3.<br>• If MURIE bit is set to "1", then General Interrupt 3 request is issued to the Processor B when the MURIP bit in the MUB SR register is set to "1".<br>• If MURIE is cleared, then the value of the MURIP bit is ignored and no General Interrupt 3 request will be issued.<br>• Only system reset can reset MURIE, MU reset cannot reset this bit.<br><br>0b - Disables Processor B-side General Purpose Interrupt 3 request due to MU reset issued by MUA.<br>1b - Enables Processor B-side General Purpose Interrupt 3 request due to MU reset issued by MUA. |
| 10-9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>HRIE | Processor B hardware reset interrupt enable<br>Processor B Hardware Reset Interrupt Enable. (Read-Write)<br>• HRIE bit enables Processor B General Interrupt 3.<br>• If HRIE bit is set to "1", then General Interrupt 3 request is issued to the Processor B when the HRIP bit in the MUB SR register is set to "1". |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • If HRIE is cleared, then the value of the HRIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The HRIE bit is cleared when the MU resets.<br><br>0b - Disables Processor B General Purpose Interrupt 3 request due to Processor A issued HR to Processor B.<br>1b - Enables Processor B General Purpose Interrupt 3 request due to Processor A issued HR to Processor B. |
| 6<br>RDIE | RDIE<br>Processor A Reset De-assertion Interrupt Enable. (Read-Write)<br>• RDIE bit enables Processor B General Interrupt 3.<br>• If RDIE bit is set to "1", then General Interrupt 3 request is issued to the Processor B when the RDIP bit in the MUB SR register is set to "1".<br>• If RDIE is cleared, then the value of the RDIP bit is ignored and no General Interrupt 3 request will be issued.<br>• The RDIE bit is cleared when the MU resets.<br><br>0b - Disables Processor B General Purpose Interrupt 3 request due to Processor A reset de-assertion.<br>1b - Enables Processor B General Purpose Interrupt 3 request due to Processor A reset de-assertion. |
| 5<br>MUR | MUR<br>MU Reset.<br>• Setting MUR bit to "1" resets both the Processor A and the Processor B sides of the MU module, forcing all control and status registers to return to their default values and all internal states to be cleared.<br>• The BOOT and RSTH fields will not be affected by the MUR bit.<br>• Before setting the MUR bit to "1", it is advisable to interrupt the Processor A , because setting the MUR bit may affect the ongoing Processor A program.<br>• MUR bit can only be written as "1".<br>• MUR bit is always read as "0".<br>• MUR bit is cleared during the MU reset sequence.<br><br>0b - N/A. Self clearing bit (default).<br>1b - Asserts the MU reset. |
| 4<br>— | Reserved |
| 3<br>NMI | NMI<br>MUA Non-maskable Interrupt. (Read-Write)<br>• When NMI bit is set to "1", it initiates a Non-Maskable Interrupt to the Processor A.<br>• NMI bit is cleared by the MU after the MUA asserts the NMIC bit in the MUA SR register.<br>• After the NMI bit is cleared, the MUB can initiate another non-maskable interrupt to the MUA.<br>• The NMI bit is cleared when the MU resets.<br><br>0b - Non-maskable interrupt is not issued to the Processor A by the Processor B (default).<br>1b - Non-maskable interrupt is issued to the Processor A by the Processor B. |
| 2-0<br>Fn | Fn<br>For n = {0, 1, 2} MUB to MUA Flag n. (Read-Write)<br>• Fn bit is a read-write flag that is reflected in Fn bit in the MUA SR register on the MUA side.<br>• Fn bit is cleared when the MU resets.<br><br>000b - Clears the Fn bit in the SR register.<br>001b - Sets the Fn bit in the SR register. |

## 40.2.2.8 Core Control Register (CCR)

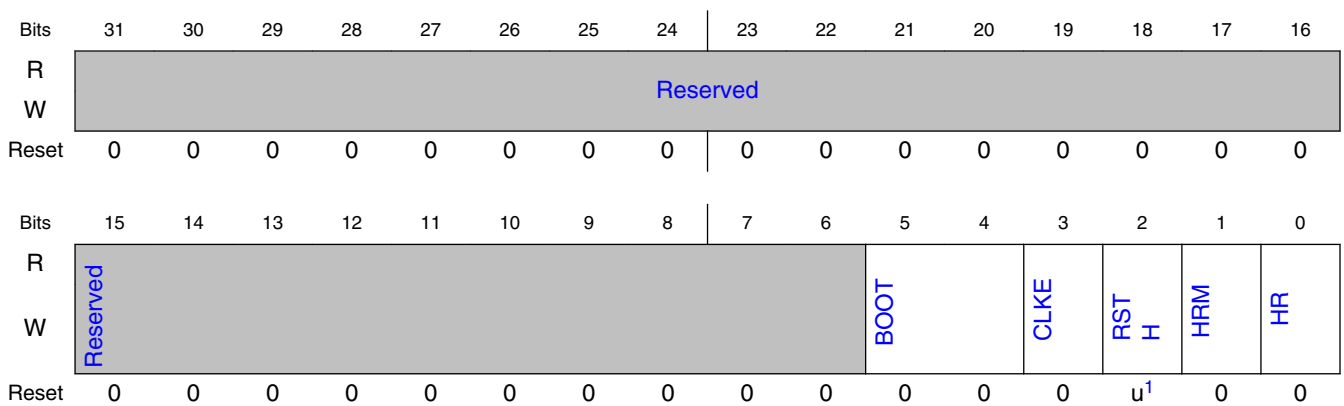
### 40.2.2.8.1 Offset

| Register | Offset |
|---|---|
| CCR | 68h |

### 40.2.2.8.2 Function

Use the Core Control Register (CCR, 32-bit, read-write) to enable the MUB to control the Processor on MUA-side.

### 40.2.2.8.3 Diagram




1. Reset value loaded during Processor B reset from Flash IFR.

### 40.2.2.8.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-4<br>BOOT | Slave Processor A Boot Config.<br>Configures the boot source for Processor A when Processor A is slave core.<br>00b - Boot from Pflash base<br>01b - Reserved<br>10b - Boot from CM4 RAM base<br>11b - Reserved |
| 3 | MUA clock enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CLKE | When set, forces MUA platform clock to remain enabled even after MUA-side has entered a stop mode. When clear, allows MUA to gate its platform clock whenever desired after entering a stop mode. Note that if MUB also enters a stop mode, then this bit is overidden to allow MUA to gate its platform clock.<br><br>0b - MUA platform clock gated when MUA-side enters a stop mode.<br>1b - MUA platform clock kept running after MUA-side enters a stop mode, until MUB also enters a stop mode. |
| 2<br><br>RSTH | Processor A Reset Hold<br><br>When set, will cause the Processor A to be held in reset following any reset event. When cleared, allows the Processor A to be released from reset.<br><br>Setting this bit will not force the Processor A into reset. It will only hold Processor A if it resets itself.<br><br>0b - Release Processor A from reset<br>1b - Hold Processor A in reset |
| 1<br><br>HRM | When set, HR bit in MUA CCR has no effect<br><br>Processor B Hardware Reset Mask. (Read-Write)<br>• The Processor A can give a hardware reset to the Processor B by setting the HR bit in the MUA CCR Register to “1”.<br>• When the HRM bit is set to “1” by the Processor B, the HR reset issued by the Processor A is masked (disabled by the Processor B).<br>• When the HRM bit is cleared, the HR reset issued by the Processor A to the Processor B is not masked (enabled by the Processor B).<br><br>0b - HR bit in MUA CCR is not masked, enables the hardware reset to the Processor B (default after hardware reset).<br>1b - HR bit in MUA CCR is masked, disables the hardware reset request to the Processor B. |
| 0<br><br>HR | HR<br><br>Processor A Hardware Reset. (Read-Write)<br>• HR bit asserts and de-asserts the hardware reset of the Processor A.<br>• Set HR bit to "1" to start a hardware reset of the Processor A.<br>• Clear the HR bit to de-assert the Processor A hardware reset input.<br>• Assert the HR bit for a minimum of 3 clock cycles. The RAIP bit in MUB SR register indicates the state of the Processor on MUA-side. As soon as the Processor A goes into Reset (MUB RAIP bit is set to "1"), the HR bit can be de-asserted.<br>• Strobe-setting the HR bit will not cause an internal MU reset but will be routed outside MU to Processor A reset logic<br>• The HR reset issued by the Processor B to the Processor A is maskable by the Processor A (according to the settings of the HRM bit in the MUA CCR register). If the HRM bit (in the MUA CCR register) is set to "1", then the HR reset is masked; if the HRM bit (in the MUA CCR register) is cleared (default), then the HR reset is enabled.<br>• The HR bit does not return to the reset value during the software (MUR) reset.<br><br>0b - De-assert Hardware reset to the Processor A. (default)<br>1b - Assert Hardware reset to the Processor A. |

## 40.3 Functional Description

**Table 40-1. Major Features of the MU**

| Major Feature | Description |
|---|---|
| Interprocessor Interrupts | • The MU has 12 interrupt sources on each side (Processor A-side, Processor B-side) that are used for signaling the other processor. The interrupts can be used for notification of RX/TX events and general-purpose signaling between the processors. |
| MU Reset | • The Processor A/B can issue a reset to the entire MU, using a control bit (MUR) in the Processor A/B Control Register (CR).<br>• The MUR bit is a self-clearing bit. |
| Processor B/A Boot Configuration | • The Boot Source for Processor B/A can be configured with the BOOT bits in the CCR register. See BOOT bits in CCR register for details |
| Processor B/A Reset Hold | • Processor B/A can be held in reset following any reset event. This is done by setting the RSTH bit in the CCR register.<br>• Processor B/A will be released from reset when this bit is cleared.<br>• The value at reset is loaded from Flash IFR. |
| Processor A/B Clock Enable | • The Processor A/B platform clock can be enabled to continue running when Processor A/B enters Stop Mode, until Processor B/A also enters Stop Mode. This allows Processor B/A to continue accessing peripherals on Processor A/B's AIPS bus even when it has entered a Stop Mode. |
| Status and Control Communications between Cores | • The MU provides a way for the two cores to communicate using the status and control registers present on both the Processor B and Processor A sides of the MU.<br>• The status register of one MU side reflects the status of the other MU side.<br>• The control register is used for control operations, such as enabling an interrupt and sending an interrupt to the other processor. |
| Synchronized Message Transfers between Cores | • The transfer of data messages between cores uses transmit empty and receive full flags provided on both sides of the MU.<br>• The update of these transmit and receive flags is accomplished using a synchronization mechanism. There is inherent latency between updating the flag on one side and reflecting its status on other side. For more about latency, see Event Update Timing |
| Accessing Shared Memory Directly and Avoiding Collisions | • For sending data or messages from one MU-side to the other MU-side, the MU provides 4 transmit registers and 4 receive registers on each side of the MU.<br>• The Processor A or Processor B can access shared memory resources of the SoC directly. However, to avoid simultaneous access to shared memory by both cores, the MU provides a method (to prevent simultaneous access) using interrupts and transmit-receive registers for both processors. |
| Memory-Mapped Registers | • The MU is connected as a peripheral under the Peripheral bus on both sides—on the Processor A-side, the Processor A Peripheral Bus, and on the Processor B-side, the Processor B Peripheral Bus. |

### 40.3.1 Processor A Side Memory-Mapping

The messaging, control, and status registers of the Processor A-side for the MU are mapped to the Processor A memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

### 40.3.2 Processor B Side Memory-Mapping

The messaging, control, and status registers of the Processor B-side for the MU are mapped to the Processor B memory as a regular peripheral. The Peripheral bus data bus is 32 bits wide inside the MU module.

### 40.3.3 MU Messaging

The MU provides 32-bit status and control registers to the Processor B and Processor A sides for control operations (such as interrupts and reset), and for status checking of the other MU-side.

For messaging, the MU has four, 32-bit write-only transmit registers and four, 32-bit read-only receive registers on the Processor B and Processor A-sides. These registers are used for sending messages to each other. These messages can be also be controlled using the 3 general purpose flags provided in the control and status registers of either MU-side.

#### 40.3.3.1 Programmer Model

The messaging logic is used in conjunction with external memory. You have various messaging methods, which you can use to implement a messaging protocol. Some of these messages could mean “I have just written a message of N words, starting at offset X in the memory,” or “I have just finished reading the previous data block that was sent.” Having the messaging logic independent from the memory array does not restrict you to a predefined hardware protocol. On the other hand, the software needed to manage the messaging is short and straightforward.

Most of the messaging mechanisms are symmetric; they are duplicated and are available on both the Processor B-side and the Processor A-side. The messaging mechanisms are:

- Four, 32-bit write-only transmit registers, which are each reflected in four, read-only receive registers in the other processor’s side. You can use these registers to transfer 32-bit word messages or frame information of messages written to the shared memory (number of words, initial address, and message type code).

- A write to a transmit register on the transmitter side clears a “transmitter empty” bit in the Status Register on the transmitter side, and sets a “receiver full” bit in the Status Register on the receiver side. The setting of the bit at the receiver side can optionally trigger an interrupt at the receiver side (maskable receive interrupt).
- A read of one of the receive registers at the receiver side clears the “receiver full” bit in the Status Register at the receiver side, and sets the “transmitter empty” bit in the Status Register on the transmitter side. The setting of the “transmitter empty” bit can optionally trigger an interrupt at the transmitter side (maskable transmit interrupt).
- Four general purpose flags are reflected in the Status Register on the receiver side
- A read/write access to any reserved location and a write to a read-only register on the Processor A-side of the MU will generate a module transfer error acknowledge to the Processor A.
- A read/write access to any reserved location and write to a read-only register on the Processor B-side of the MU will generate a module transfer error acknowledge to the Processor B.

#### 40.3.3.2 Messaging Examples

The following are messaging examples:

- **Passing short messages:** Transmit register(s) can be used to pass short messages from one to four words in length. For example, when a four-word message is desired, only one of the registers needs to have its corresponding interrupt enable bit set at the receiver side; the message’s first three words are written to the registers whose interrupt is masked, and the fourth word is written to the other register (which triggers an interrupt at the receiver side).
- **Passing frame information:** Transmit registers can be used to pass frame information for long messages written to the shared system (SDRAM and SyncFLASH). Such frame information normally includes a start address, number of words, and perhaps a message type code.
- **Passing event notices and requests:** Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the general interrupts, such as acknowledging that a long message was read from the shared system memory.
- **Passing fixed length data:** Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general interrupt (Processor A or Processor B) to signal the other processor that the data is ready.
- **Passing announcements:** The three flags can be used by a processor to announce its current program state or other billboard messages to the other processor.

Figure 40-2 shows the MU registers schematic.

## 40.3.4 Low Power Modes

This section describes the low power operating modes of the MU module.

### 40.3.4.1 Processor Low Power Modes

The Processors have four power modes:

- Run
- WAIT
- STOP/VLPS
- LLS/VLLS

The Processor can be awakened from a low-power mode by any enabled Processor side MU interrupt, as reflected in the xSR “status” register (RF0–3, TE0–3, GIP0–3 bits are set) and enabled in the xCR control register. Using these bits, the Processor can actively control when to wake the other Processor.

While the Processor is in STOP/VLPS mode (such that the xSR register bits cannot be updated with events), special logic drives the enabled Processor interrupts directly from the other Processor-side (instead of from the xSR register).

While the Processor is in STOP/VLPS mode, the asynchronous Processor interrupt will be asserted to wake the Processor:

- If any transmit data register of the other Processor-side is full, because of a write to it (transmit data register); that is, its “empty” bit in the xSR register is cleared while its corresponding receive interrupt is enabled on the Processor-side.
- If any receive data register of the other Processor-side is empty, because of a read on the other Processor -side; that is, its “full” bit in the xSR register is cleared while its corresponding transmit interrupt is enabled on the Processor-side.
- If any general purpose interrupt is set in the xCR register while the corresponding interrupt is enabled on the Processor-side.
- If the other Processor issues a non-maskable interrupt to the Processor.

The logic enables the other Processor to operate independently while the Processor is in any power mode (including STOP/VLPS). However, the Processor power mode change protocol should be handled with care regarding:

- The interrupts that are enabled on the Processor-side
- The events that could be triggered by the other Processor-side
- The compatibility with the other Processor protocol of entering STOP/VLPS mode

If the Processor is in STOP/VLPS mode and an event on the other Processor is triggered, the EP bit (in the xSR register) will remain high until the Processor wakes up.

Before entering STOP/VLPS mode, the Processor programmer should verify that the EP bit (in the xSR register) is cleared. This check is needed to ensure that all pending updates from the Processor, including the power mode change when STOP/VLPS or WAIT is executed, will be updated in the xSR register.

- If the other Processor is in STOP/VLPS mode or LLS/VLLS mode, the EP bit (in the xSR register) may be stuck high; in this case, the Processor need not check the EP bit before entering STOP/VLPS mode.

## 40.3.5 Event Update Timing

Each processor's MU messaging side (Processor A or Processor B) has a hardware mechanism to send "event update requests" to the other processor's side. An "event" is considered when any information change should be reflected at the Status Register of the receiving processor. The event update latency is the delay between the event being ready at one processor and the resulting update at the Status Register of the other processor.

- The minimum event latency is "1 clock of the sending side" + "2 1/2 clocks of the receiving side". The minimum case is if there is no event pending when the new event occurs.
- The maximum event latency is "6 clocks of the sending side" + "6 1/2 clocks of the receiving side." The maximum case is if the event occurred just after a previous event was sent to the other side. The event update latency will vary between the above-mentioned minimum and maximum latencies, depending on the time at which the subsequent event is triggered.

## 40.3.6 Interrupts

The MU controls the Processor B interrupt requests to the Processor A, and the Processor A interrupt requests to the Processor B.

This section describes all the interrupts that the module generates.

### 40.3.6.1 Interrupts to the Processors

There are 12 interrupt sources from the MU to the Processors:

- Four receive interrupts (asserted when the Processors receive full bits are set and enabled in the xCR register) for each of the receive registers
- Four transmit interrupts (asserted when the Processor transmit empty bits are set and enabled in the xCR register) for each of the transmit registers
- Four general purpose interrupts (asserted when the GIP bits are set and enabled in the xCR register)

All the interrupts are maskable in the Processor Control Register (xCR). The MU does not assume any internal priority of these interrupts. Multiple interrupts (for example, Receive 0 and Receive 1 interrupts or any of the transmit and general purpose interrupts) can be asserted at one time. The priority of these interrupts should be resolved by the interrupt controller at the chip level.

The General Purpose Interrupt Pending bits (GIP0, GIP1, GIP2, and GIP3) should be cleared by the software (as part of the interrupt service routine) to de-assert the request to the interrupt controller.

#### 40.3.6.2 General Purpose Interrupt Clearing Sequence

When a Processor writes to the general interrupt bit (GIR), the write event is synchronized to the other Processor clock to set the general interrupt request pending bit (GIP). When the GIP bit is set, and if the general purpose interrupt is enabled on the transmitting Processor side (GIE bit is set), then the receiving Processor general purpose interrupt is issued to the transmitting Processor. The transmitting Processor clears this interrupt by writing a “1” on the GIP bit. The interrupt is de-asserted as soon as the GIP bit is written. The write event of the GIP bit is synchronized to the other Processor clock. The synchronized signal clears the GIR bit. The software should not write the GIR bit again until the GIR bit is cleared.

### 40.3.7 Interrupt Messaging Protocols

#### 40.3.7.1 Messaging Protocols using Interrupts

The example below describes a four-word messaging sequence sent by the Processor to the other Processor.

In this example, the first, second, and third receive interrupts are disabled, and the fourth receive interrupt is enabled. We write registers sequentially for n = 0, 1, 2, 3. For n = 0, 1, 2, the interrupts are disabled, therefore no interrupt will go to the other core (although interrupt conditions occur). For n = 3, the interrupt is enabled, and the last Receive Interrupt request is generated.

1. Write Sequence

- The Processor writes the message information sequentially to its Transmit Registers 0, 1, 2.
- When the write to the Transmit Register 3 occurs, the RF3 bit of the xSR is set after synchronization, and it immediately trigger the Receive Full 3 interrupt to the other Processor.

2. Read Sequence
   - The other Processor receives the Receive Full 3 interrupt and starts reading the message transferred from the receive registers.
   - After Receive Register 3 is read, the interrupt bit is cleared.

The following table and diagram describe the messaging model using transmit/receive registers and interrupt messaging protocol.

**Table 40-2. Interrupt Messaging Protocol (Generalized)**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor A Data write | A data write to the TRn register by Processor A is immediately reflected in the Processor B RRn register. |
| 2 | Clear Tx Empty bit and Set Rx Full bit | The data write to the TRn register<br>• Clears the transmitter empty bit (TEn) in the Processor A Transmit Status Register<br>• Sets the receiver full bit (RFn) in the Processor B Receive Status Register |
| 3 | Generate Receive Interrupt request | The setting of the receiver full bit (RFn) in the Receive Status Register generates a Receive Interrupt request to Processor B. |
| 4 | Processor B Data read | After receiving the Receive Interrupt request, Processor B performs a data read of the RRn register. |
| 5 | Clear Rx Full bit and Set Tx Empty bit | Reading the data out of the RRn register<br>• Clears the receiver full bit (RFn) in the Processor B Receive Status Register<br>• Sets the transmitter empty bit (TEn) in the Processor A Transmit Status Register |
| 6 | Generate Transmit Interrupt request | The setting of the transmitter empty bit (TEn) in the Transmit Status Register generates a Transmit Interrupt request to Processor A. |

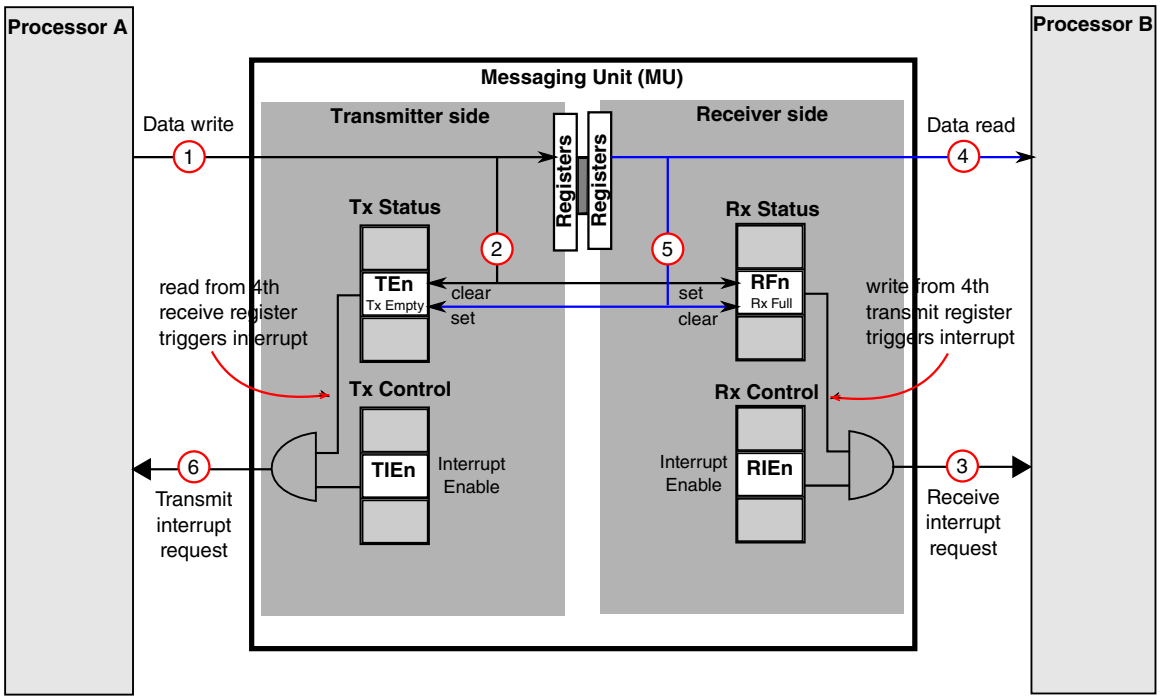



**Figure 40-3. Messaging Model Using Transmit and Receive Registers**

The messaging hardware can be used by software to implement messaging protocols for a wide array of message types. Full support is given for both interrupt and polling management schemes.

### 40.3.7.2 Messaging Protocols using Event Interrupts

Events and requests that do not include data words can be signaled from the Processor B to the Processor A using the two general interrupts.

Formatted data with a fixed length can be written in predetermined locations in the shared memory. A processor can use a general purpose interrupt to signal the other processor that the data is ready.

The three flags can be used by a processor to announce to the other processor the program state it is currently in, or to announce similar messages.

Table 40-3 and Figure 40-4 describe the event sequence when the Processor triggers an interrupt.

**Table 40-3. Interrupt Messaging Protocol (Generalized)**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor A sets General Interrupt request bit | Processor A sets its associated General Interrupt request bit (GIRn = 1) in the control register (ACR). |
| 2 | General Interrupt Request Pending status bit is set | The General Interrupt Request Pending status bit (GIPn) in the status register (BSR) is set to "1" |
| 3 | General Interrupt request to Processor B is generated | Setting the GIPn bit generates the General Interrupt request to Processor B (Interrupt Request Enable bit, GIEn, must be set for Processor B) |
| 4 | Processor B reads status register | The Processor B reads the GIPn bit in the BSR register. |
| 5 | Processor B services the interrupt | - |
| 6 | Processor B sets GIPn bit to clear interrupt | The Processor B writes "1" to the corresponding GIPn bit to clear the interrupt |
| 7 | GIRn bit is cleared | Setting the GIPn bit to "1" clears the General Interrupt request bit (GIRn) in the Processor A control register (ACR). |

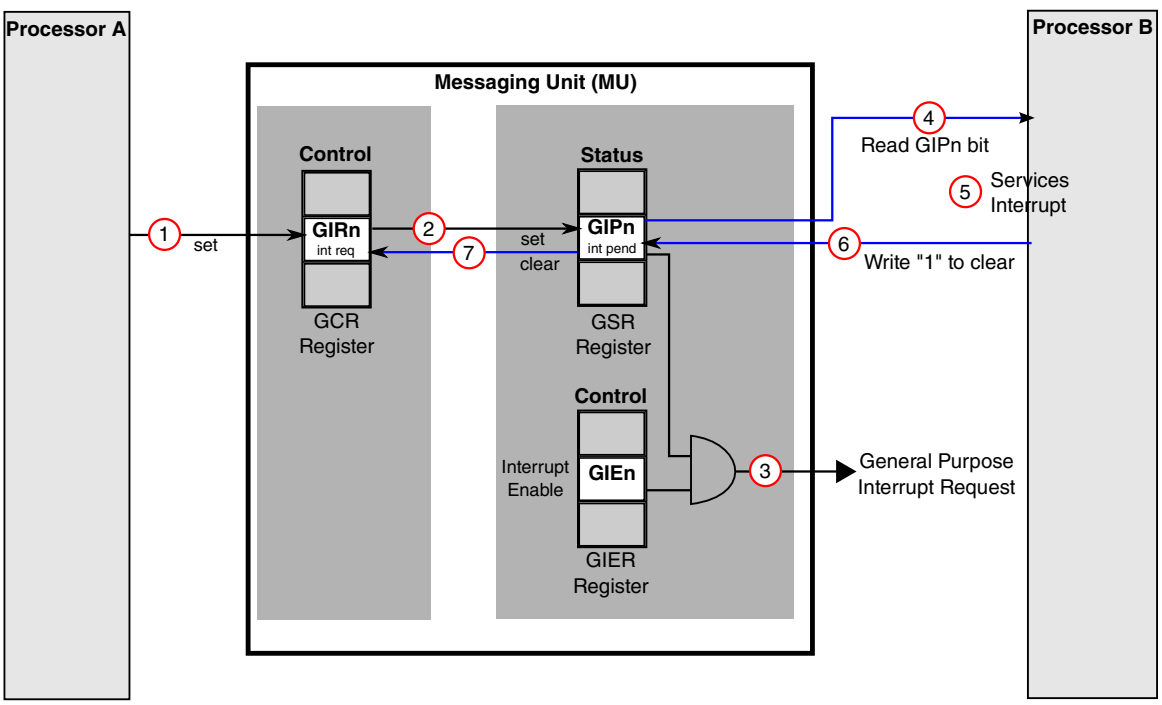



**Figure 40-4. Messaging Model Using a General Purpose Interrupt**

## 40.3.8 Exclusive Access to Shared Memory

You can use the MU to signal one processor about its current access to the shared memory, so that the data is not overwritten by the other processor during the exclusive memory access period.

The following tables describe the signaling protocol that the Processor A uses to inform the Processor B about its current access (write) to the shared memory, assuming that the set of bits and registers (GIRn bit, RRn register, TRn register) are reserved to support exclusive access to the shared memory protocol.

**Table 40-4. How the Processor A Performs an Exclusive Access to Shared Memory**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor A sends GIRn request to Processor B using Processor A control register | When the Processor A wants to perform an exclusive access to the shared memory, the Processor A sends an GIR0 request to the Processor B. |
| 2 | Processor A sends an exclusive-access request using a transmit data register (TRn) | The Processor A will send an exclusive-access request (command, location, and length of target access) to Processor B using a selected transmit data register (TR0). |
| 3 | Processor A waits for a dedicated interrupt from Processor B | The Processor A waits for a dedicated interrupt (as an acknowledgement) triggered by the Processor B before proceeding. |
| 4 | Processor A accesses shared memory | After receiving a dedicated interrupt from the Processor B, Processor A proceeds. |

**Table 40-5. How the Processor B Scans for Transaction Information**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor B receives an interrupt from a receive data register (RRn) | - |
| 2 | Processor B reads the receive data register (RRn) | - |
| 3 | Processor B scans the receive data register contents | For transaction information (whether Processor A has requested an exclusive-access) |

**Table 40-6. How the Processor B Accepts Exclusive Access by Processor A**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor B triggers a dedicated interrupt | Processor B acknowledges the Processor A request by triggering a dedicated interrupt (ack) to the Processor A. |
| 2 | Processor B sends a code message to Processor A | Along with the acknowledge interrupt, the Processor B sends a code message to the Processor A through the selected transmit register (TRn). The message informs the Processor A that it can exclusively access the shared memory. |

**Table 40-7. How the Processor B Rejects Exclusive Access by Processor A**

| Sequence | Action | Description |
|---|---|---|
| 1 | Processor B ignores Processor A request for exclusive access | If the Processor B does not want to give go-ahead permission to the Processor A, Processor B ignores the exclusive access request. |

## 40.3.9 Packet Data Transfers

The following example describes the packet transfer sequence between the Processor B and Processor A subsystems:

**Table 40-8. Packet Data Transfer Sequence**

| Action | Sequence | Description |
|---|---|---|
| Processor B requests DMA | 1 | The Processor B sends a DMA request to initiate the packet data transfer |
| DMA data transfer | 2 | DMA acknowledges. |
| | 3 | DMA starts transferring data from the specified Processor B location to the specified shared memory |
| | 4 | DMA interrupts the Processor B to signal that the packet transfer has finished. |
| Processor B informs Processor A that data is in shared memory | 5 | Using an MU Processor B-side transmit register, the Processor B sends a packet information message to the Processor A to inform the Processor A of the arrival of new packet data that is stored in shared memory. The message contains the command, location, and length of packet data information. |
| Processor A receives interrupt | 6 | The Processor A receives an interrupt (assuming its corresponding Processor A MU-side receive interrupt is enabled), and the pending processing task becomes active and processes packet data from memory. |
| Processor A reads data, writes data | 7 | The Processor A reads or processes packet data from shared memory. |
| | 8 | The Processor A writes the result from packet processing to a separate buffer. |
| Processor A informs Processor B that transfer is finished | 9 | After the processing of the packet data finishes, the Processor A informs the Processor B (using the MU Processor A-side transmit register, ATRn). |
| Processor A sends interrupt to Processor B (request for more data) | 10 | The Processor B receives the next interrupt from the Processor A, in which the Processor A requests more packet data. |

## 40.3.10 Resets

The MU has two sources of reset, and each reset has a different function from the MU or system perspective.

- One asynchronous system that is connected to both sides of the MU interface.
- One programmable hardware reset (MUR bit) in the CR and CCR register (on the Processor A and B side).

**Table 40-9. MU programmable resets**

| Reset | Description |
|---|---|
| Processor A/B MU reset | • Processor A/B MU Reset bit (MUR) of the CR register<br>• The MUR reset affects the messaging section on both the Processor A and the Processor B sides. The MUR reset causes most of control and status registers to return to their default values and all internal states to be cleared.<br>• The instruction immediately following assertion of the MUR bit should not write to MU registers. Such a write may be overwritten by the reset sequence and the register will remain with the reset value. You should wait at least one instruction (after assertion of the MUR bit) before attempting a write to MU registers. |

**NOTE**

MUR bit assertion may cause unpredictable behavior if, for example, the other Processor is concurrently testing an MU register bit (TE bit in Processor B/A SR register). Before asserting the MUR bit, you should verify that the other Processor is not presently engaged in an MU signalling activity.

## 40.4 Software Restrictions

This section describes certain software restrictions when accessing the MU.

### 40.4.1 General Restrictions

This section lists the restrictions that apply to both the sides (Processor A, Processor B) of the MU.

#### 40.4.1.1 Write-After-Write to a Transmit Register

A write to a transmit register signals the receiver side that data is ready for retrieval.

- Writing to the transmit register again without verifying that the data was retrieved is prohibited, because the transmitter side has no way of knowing the exact time that the receiver will attempt to retrieve the data.
- Before attempting to write the transmit register again, the transmitter side should wait for a “Transmitter Empty” interrupt, or should poll the “Transmitter Empty” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being read on the receiver side of the MU.

#### 40.4.1.2 Read-After-Read from a Receive Register

A read of a receive register signals the transmitter side that data can be written to that register. In the same way, the receiver processor should not read a receive register before receiving a “Receiver Full” interrupt or polling the “Receiver Full” bit in the Status Register.

- Reading the receive register again without verifying that the data was written is prohibited, because the receiver side has no way of knowing the exact time that the transmitter will attempt to write the data.
- Before attempting to read the receive register again, the receiver side should wait for a “Receiver Full” interrupt, or should poll the “Receiver Full” bit in the Status Register.
- Failure to follow this restriction may result in the wrong data being written on the transmitter side of the MU.

### 40.4.2 Processor Restrictions

This section lists the restrictions that apply each side of the processor in the MU.

#### 40.4.2.1 Before Entering Low Power Mode

Before entering Low Power mode, the Processor should verify that the Processor Event Pending (EP) bit in the Status Register is cleared.

- If the Event Pending bit (EP) is still set to “1”, then the Processor should wait and poll the EP bit until it is cleared, before executing the LPM instruction.
- Note that if the other Processor is in Low Power mode (programmed for clock gating in CCM), the EP bit may be stuck high. In this case, the other Processor clock must be turned ON to get the EP bit cleared before the Processor can enter Low Power mode.
- To discover which power mode the other Processor is in, the Processor can check the PM bits in the xSR register.

#### 40.4.2.2 Before Setting a General Interrupt Request Bit (GIR0–3)

Before setting a General Interrupt Request bit (GIR0–3), you must verify that the GIRn bit is cleared, which means that a general interrupt is not pending. Generally, setting the GIRn bit while the bit is set to “1” will be ignored, but in some cases it may issue a second interrupt. This restriction is meant to prevent this indeterministic behavior.

### 40.4.2.3 Reset Bit Restrictions

The reset bit (MUR, HR) restrictions are:

- Before asserting the MUR bit in the CR register, verify that the Processor B/A-side is not engaged in some MU activity.
- Do not write to an MU register in the instruction immediately after the assertion of the MUR bit in the CR register, because the written data can be overridden by the reset value.

# Chapter 41
# Watchdog timer (WDOG)

## 41.1 Chip-specific WDOG information

### 41.1.1 WDOG clocks

The following table shows the WDOG module clocks and corresponding chip clocks.

**Table 41-1. WDOG module clocks and corresponding chip clocks**

| Module clock | Chip clock |
|---|---|
| BUS_CLK | SLOW_CLK |
| LPO_CLK | LPO_CLK |
| INTCLK | SIRC_CLK |
| ERCLK | SOSC_CLK |

### 41.1.2 WDOG low-power modes

This table shows the WDOG low-power modes and the corresponding chip low-power modes.

**Table 41-2. WDOG low-power modes**

| Module mode | Chip mode |
|---|---|
| Wait | Wait, VLPW |
| Standby | Stop, VLPS |
| Stop | Stop, VLPS |
| Power Down | LLS, VLLSx |

**NOTE**

To enable the WDOG module when the chip is in Stop mode, write ones to both the STNDBYEN bit and the STOPEN bit of the Watchdog Status and Control Register High.

## 41.2 Introduction

The Watchdog Timer (WDOG) module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the MCU.

### 41.2.1 Features

Features of the WDOG module include:

- Configurable clock source inputs independent from the bus clock
  - Bus clock (slow clock)
  - LPO clock
  - INTCLK (internal clock)
  - ERCLK (external reference clock)
- Programmable timeout period
  - Programmable 16-bit timeout value
  - Optional fixed 256 clock prescaler when longer timeout periods are needed
- Robust write sequence for counter refresh
  - Refresh sequence of writing 0xA602 and then 0xB480
- Window mode option for the refresh mechanism
  - Programmable 16-bit window value
  - Provides robust check that program flow is faster than expected
  - Early refresh attempts trigger a reset.
- Optional timeout interrupt to allow post-processing diagnostics

- Interrupt request to CPU with interrupt vector for an interrupt service routine (ISR)
- Forced reset occurs 128 bus clocks after the interrupt vector fetch.

- Configuration bits are write-once-after-reset to ensure watchdog configuration cannot be mistakenly altered.
- Robust write sequence for unlocking write-once configuration bits
  - Unlock sequence of writing 0xC520 and then 0xD928 for allowing updates to write-once configuration bits
  - Software must make updates within 128 bus clocks after unlocking and before WDOG closing unlock window.

## 41.2.2 Block diagram

The following figure shows a block diagram of the WDOG module.

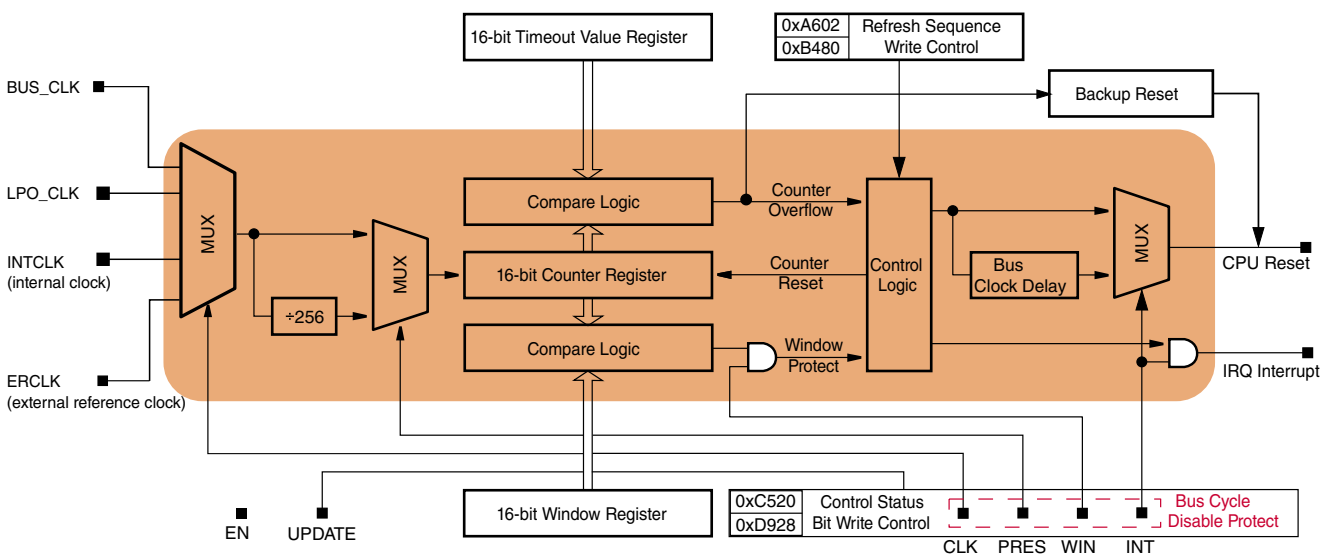



**Figure 41-1. WDOG block diagram**

# 41.3 Memory map and register definition

## 41.3.1 WDOG0 register descriptions

### 41.3.1.1 WDOG0 Memory map

WDOG0 base address: 4002_A000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Watchdog Control and Status Register (CS) | 32 | RW | 0000_2980h |
| 4h | Watchdog Counter Register (CNT) | 32 | RW | 0000_0000h |
| 8h | Watchdog Timeout Value Register (TOVAL) | 32 | RW | 0000_0400h |
| Ch | Watchdog Window Register (WIN) | 32 | RW | 0000_0000h |

### 41.3.1.2 Watchdog Control and Status Register (CS)

#### 41.3.1.2.1 Offset

| Register | Offset |
|---|---|
| CS | 0h |

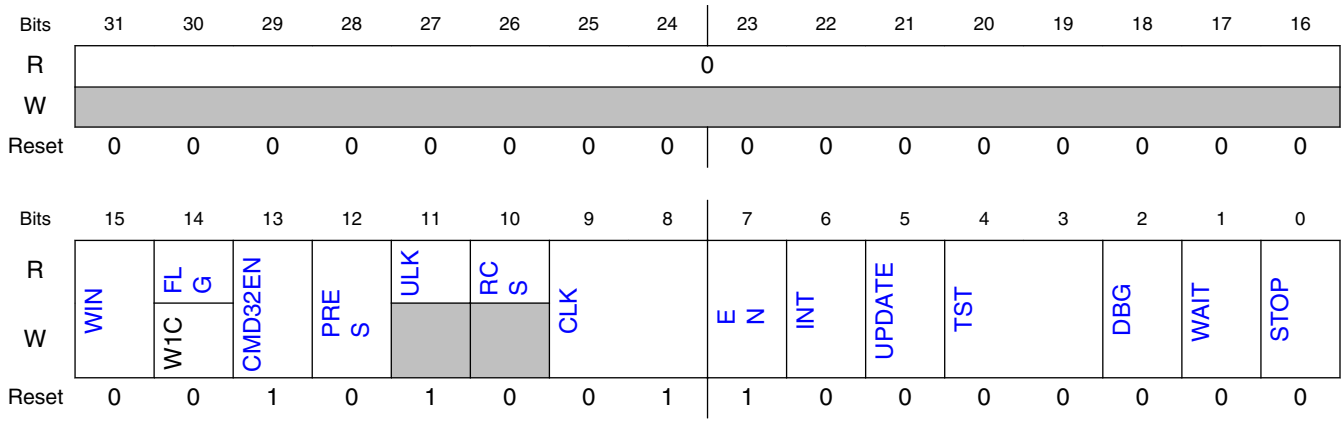
#### 41.3.1.2.2 Function

This section describes the function of Watchdog Control and Status Register.

**NOTE**

TST is cleared (0:0) on POR only. Any other reset does not affect the value of this field.

### 41.3.1.2.3 Diagram




### 41.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>WIN | Watchdog Window<br>This write-once bit enables window mode. See the Window mode section.<br>0b - Window mode disabled.<br>1b - Window mode enabled. |
| 14<br>FLG | Watchdog Interrupt Flag<br>This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it.<br>0b - No interrupt occurred.<br>1b - An interrupt occurred. |
| 13<br>CMD32EN | Enables or disables WDOG support for 32-bit (otherwise 16-bit or 8-bit) refresh/unlock command write words<br>This is write-once field, and the user needs to unlock WDOG after writing this field for reconfiguration.<br>0b - Disables support for 32-bit refresh/unlock command write words. Only 16-bit or 8-bit is supported.<br>1b - Enables support for 32-bit refresh/unlock command write words. 16-bit or 8-bit is NOT supported. |
| 12<br>PRES | Watchdog prescaler<br>This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock. (The block diagram shows this clock divider option.)<br>0b - 256 prescaler disabled.<br>1b - 256 prescaler enabled. |
| 11<br>ULK | Unlock status<br>This read-only bit indicates whether WDOG is unlocked or not.<br>0b - WDOG is locked.<br>1b - WDOG is unlocked. |
| 10<br>RCS | Reconfiguration Success |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This read-only bit indicates whether the reconfiguration is successful or not. Default reset value is 0.This bit is set when new configuration takes effect, and is cleared by successful unlock command.<br>0b - Reconfiguring WDOG.<br>1b - Reconfiguration is successful. |
| 9-8<br>CLK | Watchdog Clock<br>This write-once field indicates the clock source that feeds the watchdog counter. See the Clock source section.<br>00b - Bus clock<br>01b - LPO clock<br>10b - INTCLK (internal clock)<br>11b - ERCLK (external reference clock) |
| 7<br>EN | Watchdog Enable<br>This write-once bit enables the watchdog counter to start counting.<br>0b - Watchdog disabled.<br>1b - Watchdog enabled. |
| 6<br>INT | Watchdog Interrupt<br>This write-once bit configures the watchdog to immediately generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), before forcing a reset. After the interrupt vector fetch (which comes after the reset-triggering event), the reset occurs after a delay of 128 bus clocks.<br>0b - Watchdog interrupts are disabled. Watchdog resets are not delayed.<br>1b - Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks from the interrupt vector fetch. |
| 5<br>UPDATE | Allow updates<br>This write-once bit allows software to reconfigure the watchdog without a reset.<br>0b - Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset.<br>1b - Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence. |
| 4-3<br>TST | Watchdog Test<br>Enables the fast test mode. The test mode allows software to exercise all bits of the counter to demonstrate that the watchdog is functioning properly. See the Fast testing of the watchdog section.<br>This write-once field is cleared (0:0) on POR only. Any other reset does not affect the value of this field.<br>00b - Watchdog test mode disabled.<br>01b - Watchdog user mode enabled. (Watchdog test mode disabled.) After testing the watchdog, software should use this setting to indicate that the watchdog is functioning normally in user mode.<br>10b - Watchdog test mode enabled, only the low byte is used. CNT[CNTLOW] is compared with TOVAL[TOVALLOW].<br>11b - Watchdog test mode enabled, only the high byte is used. CNT[CNTHIGH] is compared with TOVAL[TOVALHIGH]. |
| 2<br>DBG | Debug Enable<br>This write-once bit enables the watchdog to operate when the chip is in debug mode.<br>0b - Watchdog disabled in chip debug mode.<br>1b - Watchdog enabled in chip debug mode. |
| 1<br>WAIT | Wait Enable<br>This write-once bit enables the watchdog to operate when the chip is in wait mode.<br>0b - Watchdog disabled in chip wait mode.<br>1b - Watchdog enabled in chip wait mode. |
| 0<br>STOP | Stop Enable<br>This write-once bit enables the watchdog to operate when the chip is in stop mode. |

| Field | Function |
|---|---|
| | 0b - Watchdog disabled in chip stop mode.<br>1b - Watchdog enabled in chip stop mode. |

## 41.3.1.3 Watchdog Counter Register (CNT)

### 41.3.1.3.1 Offset

| Register | Offset |
|---|---|
| CNT | 4h |

### 41.3.1.3.2 Function

This section describes the watchdog counter register.

The watchdog counter register provides access to the value of the free-running watchdog counter. Software can read the counter register at any time.

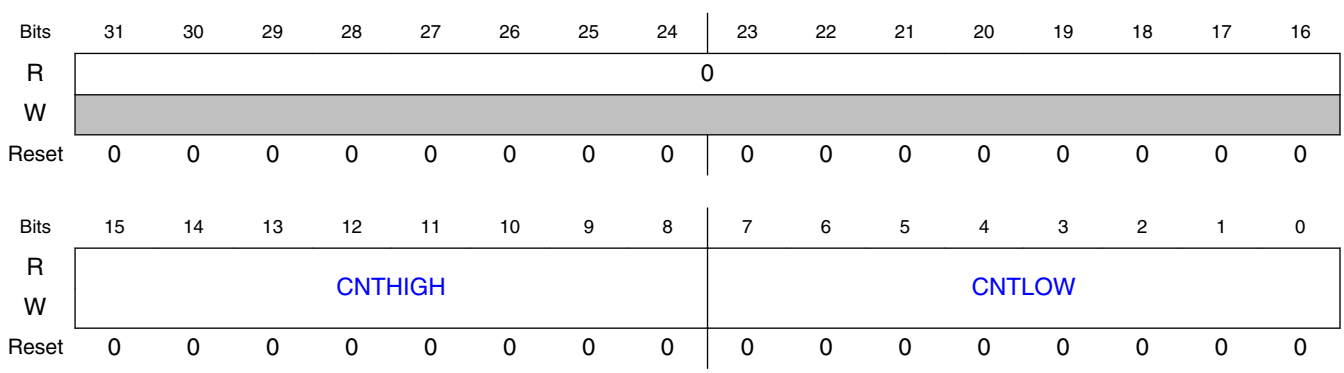
Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions:

1. The *refresh sequence* resets the watchdog counter to 0x0000. See the "Refreshing the Watchdog" section.
2. The *unlock sequence* allows the watchdog to be reconfigured without forcing a reset (when CS[UPDATE] = 1). See the "Configure for reconfigurable" section.

**NOTE**

All other writes to this register are illegal and force a reset.

### 41.3.1.3.3 Diagram



#### 41.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CNTHIGH | High byte of the Watchdog Counter |
| 7-0<br>CNTLOW | Low byte of the Watchdog Counter |

### 41.3.1.4 Watchdog Timeout Value Register (TOVAL)

#### 41.3.1.4.1 Offset

| Register | Offset |
|---|---|
| TOVAL | 8h |

#### 41.3.1.4.2 Function

This section describes the watchdog timeout value register. TOVAL contains the 16-bit value used to set the timeout period of the watchdog.

The watchdog counter (CNT) is continuously compared with the timeout value (TOVAL). If the counter reaches the timeout value, the watchdog forces a reset triggering event.

**NOTE**

Do not write 0 to the TOVAL register (if CS[TST]=11b, then TOVALHIGH cannot be written as 0; if CS[TST]=10b, then TOVALLOW cannot be 0); otherwise, the watchdog always generates a reset.

#### 41.3.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOVALHIGH | | | | | | | | TOVALLOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 41.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TOVALHIGH | High byte of the timeout value |
| 7-0<br>TOVALLOW | Low byte of the timeout value |

### 41.3.1.5 Watchdog Window Register (WIN)

#### 41.3.1.5.1 Offset

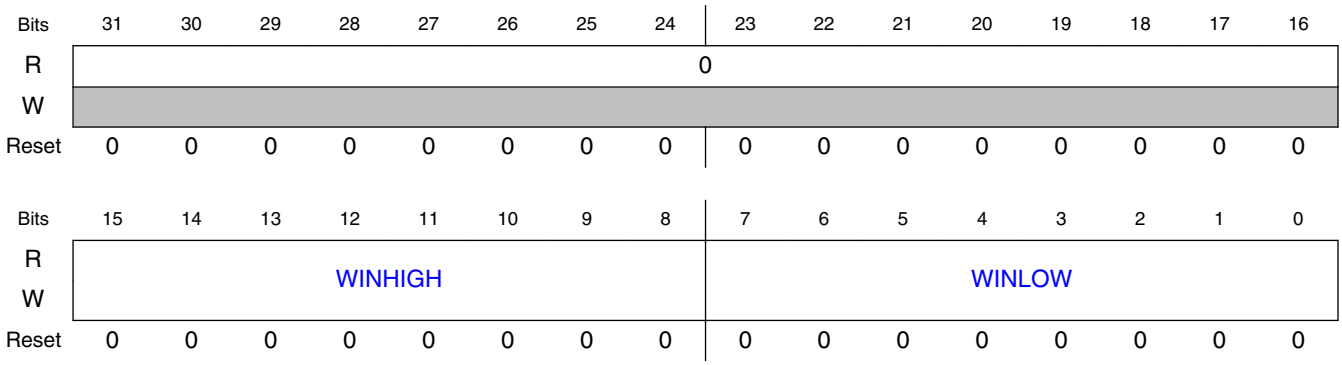
| Register | Offset |
|---|---|
| WIN | Ch |

#### 41.3.1.5.2 Function

This section describes the watchdog window register. When window mode is enabled (CS[WIN] is set), The WIN register determines the earliest time that a refresh sequence is considered valid. See the Watchdog refresh mechanism section.

The WIN register value must be less than the TOVAL register value.

#### 41.3.1.5.3 Diagram




#### 41.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>WINHIGH | High byte of Watchdog Window |
| 7-0<br>WINLOW | Low byte of Watchdog Window |

## 41.3.2 WDOG1 register descriptions

### 41.3.2.1 WDOG1 Memory map

WDOG1 base address: 4102_6000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Watchdog Control and Status Register (CS) | 32 | RW | 0000_2980h |
| 4h | Watchdog Counter Register (CNT) | 32 | RW | 0000_0000h |
| 8h | Watchdog Timeout Value Register (TOVAL) | 32 | RW | 0000_0400h |
| Ch | Watchdog Window Register (WIN) | 32 | RW | 0000_0000h |

## 41.3.2.2 Watchdog Control and Status Register (CS)

### 41.3.2.2.1 Offset

| Register | Offset |
|---|---|
| CS | 0h |

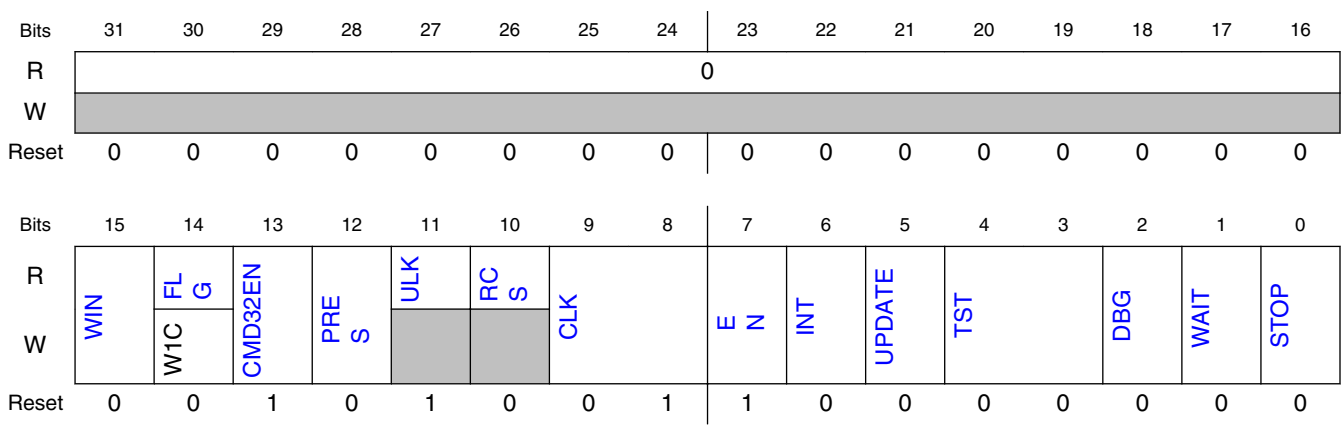
### 41.3.2.2.2 Function

This section describes the function of Watchdog Control and Status Register.

**NOTE**

TST is cleared (0:0) on POR only. Any other reset does not affect the value of this field.

### 41.3.2.2.3 Diagram




### 41.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>WIN | Watchdog Window<br>This write-once bit enables window mode. See the Window mode section.<br>0b - Window mode disabled.<br>1b - Window mode enabled. |
| 14<br>FLG | Watchdog Interrupt Flag<br>This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it.<br>0b - No interrupt occurred. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - An interrupt occurred. |
| 13<br>CMD32EN | Enables or disables WDOG support for 32-bit (otherwise 16-bit or 8-bit) refresh/unlock command write words<br>This is write-once field, and the user needs to unlock WDOG after writing this field for reconfiguration.<br>0b - Disables support for 32-bit refresh/unlock command write words. Only 16-bit or 8-bit is supported.<br>1b - Enables support for 32-bit refresh/unlock command write words. 16-bit or 8-bit is NOT supported. |
| 12<br>PRES | Watchdog prescaler<br>This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock. (The block diagram shows this clock divider option.)<br>0b - 256 prescaler disabled.<br>1b - 256 prescaler enabled. |
| 11<br>ULK | Unlock status<br>This read-only bit indicates whether WDOG is unlocked or not.<br>0b - WDOG is locked.<br>1b - WDOG is unlocked. |
| 10<br>RCS | Reconfiguration Success<br>This read-only bit indicates whether the reconfiguration is successful or not. Default reset value is 0.This bit is set when new configuration takes effect, and is cleared by successful unlock command.<br>0b - Reconfiguring WDOG.<br>1b - Reconfiguration is successful. |
| 9-8<br>CLK | Watchdog Clock<br>This write-once field indicates the clock source that feeds the watchdog counter. See the Clock source section.<br>00b - Bus clock<br>01b - LPO clock<br>10b - INTCLK (internal clock)<br>11b - ERCLK (external reference clock) |
| 7<br>EN | Watchdog Enable<br>This write-once bit enables the watchdog counter to start counting.<br>0b - Watchdog disabled.<br>1b - Watchdog enabled. |
| 6<br>INT | Watchdog Interrupt<br>This write-once bit configures the watchdog to immediately generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), before forcing a reset. After the interrupt vector fetch (which comes after the reset-triggering event), the reset occurs after a delay of 128 bus clocks.<br>0b - Watchdog interrupts are disabled. Watchdog resets are not delayed.<br>1b - Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks from the interrupt vector fetch. |
| 5<br>UPDATE | Allow updates<br>This write-once bit allows software to reconfigure the watchdog without a reset.<br>0b - Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset.<br>1b - Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence. |
| 4-3<br>TST | Watchdog Test<br>Enables the fast test mode. The test mode allows software to exercise all bits of the counter to demonstrate that the watchdog is functioning properly. See the Fast testing of the watchdog section. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This write-once field is cleared (0:0) on POR only. Any other reset does not affect the value of this field.<br>00b - Watchdog test mode disabled.<br>01b - Watchdog user mode enabled. (Watchdog test mode disabled.) After testing the watchdog, software should use this setting to indicate that the watchdog is functioning normally in user mode.<br>10b - Watchdog test mode enabled, only the low byte is used. CNT[CNTLOW] is compared with TOVAL[TOVALLOW].<br>11b - Watchdog test mode enabled, only the high byte is used. CNT[CNTHIGH] is compared with TOVAL[TOVALHIGH]. |
| 2<br>DBG | Debug Enable<br>This write-once bit enables the watchdog to operate when the chip is in debug mode.<br>0b - Watchdog disabled in chip debug mode.<br>1b - Watchdog enabled in chip debug mode. |
| 1<br>WAIT | Wait Enable<br>This write-once bit enables the watchdog to operate when the chip is in wait mode.<br>0b - Watchdog disabled in chip wait mode.<br>1b - Watchdog enabled in chip wait mode. |
| 0<br>STOP | Stop Enable<br>This write-once bit enables the watchdog to operate when the chip is in stop mode.<br>0b - Watchdog disabled in chip stop mode.<br>1b - Watchdog enabled in chip stop mode. |

## 41.3.2.3 Watchdog Counter Register (CNT)

### 41.3.2.3.1 Offset

| Register | Offset |
|---|---|
| CNT | 4h |

### 41.3.2.3.2 Function

This section describes the watchdog counter register.

The watchdog counter register provides access to the value of the free-running watchdog counter. Software can read the counter register at any time.

Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions:

1. The *refresh sequence* resets the watchdog counter to 0x0000. See the "Refreshing the Watchdog" section.
2. The *unlock sequence* allows the watchdog to be reconfigured without forcing a reset (when CS[UPDATE] = 1). See the "Configure for reconfigurable" section.

**NOTE**

All other writes to this register are illegal and force a reset.

#### 41.3.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTHIGH | | | | | | | | CNTLOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 41.3.2.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CNTHIGH | High byte of the Watchdog Counter |
| 7-0<br>CNTLOW | Low byte of the Watchdog Counter |

### 41.3.2.4 Watchdog Timeout Value Register (TOVAL)

#### 41.3.2.4.1 Offset

| Register | Offset |
|---|---|
| TOVAL | 8h |

#### 41.3.2.4.2 Function

This section describes the watchdog timeout value register. TOVAL contains the 16-bit value used to set the timeout period of the watchdog.

The watchdog counter (CNT) is continuously compared with the timeout value (TOVAL). If the counter reaches the timeout value, the watchdog forces a reset triggering event.

**NOTE**

Do not write 0 to the TOVAL register (if CS[TST]=11b, then TOVALHIGH cannot be written as 0; if CS[TST]=10b, then TOVALLOW cannot be 0); otherwise, the watchdog always generates a reset.

#### 41.3.2.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOVALHIGH | | | | | | | | TOVALLOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 41.3.2.4.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TOVALHIGH | High byte of the timeout value |
| 7-0<br>TOVALLOW | Low byte of the timeout value |

### 41.3.2.5 Watchdog Window Register (WIN)

#### 41.3.2.5.1 Offset

| Register | Offset |
|---|---|
| WIN | Ch |

#### 41.3.2.5.2 Function

This section describes the watchdog window register. When window mode is enabled (CS[WIN] is set), The WIN register determines the earliest time that a refresh sequence is considered valid. See the Watchdog refresh mechanism section.

The WIN register value must be less than the TOVAL register value.

#### 41.3.2.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WINHIGH | | | | | | | | WINLOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 41.3.2.5.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>WINHIGH | High byte of Watchdog Window |
| 7-0<br>WINLOW | Low byte of Watchdog Window |

## 41.4 Functional description

The WDOG module provides a fail safe mechanism to ensure the system can be reset to a known state of operation in case of system failure, such as the CPU clock stopping or there being a run away condition in the software code. The watchdog counter runs continuously off a selectable clock source and expects to be serviced (refreshed) periodically. If it is not, it generates a reset triggering event.

**The timeout period, window mode, and clock source are all programmable but must be configured within 128 bus clocks after a reset.**

### 41.4.1 Clock source

The watchdog counter has the following clock source options selected by programming CS[CLK]:

- bus clock
- Low-Power Oscillator clock (LPO_CLK)
- internal clock
- external clock

The options allow software to select a clock source independent of the bus clock for applications that need to meet more robust safety requirements. Using a clock source other than the bus clock ensures that the watchdog counter continues to run if the bus clock is somehow halted; see Backup reset.

**NOTE**

The *default* clock source of WDOG should be enabled after its funcitonal reset is deasserted, for the WDOG module to function properly.

An optional fixed prescaler for all clock sources allows for longer timeout periods. When CS[PRES] is set, the clock source is prescaled by 256 before clocking the watchdog counter.

The following table summarizes the different watchdog timeout periods which could be available, as an example.

**Table 41-3. Watchdog timeout availability (example)**

| Reference clock | Prescaler | Watchdog time-out availability |
|---|---|---|
| LPO_CLK | Pass through | ~1 ms–65.5 s (if LPO_CLK = 1 kHz); (~1 ms–65.5 s)/128 (if LPO_CLK = 128 kHz). [1] |
| | ÷256 | ~256 ms–16,777.2 s (if LPO_CLK = 1 kHz); ~2 ms–131.1 s (if LPO_CLK = 128 kHz). |
| Internal clock 8 MHz | Pass through | 125 ns–8.1925 ms |
| | ÷256 | 32 µs–2.09728 s |
| 1 MHz (from bus or external) | Pass through | 1 µs–65.54 ms |
| | ÷256 | 256 µs–16.777 s |
| 20 MHz (from bus or external) | Pass through | 50 ns–3.277 ms |
| | ÷256 | 12.8 µs–838.8 ms |

1. The default timeout value after reset is approximately 1024 ms (if LPO_CLK = 1 kHz), or 1024/128 ms (if LPO_CLK = 128 kHz).

**NOTE**

When the programmer switches clock sources during reconfiguration, the watchdog hardware holds the counter at zero for 2.5 periods of the previous clock source and 2.5 periods of the new clock source after the configuration time period ( 128 bus clocks) ends. This delay ensures a smooth transition before restarting the counter with the new configuration.

## 41.4.2 Watchdog refresh mechanism

The watchdog resets the MCU if the watchdog counter is not refreshed. A robust refresh mechanism makes it very unlikely that the watchdog can be refreshed by runaway code.

To refresh the watchdog counter, software must execute a refresh write sequence before the timeout period expires. In addition, if window mode is used, software must not start the refresh sequence until after the time value set in the WIN register. See the following figure.

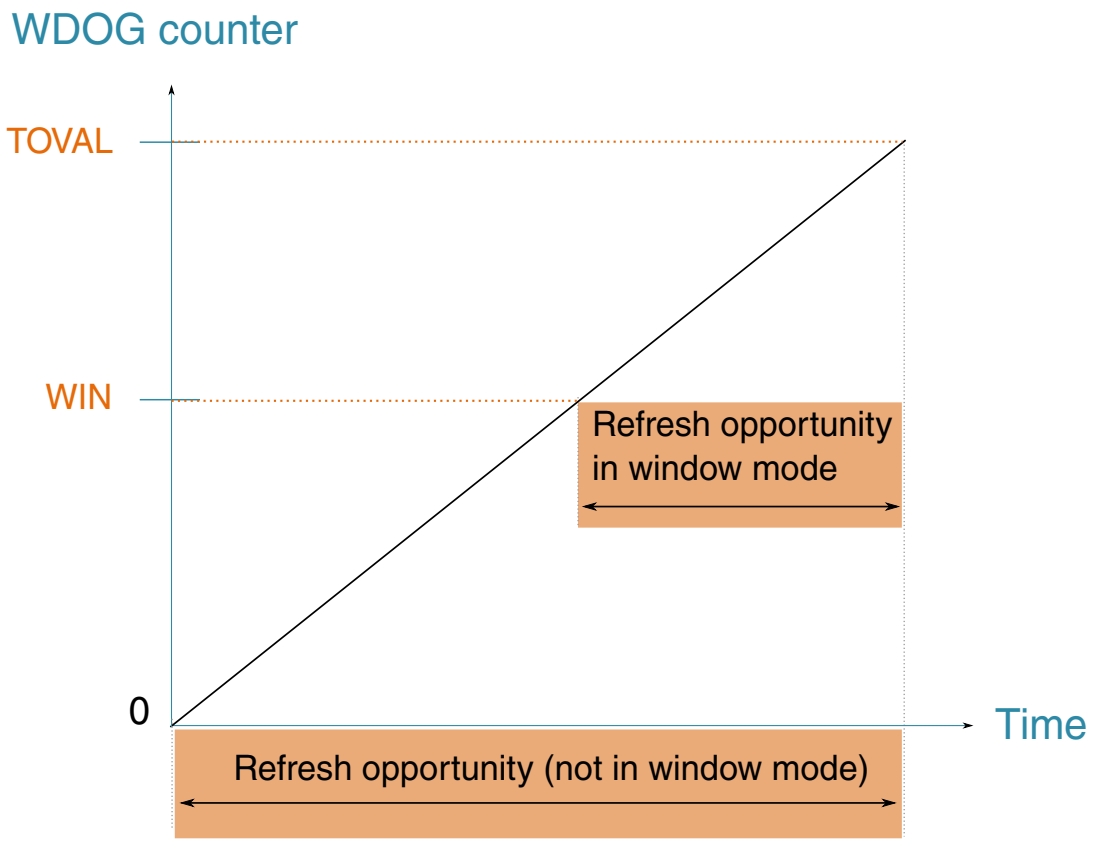



**Figure 41-2. Refresh opportunity for the Watchdog counter**

### 41.4.2.1 Window mode

Software finishing its main control loop faster than expected could be an indication of a problem. Depending on the requirements of the application, the WDOG can be programmed to force a reset when refresh attempts are early.

When Window mode is enabled, the watchdog must be refreshed after the counter has reached a minimum expected time value; otherwise, the watchdog resets the MCU. The minimum expected time value is specified in the WIN register. Setting CS[WIN] enables Window mode.

### 41.4.2.2 Refreshing the Watchdog

The refresh write sequence can be

- either two 16-bit writes ( 0xA602, 0xB480) or four 8-bit writes (0xA6, 0x02, 0xB4, 0x80; applicable to Cortex-M core cases) if WDOG_CS[CMD32EN] is 0;
- one 32-bit write (0xB480_A602) if WDOG_CS[CMD32EN] is 1.

to the CNT register. Both methods must occur before the WDG timeout; otherwise, the watchdog resets the MCU.

**Note**

Before starting the refresh sequence, disable the global interrupts. Otherwise, an interrupt could effectively invalidate the refresh sequence, if the interrupt occurs before the refresh writes finish. After the sequence finishes, restore the global interrupt control state.

The example codes can be found at the end of this chapter.

## 41.4.3 Configuring the Watchdog

### 41.4.3.1 Configuring the Watchdog Once

**All watchdog control bits, timeout value, and window value are write-once after reset *within 128 bus clocks*. This means that after a write has occurred they cannot be changed unless a reset occurs.** This is guaranteed by the user configuring the window and timeout value first, followed by the other control bits, and ensuring that CS[UPDATE] is also set to 0.

This provides a robust mechanism to configure the watchdog and ensure that a runaway condition cannot mistakenly disable or modify the watchdog configuration after configured.

The new configuration takes effect only after all registers except CNT are written after reset. Otherwise, the WDOG uses the reset values by default. If window mode is not used (CS[WIN] is 0), writing to WIN is not required to make the new configuration take effect.

### 41.4.3.2 Reconfiguring the Watchdog

In some cases (like when supporting a bootloader function), you may want to reconfigure or disable the watchdog, *without forcing a reset first*.

- By setting CS[UPDATE] to 1 on the initial configuration of the watchdog after a reset, you can reconfigure the watchdog at any time by executing an unlock sequence.
- Conversely, if CS[UPDATE] remains 0, the only way to reconfigure the watchdog is by initiating a reset.

The unlock sequence is similar to the refresh sequence but uses different values.

#### 41.4.3.2.1 Unlocking the Watchdog

The unlock sequence is a write to the CNT register of 0xC520 followed by 0xD928 at any time after the watchdog has been configured. On completing the unlock sequence, the user must reconfigure the watchdog within 128 bus clocks; otherwise, the watchdog closes the unlock window.

**NOTE**

Due to the 128 bus clocks requirement for reconfiguring the watchdog, some delays must be inserted before executing STOP or WAIT instructions after reconfiguring the watchdog. This ensures that the watchdog's new configuration takes effect before the MCU enters low power mode. Otherwise, the MCU may not be waken up from low power mode.

The example codes can be found at end of this chapter.

### 41.4.4 Using interrupts to delay resets

- **When interrupts are enabled (CS[INT] = 1):** After a reset-triggering event (like a counter timeout or invalid refresh attempt), the watchdog first generates an interrupt request. Next, the watchdog delays 128 bus clocks (from the interrupt vector fetch, not the reset-triggering event) before forcing a reset, to allow the interrupt service routine (ISR) to perform tasks (like analyzing the stack to debug code).
- **When interrupts are disabled (CS[INT] = 0):** the watchdog does not delay the forcing a reset.

### 41.4.5 Backup reset

**NOTE**

A clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the backup reset function is not available.

The backup reset function is a safeguard feature that independently generates a reset in case the main WDOG logic loses its clock (the bus clock) and can no longer monitor the counter. If the watchdog counter overflows twice in succession (without an intervening reset), the backup reset function takes effect and generates a reset.

Backup reset becomes valid when interrupt is enabled and the watchdog clock is from non bus clock. If interrupt is enabled, once the bus clock is cut off before exiting interrupt routine, the normal watchdog reset will be blocked. Under this case, the second overflow will cause backup reset directly.

### 41.4.6 Functionality in debug and low-power modes

By default, the watchdog is not functional in Debug mode, Wait mode, or Stop mode. However, the watchdog can remain functional in these modes as follows:

- **For Debug mode,** set CS[DBG]. (This way the watchdog is functional in Debug mode even when the CPU is held by the Debug module.)
- **For Wait mode,** set CS[WAIT].
- **For Stop mode,** set CS[STOP], CS[WAIT], and ensure the clock source is active in Stop mode.

**NOTE**

The watchdog can generate interrupt in Stop mode.

For Debug mode and Stop mode, in addition to the above configurations, a clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the watchdog cannot function.

### 41.4.7 Fast testing of the watchdog

Before executing application code in safety critical applications, users are required to test that the watchdog works as expected and resets the MCU. Testing every bit of a 16-bit counter by letting it run to the overflow value takes a relatively long time (64 k clocks).

To help minimize the startup delay for application code after reset, the watchdog has a feature to test the watchdog more quickly by splitting the counter into its constituent byte-wide stages. The low and high bytes are run independently and tested for timeout against the corresponding byte of the timeout value register. (For complete coverage when testing the high byte of the counter, the test feature feeds the input clock via the 8th bit of the low byte, thus ensuring that the overflow connection from the low byte to the high byte is tested.)

Using this test feature reduces the test time to 512 clocks (not including overhead, such as user configuration and reset vector fetches). To further speed testing, use a faster clock (such as the bus clock) for the counter reference.

On a power-on reset, the POR bit in the system reset register is set, indicating the user should perform the WDOG fast test.

### 41.4.7.1 Testing each byte of the counter

The test procedure follows these steps:

1. Program the preferred watchdog timeout value in the TOVAL register during the watchdog configuration time period.
2. Select a byte of the counter to test using the CS[TST] = 10b for the low byte; CS[TST] = 11b for the high byte.
3. Wait for the watchdog to timeout. Optionally, in the idle loop, increment RAM locations as a parallel software counter for later comparison. Because the RAM is not affected by a watchdog reset, the timeout period of the watchdog counter can be compared with the software counter to verify the timeout period has occurred as expected.
4. The watchdog counter times out and forces a reset.
5. Confirm the WDOG flag in the system reset register is set, indicating that the watchdog caused the reset. (The POR flag remains clear.)
6. Confirm that CS[TST] shows a test (10b or 11b) was performed.

If confirmed, the count and compare functions work for the selected byte. Repeat the procedure, selecting the other byte in step 2.

**NOTE**

CS[TST] is cleared by a POR only and not affected by other resets.

### 41.4.7.2 Entering user mode

After successfully testing the low and high bytes of the watchdog counter, the user can configure CS[TST] to 01b to indicate the watchdog is ready for use in application user mode. Thus if a reset occurs again, software can recognize the reset trigger as a real watchdog reset caused by runaway or faulty application code.

As an ongoing test when using the default clock source, software can periodically read the CNT register to ensure the counter is being incremented.

## 41.5 Application Information

The watchdog is enabled by default after reset. To disable or reconfigure the watchdog, it is better to be done before the first watchdog timeout. It is suggested to disable or reconfigure the watchdog at the very beginning of the software code, e.g. beginning of the startup or main function.

**NOTE**

When the watchdog is configured by user, it needs at least 2.5 periods of watchdog clock to take effect. This means interval between two configures by user must be larger than 2.5 clocks.

**NOTE**

When Chip startup from BOOT ROM then jump to flash, the watchdog would be disabled in the beginning of bootloader, and enabled when bootloader exits. If there is any code in the flash program want to reconfigure the watchdog, it must be run 2.5 watchdog clocks later after the bootloader exits.

To disable or reconfigure the watchdog without forcing a reset, WDOG_CS[UPDATE] bit must be set during the initial configuration of the WDOG module. Then, the unlock sequence can be used at any time within the timeout limit to reconfigure the watchdog.

### 41.5.1 Disable Watchdog

To disable the watchdog, first do unlock sequence, then unset the WDOG_CS[EN] bit. The code snippet below shows an example for 32-bit write.

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
WDOG_CS &= ~WDOG_CS_EN_MASK; //disable watchdog
EnableInterrupts; //enable global interrupt
```

### 41.5.2 Disable Watchdog after Reset

All of watchdog registers are unlocked by reset. Therefore, unlock sequence is unnecessary, but it needs to write all of watchdog registers to make the new configuration take effect. The code snippet below shows an example of disabling watchdog after reset.

```
DisableInterrupts; // disable global interrupt
WDOG_CS &= ~WDOG_CS_EN_MASK; // disable watchdog
WDOG_TOVAL= 0xFFFF;
while(WDOG_CS[ULK]);  // waiting for lock
while(~WDOG_CS[RCS]); // waiting for new configuration to take effect
EnableInterrupts; // enable global interrupt
```

## 41.5.3 Configure Watchdog

The watchdog can be configured once by set the WDOG_CS[UPDATE]=0. After that, the watchdog cannot be reconfigured until a reset. If set WDOG_CS[UPDATE]=1 when configuring the watchdog, the watchdog can be reconfigured without forcing a reset. The following example code shows how to configure the watchdog without window mode, clock source as LPO, interrupt enabled and timeout value to 256 clocks. The code snippet below shows an example for 32-bit write.

*Configure once*

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
while(WDOG_CS[ULK]==0);  //wait until registers are unlocked
WDOG_TOVAL = 256; //set timeout value
WDOG_CS = WDOG_CS_EN(1) | WDOG_CS_CLK(1) | WDOG_CS_INT(1) |
          WDOG_CS_WIN(0) | WDOG_CS_UPDATE(0);
while(WDOG_CS[RCS]==0); //wait until new configuration takes effect
EnableInterrupts; //enable global interrupt
```

*Configure for reconfigurable*

```
DisableInterrupts; //disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
while(WDOG_CS[ULK]==0);  //wait until registers are unlocked
WDOG_TOVAL = 256; //set timeout value
WDOG_CS = WDOG_CS_EN(1) | WDOG_CS_CLK(1) | WDOG_CS_INT(1) |
          WDOG_CS_WIN(0) | WDOG_CS_UPDATE(1);
while(WDOG_CS[RCS]==0); //wait until new configuration takes effect
EnableInterrupts; //enable global interrupt
```

## 41.5.4 Refreshing the Watchdog

To refresh the watchdog and reset the watchdog counter to zero, a refresh sequence is required. The code snippet below shows an example for 32-bit write.

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xB480A602; // refresh watchdog
EnableInterrupts; // enable global interrupt
```

# Chapter 42
# External Watchdog Monitor (EWM)

## 42.1 Chip-specific EWM information

### 42.1.1 EWM clocks

This table shows the EWM clocks and the corresponding chip clocks.

**Table 42-1. EWM clock connections**

| Module clock | Chip clock |
|---|---|
| Low Power Clock | 1 kHz LPO Clock |

### 42.1.2 EWM low-power modes

This table shows the EWM low-power modes and the corresponding chip low-power modes.

**Table 42-2. EWM low-power modes**

| Module mode | Chip mode |
|---|---|
| Wait | Wait, VLPW |
| Stop | Stop, VLPS, LLS |

### 42.1.3 $\overline{\text{EWM_OUT}}$ pin state in low power modes

During Wait, Stop and Power Down modes the $\overline{\text{EWM_OUT}}$ pin preserves its state before entering Wait or Stop mode. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode from Power Down, the pin returns to its reset state.

## 42.2 Introduction

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally assert the RESET_B pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM provides an independent EWM_OUT_b signal that when asserted resets or places an external circuit into a safe mode. The EWM_OUT_b signal is asserted upon the EWM counter time-out. An optional external input EWM_in is provided to allow additional control of the assertion of EWM_OUT_b signal.

### 42.2.1 Features

Features of EWM module include:

- Independent LPO_CLK clock source
- Programmable time-out period specified in terms of number of EWM LPO_CLK clock cycles.
- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of EWM_OUT_b.
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 15 peripheral bus clock cycles.

- One output port, EWM_OUT_b, when asserted is used to reset or place the external circuit into safe mode.
- One Input port, EWM_in, allows an external circuit to control the assertion of the EWM_OUT_b signal.

## 42.2.2 Modes of Operation

This section describes the module's operating modes.

### 42.2.2.1 Stop Mode

When the EWM is in stop mode, the CPU refreshes to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU refresh mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first refresh command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next *15* peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM refresh instructions.

### 42.2.2.2 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

### 42.2.3 Block Diagram

This figure shows the EWM block diagram.

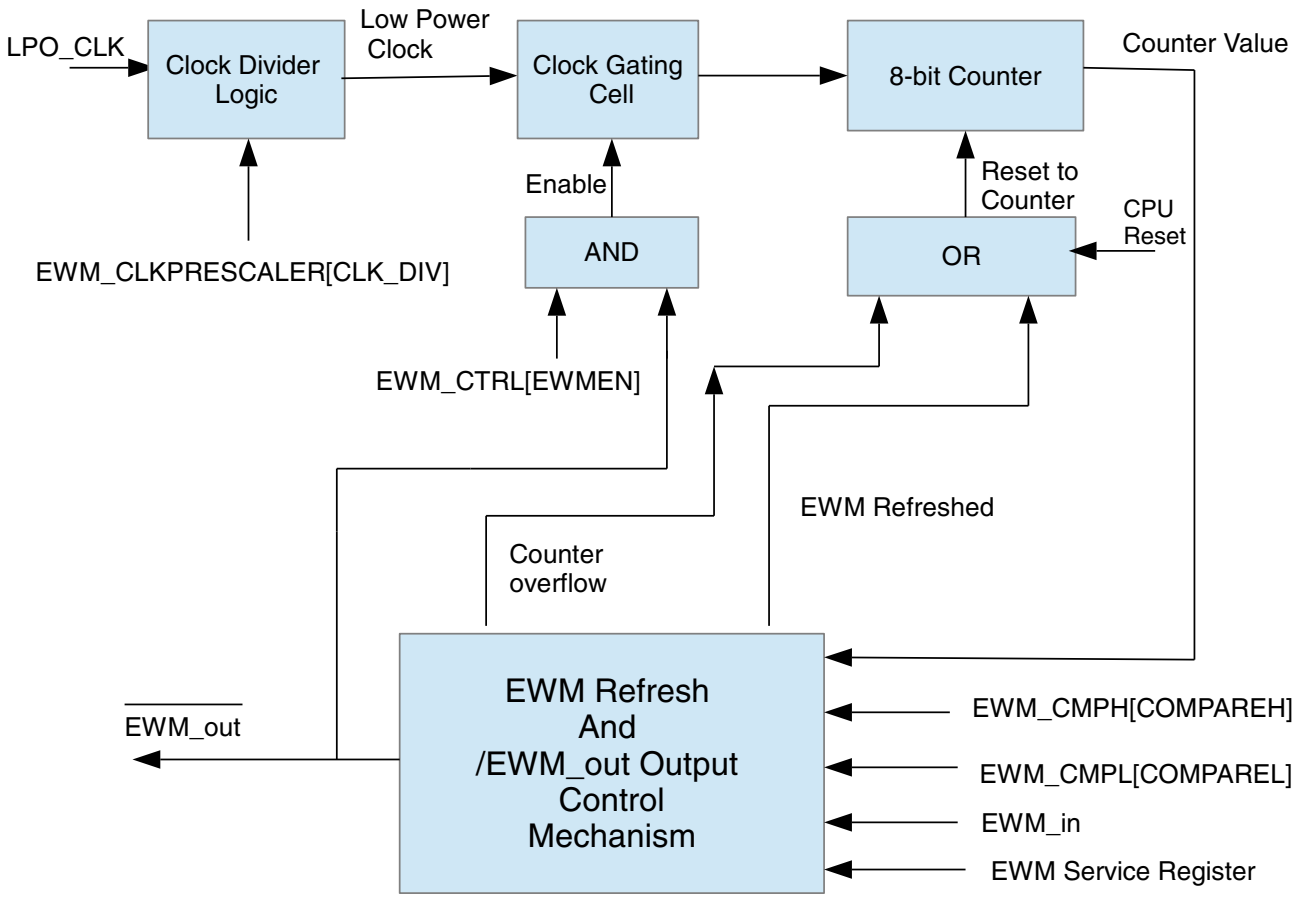



**Figure 42-1. EWM Block Diagram**

## 42.3 EWM Signal Descriptions

The EWM has two external signals, as shown in the following table.

**NOTE**

All active-low signals are now represented with the suffix "_b" throughout the chapter.

**Table 42-3. EWM Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| EWM_in | EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active low. | I |
| EWM_OUT_b | EWM reset out signal | O |

## 42.4 Memory Map/Register Definition

This section contains the module memory map and registers.

**NOTE**

EWM only supports 8-bit register access. 16-bit and 32-bit access are not possible.

### 42.4.1 EWM register descriptions

#### 42.4.1.1 EWM Memory map

EWM base address: 4002_2000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control Register (CTRL) | 8 | RWONCE | 00h |
| 1h | Service Register (SERV) | 8 | WORZ | 00h |
| 2h | Compare Low Register (CMPL) | 8 | RWONCE | 00h |
| 3h | Compare High Register (CMPH) | 8 | RWONCE | FFh |
| 5h | Clock Prescaler Register (CLKPRESCALER) | 8 | RWONCE | 00h |

#### 42.4.1.2 Control Register (CTRL)

##### 42.4.1.2.1 Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

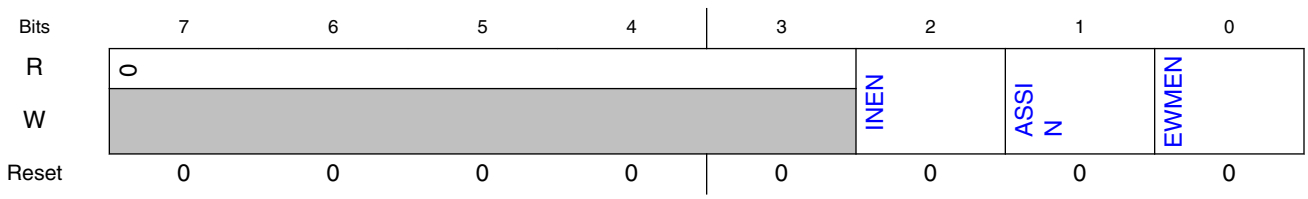
#### 42.4.1.2.2 Function

The CTRL register is cleared by any reset.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once, generates a bus transfer error.

#### 42.4.1.2.3 Diagram




#### 42.4.1.2.4 Fields

| Field | Function |
|---|---|
| 7-3<br>— | Reserved |
| 2<br>INEN | Input Enable.<br>This bit when set, enables the EWM_in port. |
| 1<br>ASSIN | EWM_in's Assertion State Select.<br>Default assert state of the EWM_in signal is logic zero. Setting the ASSIN bit inverts the assert state of EWM_in signal to a logic one. |
| 0<br>EWMEN | EWM enable.<br>This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the EWM_OUT_b signal. This bit when unset, keeps the EWM module disabled. It cannot be re-enabled until a next reset, due to the write-once nature of this bit. |

### 42.4.1.3 Service Register (SERV)

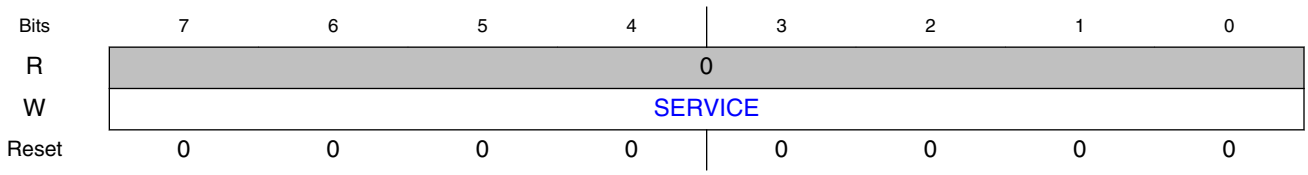
#### 42.4.1.3.1 Offset

| Register | Offset |
|---|---|
| SERV | 1h |

#### 42.4.1.3.2 Function

The SERV register provides the interface from the CPU to the EWM module. It is write-only and reads of this register return zero.

#### 42.4.1.3.3 Diagram




#### 42.4.1.3.4 Fields

| Field | Function |
|---|---|
| 7-0<br>SERVICE | SERVICE<br>The EWM refresh mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM refresh is invalid if either of the following conditions is true.<br>• The first or second data byte is not written correctly.<br>• The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte. This fixed number of cycles is called *EWM_refresh_time*.<br>15 peripheral bus clock cycles are required for *EWM_refresh_time* |

### 42.4.1.4 Compare Low Register (CMPL)

#### 42.4.1.4.1 Offset

| Register | Offset |
|---|---|
| CMPL | 2h |

#### 42.4.1.4.2 Function

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to refresh the EWM counter.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

#### 42.4.1.4.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPAREL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 42.4.1.4.4 Fields

| Field | Function |
|---|---|
| 7-0<br>COMPAREL | COMPAREL<br>To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum refresh time is required. |

### 42.4.1.5 Compare High Register (CMPH)

#### 42.4.1.5.1 Offset

| Register | Offset |
|---|---|
| CMPH | 3h |

#### 42.4.1.5.2 Function

The CMPH register is reset to 0xFF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to refresh the EWM counter.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

**NOTE**

The valid values for CMPH are up to 0xFE because the EWM counter never expires when CMPH = 0xFF. The expiration happens only if EWM counter is greater than CMPH.

#### 42.4.1.5.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPAREH | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 42.4.1.5.4 Fields

| Field | Function |
|---|---|
| 7-0<br>COMPAREH | COMPAREH<br>To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum refresh time is required. |

### 42.4.1.6 Clock Prescaler Register (CLKPRESCALER)

#### 42.4.1.6.1 Offset

| Register | Offset |
|---|---|
| CLKPRESCALER | 5h |

#### 42.4.1.6.2 Function

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

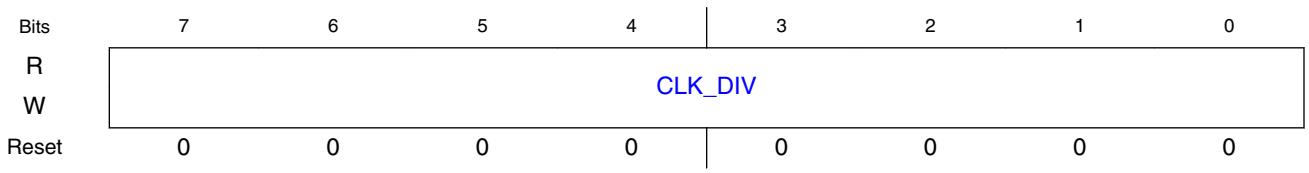
**NOTE**

Write the required prescaler value before enabling the EWM.

**NOTE**

The implementation of this register is chip-specific. See the Chip Configuration details.

#### 42.4.1.6.3 Diagram




#### 42.4.1.6.4 Fields

| Field | Function |
|---|---|
| 7-0<br>CLK_DIV | CLK_DIV<br>Selected low power clock source for running the EWM counter can be prescaled as below.<br>• Prescaled clock frequency = low power clock source frequency / ( 1 + CLK_DIV ) |

## 42.5 Functional Description

The following sections describe functional details of the EWM module.

**NOTE**

When the BUS_CLK is lost, then EWM module doesn't generate the EWM_OUT_b signal and no refresh operation is possible

### 42.5.1 The EWM_OUT_b Signal

The EWM_OUT_b is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions.

The EWM_OUT_b signal remains deasserted when the EWM is being regularly refreshed by the CPU within the programmable refresh window, indicating that the application code is executed as expected.

The EWM_OUT_b signal is asserted in any of the following conditions:

- The EWM refresh occurs when the counter value is less than CMPL value.
- The EWM counter value reaches the CMPH value, and no EWM refresh has occurred.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while refreshing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the EWM_OUT_b pin)

The EWM_OUT_b is asserted after any reset by the virtue of the external pull-down mechanism on the EWM_OUT_b signal. Then, to deassert the EWM_OUT_b signal, set EWMEN bit in the CTRL register to enable the EWM.

If the EWM_OUT_b signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. The pad state is controlled by the EWM_OUT_b signal only after the EWM is enabled by the EWMEN bit in the CTRL register.

**Note**

EWM_OUT_b pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

## 42.5.2 EWM_OUT_b pin state in low power modes

During Wait, Stop and Power Down modes the EWM_OUT_b pin enters a high-impedance state. A user has the option to control the logic state of the pin using an external pull device or by configuring the internal pull device. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode from Power Down, the pin returns to its reset state.

## 42.5.3 The EWM_in Signal

The EWM_in is a digital input signal for safety status of external safety circuits, that allows an external circuit to control the assertion of the EWM_OUT_b signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with safety function, the external circuit can then actively initiate the EWM_OUT_b signal that controls the gating circuit.

The EWM_in signal is ignored if the EWM is disabled, or if INEN bit of CTRL register is cleared, as after any reset.

On enabling the EWM (setting the CTRL[EWMEN] bit) and enabling EWM_in functionality (setting the CTRL[INEN] bit), the EWM_in signal must be in the deasserted state prior to the CPU start refreshing the EWM. This ensures that the EWM_OUT_b stays in the deasserted state; otherwise, the EWM_OUT_b output signal is asserted.

**Note**

The user must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore the user shall provide a reasonable time after a power-on reset for the external monitoring circuit to stabilize. The user shall also ensure that the EWM_in pin is deasserted.

### 42.5.4 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero after the CPU reset, or when EWM refresh action completes, or at counter overflow. The counter value is not accessible to the CPU.

### 42.5.5 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a refresh window to refresh the EWM module.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), EWM_OUT_b is asserted.

### 42.5.6 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers for correct EWM refresh operation. Therefore, three possible conditions can occur:

**Table 42-4. EWM Refresh Mechanisms**

| Condition | Mechanism |
|---|---|
| An EWM refresh action completes when:<br>CMPL < Counter < CMPH. | The software behaves as expected and the EWM counter is reset to zero. The EWM_OUT_b output signal remains in the deasserted state if, during the EWM refresh action, the EWM_in input has been in deasserted state.. |
| An EWM refresh action completes when Counter < CMPL | The software refreshes the EWM before the windowed time frame, the counter is reset to zero and the EWM_OUT_b output signal is asserted irrespective of the input EWM_in signal. |
| Counter value reaches CMPH prior to completion of EWM refresh action. | Software has not refreshed the EWM. The EWM counter is reset to zero and the EWM_OUT_b output signal is asserted irrespective of the input EWM_in signal. |

## 42.5.7 Counter clock prescaler

The EWM counter clock source can be prescaled by a clock divider, by programming CLKPRESCALER[CLK_DIV]. This divided clock is used to run the EWM counter.

**NOTE**

The divided clock used to run the EWM counter must be no more than half the frequency of the bus clock.

# Chapter 43
# Cyclic Redundancy Check (CRC)

## 43.1 Introduction

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

### 43.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register
- Programmable initial seed value and polynomial
- Option to transpose input data or output data (the CRC result) bitwise or bytewise. This option is required for certain CRC standards. A bytewise transpose operation is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user's software must perform the bytewise transpose function.
- Option for inversion of final CRC result
- 32-bit CPU register programming interface

### 43.1.2 Block diagram

The following is a block diagram of the CRC.

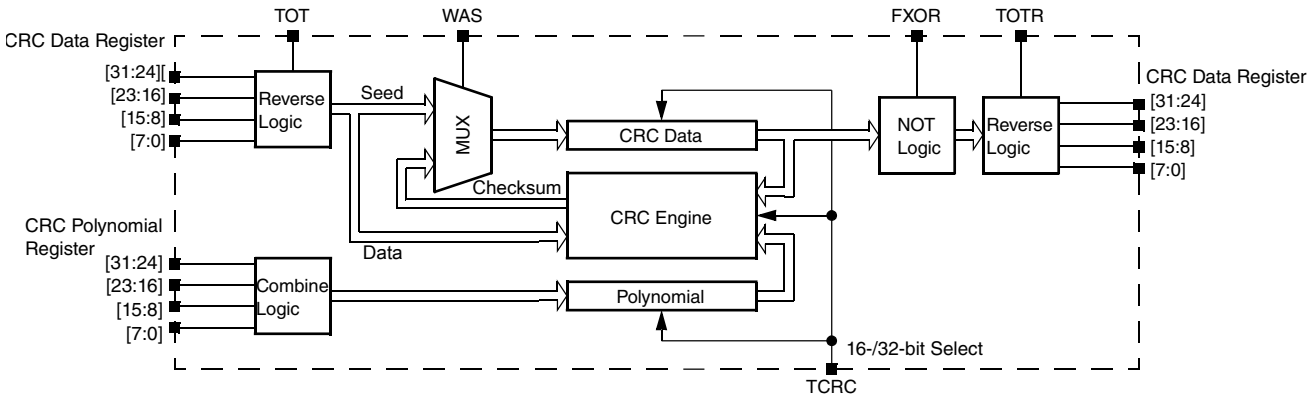



**Figure 43-1. Programmable cyclic redundancy check (CRC) block diagram**

## 43.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

### 43.1.3.1 Run mode

This is the basic mode of operation.

### 43.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is dependent on the MCU.

# 43.2 Memory map and register descriptions

## 43.2.1 CRC register descriptions

### 43.2.1.1 CRC Memory map

CRC base address: 4002_F000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | CRC Data register (DATA) | 32 | RW | FFFF_FFFFh |
| 4h | CRC Polynomial register (GPOLY) | 32 | RW | 0000_1021h |
| 8h | CRC Control register (CTRL) | 32 | RW | 0000_0000h |

## 43.2.1.2 CRC Data register (DATA)

### 43.2.1.2.1 Offset

| Register | Offset |
|---|---|
| DATA | 0h |

### 43.2.1.2.2 Function

The CRC Data register contains the value of the seed, data, and checksum. When CTRL[WAS] is set, any write to the data register is regarded as the seed value. When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

### 43.2.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HU | | | | | | | | HL | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LU | | | | | | | | LL | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 43.2.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>HU | CRC High Upper Byte<br>In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes. |
| 23-16<br>HL | CRC High Lower Byte<br>In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes. |
| 15-8<br>LU | CRC Low Upper Byte<br>When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation. |
| 7-0<br>LL | CRC Low Lower Byte<br>When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation. |

## 43.2.1.3 CRC Polynomial register (GPOLY)

### 43.2.1.3.1 Offset

| Register | Offset |
|---|---|
| GPOLY | 4h |

#### 43.2.1.3.2 Function

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

#### 43.2.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HIGH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LOW | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

#### 43.2.1.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>HIGH | High Polynominal Half-word<br>Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0). |
| 15-0<br>LOW | Low Polynominal Half-word<br>Writable and readable in both 32-bit and 16-bit CRC modes. |

### 43.2.1.4 CRC Control register (CTRL)

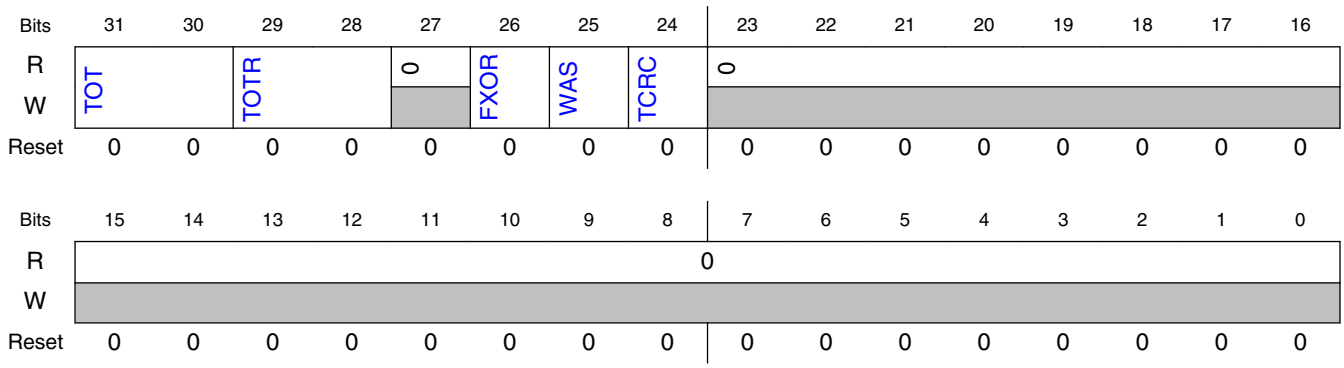
#### 43.2.1.4.1 Offset

| Register | Offset |
|---|---|
| CTRL | 8h |

### 43.2.1.4.2 Function

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by asserting CTRL[WAS] and then writing the seed into the CRC data register.

### 43.2.1.4.3 Diagram




### 43.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-30<br>TOT | Type Of Transpose For Writes<br>Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options.<br>00b - No transposition.<br>01b - Bits in bytes are transposed; bytes are not transposed.<br>10b - Both bits in bytes and bytes are transposed.<br>11b - Only bytes are transposed; no bits in a byte are transposed. |
| 29-28<br>TOTR | Type Of Transpose For Read<br>Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options.<br>00b - No transposition.<br>01b - Bits in bytes are transposed; bytes are not transposed.<br>10b - Both bits in bytes and bytes are transposed.<br>11b - Only bytes are transposed; no bits in a byte are transposed. |
| 27<br>— | Reserved |
| 26<br>FXOR | Complement Read Of CRC Data Register<br>Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data.<br>0b - No XOR on reading.<br>1b - Invert or complement the read value of the CRC Data register. |
| 25<br>WAS | Write CRC Data Register As Seed<br>When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Writes to the CRC data register are data values.<br>1b - Writes to the CRC data register are seed values. |
| 24<br>TCRC | TCRC<br>Width of CRC protocol.<br>0b - 16-bit CRC protocol.<br>1b - 32-bit CRC protocol. |
| 23-0<br>— | Reserved |

## 43.3 Functional description

### 43.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY,necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WAS] and programming a new, or previously used, seed value. All other parameters must be set before programming the seed value and subsequent data values.

### 43.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, data values can be programmed 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes can lead to an incorrect CRC computation.

#### 43.3.2.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.

3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[LU:LL]. CRC_DATA[HU:HL] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[LU:LL].

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

#### 43.3.2.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[HU:HL:LU:LL].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[HU:HL:LU:LL]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

### 43.3.3 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate a CRC. In this case, the transpose feature usefully flips the bits. This transpose option is one of the types supported by the CRC module.

### 43.3.3.1 Types of transpose

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CTRL[TOT] or CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CTRL[TOT] or CTRL[TOTR] is 00.

   No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 01

   Bits in a byte are transposed, while bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}

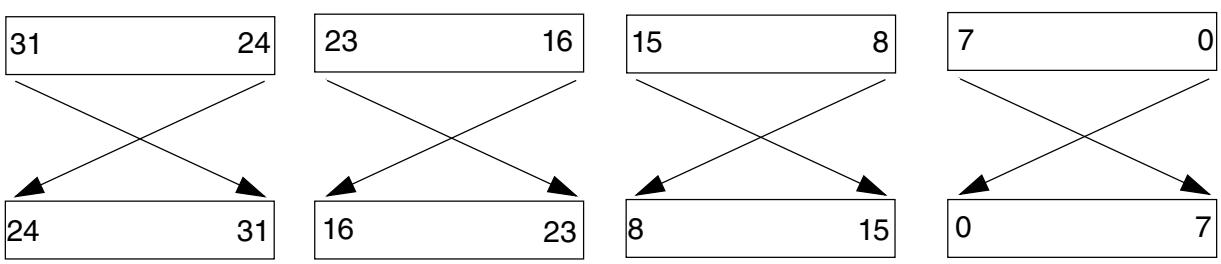



**Figure 43-2. Transpose type 01**

3. CTRL[TOT] or CTRL[TOTR] is 10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

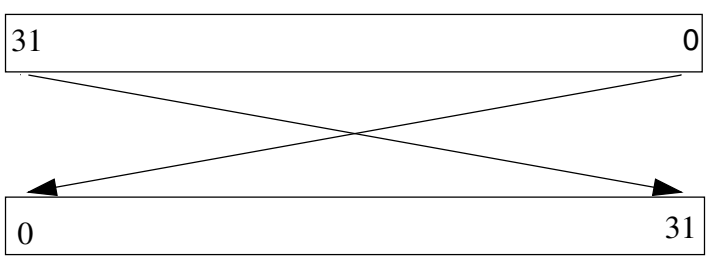



**Figure 43-3. Transpose type 10**

4. CTRL[TOT] or CTRL[TOTR] is 11.

Bytes are transposed, but bits are not transposed.

reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

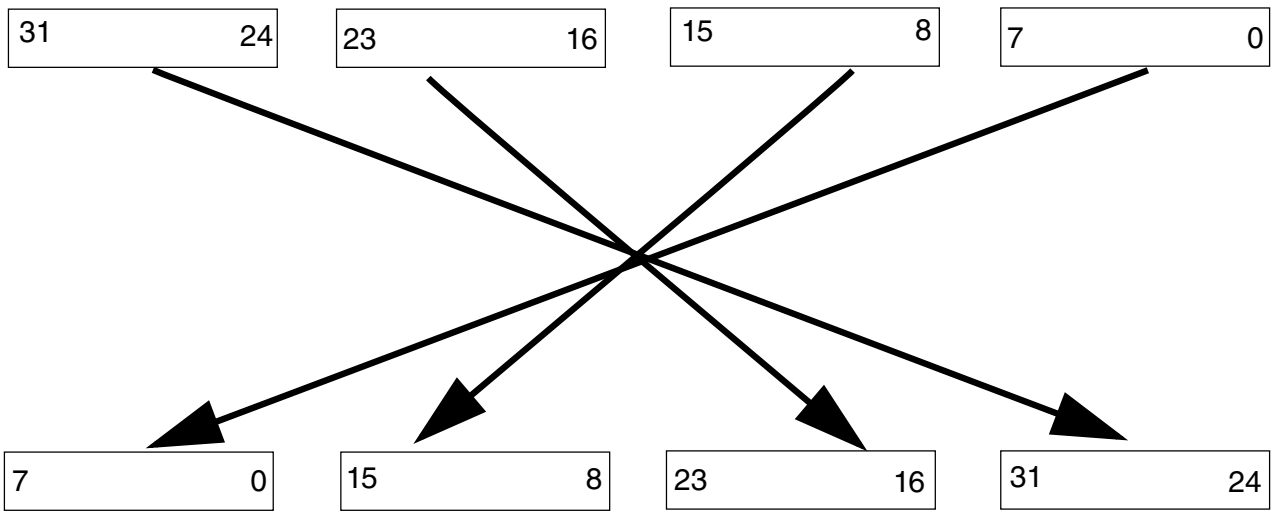



**Figure 43-4. Transpose type 11**

**NOTE**

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

## 43.3.4 CRC result complement

When CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

# Chapter 44
# Standalone TRNG

## 44.1 Standalone True Random Number Generator (SA-TRNG).

The Standalone True Random Number Generator (SA-TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy-consuming module or by other post-processing functions. A typical entropy consumer is a pseudo random-number generator (PRNG) that can be implemented to achieve both true randomness and cryptographic-strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module.

The entropy generated by a TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number be difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is particularly difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness/quality of the generated random data, PRNGs are often used to post process the output of a TRNG.

This document describes only the TRNG design functionality and usage.

Note that before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

The TRNG contains the following sub modules: IP Slave bus (SkyBlue bus) interface, the TRNG Core and the free running oscillator (OSC).

### 44.1.1 Standalone True Random Number Generator Block Diagram

The following figure is a top-level diagram of the True Random Number Generator.

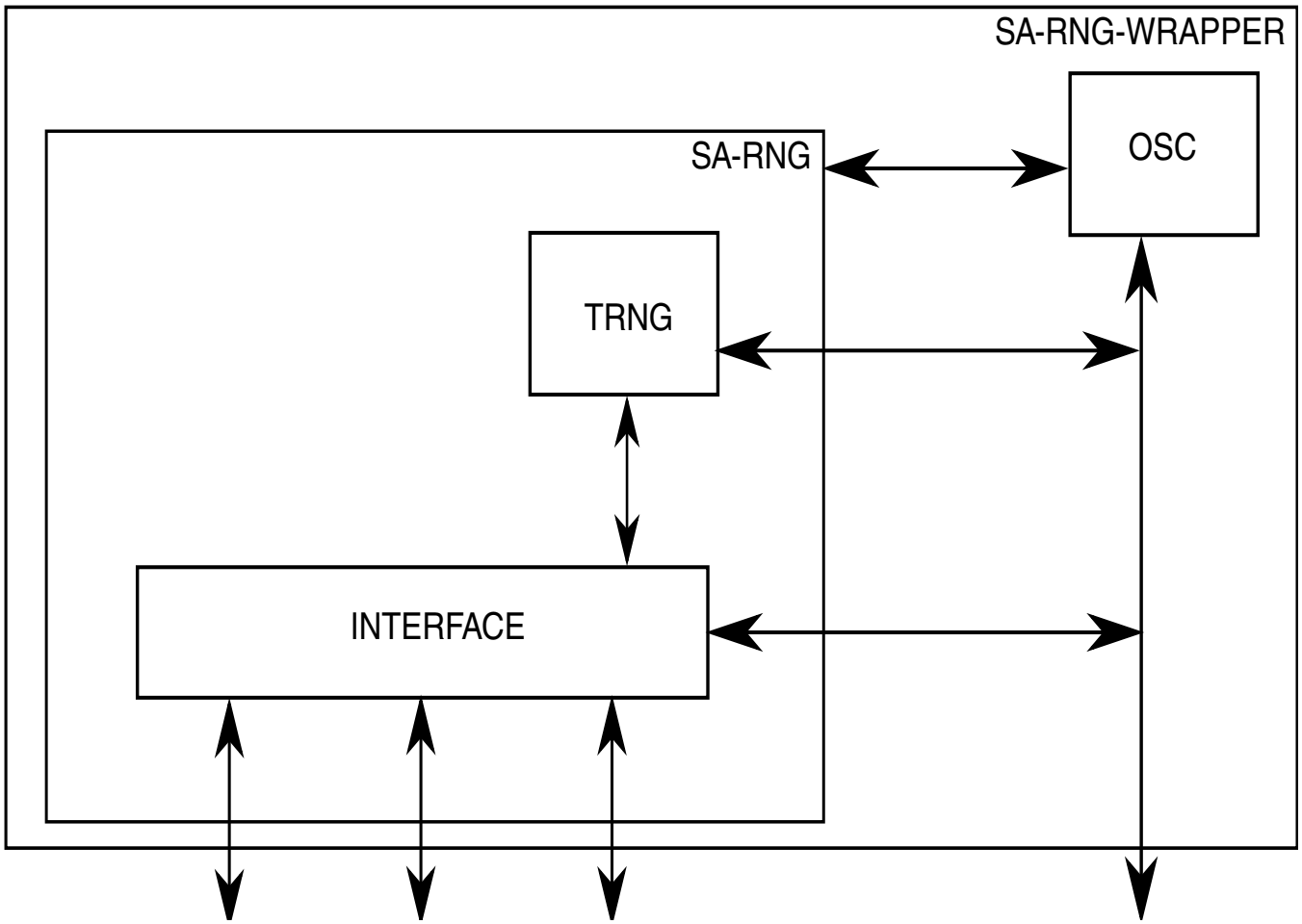



**Figure 44-1. SA TRNG Block Diagram**

## 44.1.2 TRNG Functional Description.

The TRNG consists of several functional sub-modules. Its overall functionality can be easily described from the top level in terms of generating entropy for seed generation. The functionality of each sub-module is briefly described in the following subsections.

TRNG is based on collecting bits from a random noise source. This random noise source is a ring oscillator that is sensitive to random noise (temperature variations, voltage variations, cross-talk and other random noise) within the device in which the TRNG is used. This noise causes various small changes in the period of the oscillator. Therefore, if the count of the ring oscillator clock cycles is sampled after a known period of time, this count will vary each time the sample is taken. By using the variance in this count over a large number of samples, random bits can be derived.

The TRNG comprises two entropy sources, each of which provides a single bit of output. Concatenated together, these 2 output bits are expected to provide 1 bit of entropy every 100 clock cycles. In addition to generating entropy, the TRNG also performs several statistical tests on its output.

## 44.1.3 SA-TRNG hardware functional description.

SA-TRNG functionality consists of several major subcomponents. This table describes these subcomponents.

**Table 44-1. SA-TRNG subcomponents.**

| Description | Cross-reference(s) |
|---|---|
| **Interfaces** | |
| Register interface<br>• Used for access to configuration, control, status and debugging registers | Register interface (IP Slave bus) |
| True Random Number Generator (TRNG) | Standalone True Random Number Generator (SA-TRNG). |

### 44.1.3.1 Software Use Cases for the Stand Alone TRNG.

There are four things that a user (programmer/integrator) will want to do with a TRNG.

- Initialization.

  Set up the parameters to proper values, and start generation of the first block of entropy. This is done once.

- Read entropy from the TRNG, and start generation of the next block of entropy.

  This is done many times and is the normal flow of operation.

- Run a self-test on the TRNG, to assure proper continued operation.

  This involves taking TRNG off-line, setting some self-test parameters, running TRNG, and then reading the statistical test registers, to see that they are within proper operation values. This may not be needed, as TRNG has built-in self-test.

- Off-line determination and checking of TRNG parameter values.

  This is done in development in order to determine the proper initialization and self-test parameters. The TRNG is taken off-line. Test parameter values are written and entropy generation is started. If the statistical tests indicate poor operation (i.e., failing statistical tests), the entropy_delay value should be increased and entropy

generation should be re-started. Every case is a variation of setting TRNG parameter values, starting or re-starting entropy generation and reading out the entropy. This process requires pausing or stopping and re-starting the TRNG.

The TRNG is designed to operate as a slave module on the standard IP Slave Bus. By understanding the TRNG register descriptions in "TRNG Register Descriptions" section below, the TRNG module can be controlled via the IP slave bus. In order to write to most TRNG registers, the MCTL register must be initialized in programming mode as described in the "TRNG Register Descriptions" section. At Power On Reset (POR) or asynchronous system hard reset, the TRNG resets to programming mode. TRNG will not generate entropy until it is out of programming mode (in run mode) and access to Entropy Registers have been enabled.

Here is an example program flow of using the TRNG.

- After POR the TRNG will be reset into programming mode with the OK to stop bit set(MCTL[TSTOP_OK]=1). The TRNG must be put into Run Mode for Entropy Generation to begin (MCTL[PRGM]=0). Additionally, in order to have access to the Entropy registers and other critical TRNG registers, the TRNG access bit must be set (MCTL[TRNG_ACC]=1). Using the default self test limits that exist after bootup, the entropy valid bit can be polled until asserted (MCTL[ENT_VAL]=1). Alternatively, if using the interrupt, and the interrupts are enabled via the INT_MASK register and the ipi_rng_int_b is asserted when MCTL[ENT_VAL]=1.
- After the polling completes, the 512-bit entropy generated by the TRNG can be read. The values can be read in any order from entropy register 0 to register 15 (ENT0 to ENT 15). After reading ENT 15, the old entropy value is reset and a new entropy value is generated.

**NOTE**

Reading ENT 15 always resets the entropy, so should always be read last.

- You can poll again for the new entropy value or you can use the Interrupt Status Register to handle reading the entropy values when the entropy valid interrupt is triggered.
- The interrupt can be masked or cleared as needed. See the Interrupt Status Register description.
- To change the self-test limits,the seed counters, how fast the entropy is generated, and how entropy is sampled, see the register description section. In particular, see the the TRNG Frequency Count Minimum Limit Register (FRQMIN), the seed control register(SCML), the statistical run length registers, and other parameter registers.
- Once in Run Mode, the entropy is re-generated automatically after ENT 15 is read. To stop the TRNG or access to TRNG registers at any point while in running mode,

you can always set MCTL[TRNG_ACC]=0. Setting the TRNG back to programming mode (MCTL[PRGM]=1) also achieves the purpose of stopping entropy generation.

### 44.1.4 Another TRNG usage example.

The TRNG can be used by a post processing pseudo-random number generator function. For example, TRNG can be used to seed a hardware or software based implementation of a DRBG defined by SP800-90.

### 44.1.5 TRNG register descriptions

All accesses of undefined addresses always return zero and assert IPS transfer error. Writes to undefined and read-only addresses are ignored. Undefined addresses are those undocumented, protected or reserved addresses within and outside the range of the addresses defined in the memory map below. The register addresses shown in the Memory Map below represents how these registers are accessed over the register bus as 32-bit words. Accesses to registers must use full-word (32-bit) reads or writes.

The format and fields in each TRNG register are defined below. Some of the register format figures apply to several different registers. In such cases a different register name will be associated with each of the register offset addresses that appear at the top of the register format figure. Although these registers share the same format, they are independent registers. In addition, many registers can be accessed at multiple addresses. In these cases there will be a single register name and the list of addresses at which that register is accessible will be indicated as aliases. Unless noted in the individual register descriptions, registers are reset only at Power-On Reset (POR) or with an asyncronous system hard reset.

#### 44.1.5.1 TRNG Memory map

TRNG base address: 4102_9000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Miscellaneous Control Register (MCTL) | 32 | RW | 0001_2001h |
| 4h | Statistical Check Miscellaneous Register (SCMISC) | 32 | RW | 0001_0022h |
| 8h | Poker Range Register (PKRRNG) | 32 | RW | 0000_09A3h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| Ch | Poker Maximum Limit Register (PKRMAX) | 32 | RW | 0000_6920h |
| Ch | Poker Square Calculation Result Register (PKRSQ) | 32 | RO | 0000_0000h |
| 10h | Seed Control Register (SDCTL) | 32 | RW | 0C80_09C4h |
| 14h | Sparse Bit Limit Register (SBLIM) | 32 | RW | 0000_003Fh |
| 14h | Total Samples Register (TOTSAM) | 32 | RO | 0000_0000h |
| 18h | Frequency Count Minimum Limit Register (FRQMIN) | 32 | RW | 0000_0640h |
| 1Ch | Frequency Count Register (FRQCNT) | 32 | RO | 0000_0000h |
| 1Ch | Frequency Count Maximum Limit Register (FRQMAX) | 32 | RW | 0000_6400h |
| 20h | Statistical Check Monobit Count Register (SCMC) | 32 | RO | 0000_0000h |
| 20h | Statistical Check Monobit Limit Register (SCML) | 32 | RW | 010C_0568h |
| 24h | Statistical Check Run Length 1 Count Register (SCR1C) | 32 | RO | 0000_0000h |
| 24h | Statistical Check Run Length 1 Limit Register (SCR1L) | 32 | RW | 00B2_0195h |
| 28h | Statistical Check Run Length 2 Count Register (SCR2C) | 32 | RO | 0000_0000h |
| 28h | Statistical Check Run Length 2 Limit Register (SCR2L) | 32 | RW | 007A_00DCh |
| 2Ch | Statistical Check Run Length 3 Count Register (SCR3C) | 32 | RO | 0000_0000h |
| 2Ch | Statistical Check Run Length 3 Limit Register (SCR3L) | 32 | RW | 0058_007Dh |
| 30h | Statistical Check Run Length 4 Count Register (SCR4C) | 32 | RO | 0000_0000h |
| 30h | Statistical Check Run Length 4 Limit Register (SCR4L) | 32 | RW | 0040_004Bh |
| 34h | Statistical Check Run Length 5 Count Register (SCR5C) | 32 | RO | 0000_0000h |
| 34h | Statistical Check Run Length 5 Limit Register (SCR5L) | 32 | RW | 002E_002Fh |
| 38h | Statistical Check Run Length 6+ Count Register (SCR6PC) | 32 | RO | 0000_0000h |
| 38h | Statistical Check Run Length 6+ Limit Register (SCR6PL) | 32 | RW | 002E_002Fh |
| 3Ch | Status Register (STATUS) | 32 | RO | 0000_0000h |
| 40h - 7Ch | Entropy Read Register (ENT0 - ENT15) | 32 | RO | 0000_0000h |
| 80h | Statistical Check Poker Count 1 and 0 Register (PKRCNT10) | 32 | RO | 0000_0000h |
| 84h | Statistical Check Poker Count 3 and 2 Register (PKRCNT32) | 32 | RO | 0000_0000h |
| 88h | Statistical Check Poker Count 5 and 4 Register (PKRCNT54) | 32 | RO | 0000_0000h |
| 8Ch | Statistical Check Poker Count 7 and 6 Register (PKRCNT76) | 32 | RO | 0000_0000h |
| 90h | Statistical Check Poker Count 9 and 8 Register (PKRCNT98) | 32 | RO | 0000_0000h |
| 94h | Statistical Check Poker Count B and A Register (PKRCNTBA) | 32 | RO | 0000_0000h |
| 98h | Statistical Check Poker Count D and C Register (PKRCNTDC) | 32 | RO | 0000_0000h |
| 9Ch | Statistical Check Poker Count F and E Register (PKRCNTFE) | 32 | RO | 0000_0000h |
| A0h | Security Configuration Register (SEC_CFG) | 32 | RW | 0000_0000h |
| A4h | Interrupt Control Register (INT_CTRL) | 32 | RW | FFFF_FFFFh |
| A8h | Mask Register (INT_MASK) | 32 | RW | 0000_0000h |
| ACh | Interrupt Status Register (INT_STATUS) | 32 | RO | 0000_0000h |
| F0h | Version ID Register (MS) (VID1) | 32 | RO | 0030_0100h |
| F4h | Version ID Register (LS) (VID2) | 32 | RO | 0000_0000h |

## 44.1.5.2 Miscellaneous Control Register (MCTL)

### 44.1.5.2.1 Offset

| Register | Offset |
|---|---|
| MCTL | 0h |

### 44.1.5.2.2 Function

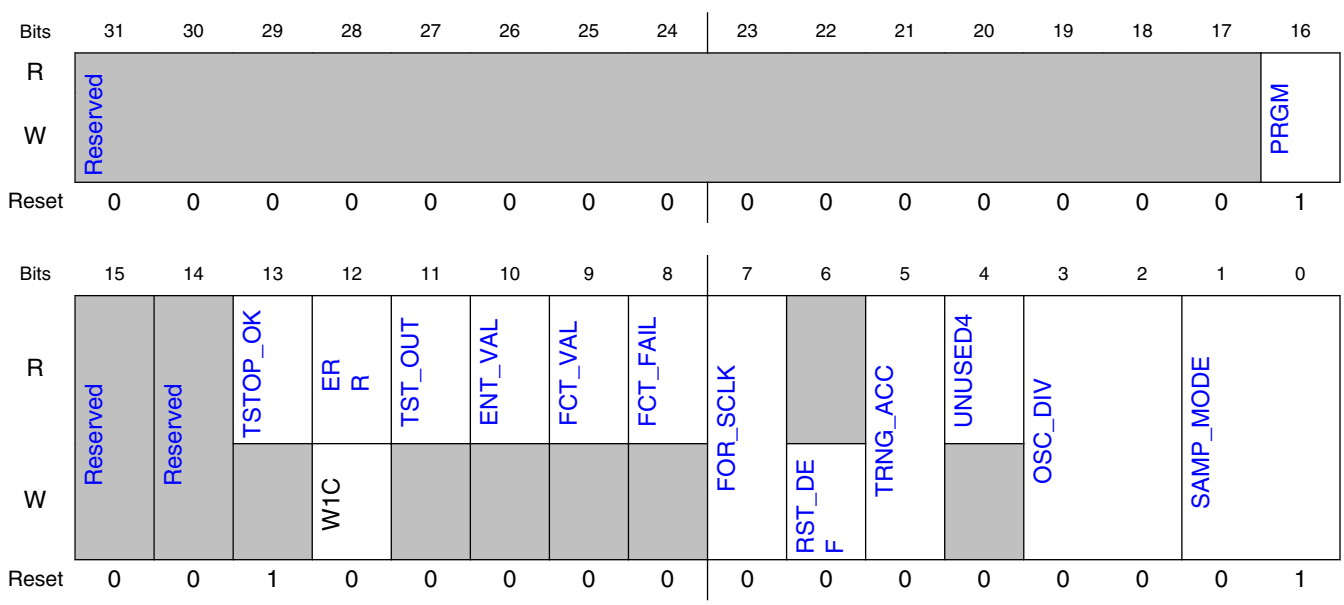
This register is intended to be used for programming, configuring and testing the RNG. It is the main register to read/write, in order to enable Entropy generation, to stop entropy generation and to block access to entropy registers. This is done via the special TRNG_ACC and PRGM bit below.

The Miscellaneous Control Register is a read/write register used to control the RNG's True Random Number Generator (TRNG) access, operation and test.

**NOTE**

Note that in many cases two RNG registers share the same address, and a particular register at the shared address is selected based upon the value in the PRGM field of the MCTL register.

### 44.1.5.2.3 Diagram



### 44.1.5.2.4 Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16<br>PRGM | Programming Mode Select. When this bit is 1, the TRNG is in Program Mode, otherwise it is in Run Mode. No Entropy value will be generated while the TRNG is in Program Mode. Note that different RNG registers are accessible at the same address depending on whether PRGM is set to 1 or 0. This is noted in the RNG register descriptions. |
| 15<br>— | Reserved. |
| 14<br>— | Reserved. |
| 13<br>TSTOP_OK | TRNG_OK_TO_STOP. Software can check that this bit is a 1 before transitioning TRNG to low power mode (TRNG clock stopped). TRNG turns on the TRNG free-running ring oscillator whenever new entropy is being generated and turns off the ring oscillator when entropy generation is complete. If the TRNG clock is stopped while the TRNG ring oscillator is running, the oscillator will continue running even though the TRNG clock is stopped. To make sure the ring oscillator is stopped, assert ipg_stop input and verify ipg_rng_stop_ack is asserted after at at least two (2) clock cycles. TSTOP_OK is asserted when the TRNG ring oscillator is not running. |
| 12<br>ERR | Read: Error status. 1 = error detected. 0 = no error.<br>Write: Write 1 to clear errors. Writing 0 has no effect. |
| 11<br>TST_OUT | Read only: Test point inside ring oscillator. |
| 10<br>ENT_VAL | Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared at most one (1) bus clock cycle after reading the ENT15 register. (ENT0 through ENT14 should be read before reading ENT15). |
| 9<br>FCT_VAL | Read only: Frequency Count Valid. Indicates that a valid frequency count may be read from FRQCNT. |
| 8<br>FCT_FAIL | Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. This error may be cleared by writing a 1 to the ERR bit. |
| 7<br>FOR_SCLK | Force System Clock. If set, the system clock is used to operate the TRNG, instead of the ring oscillator. This is for test use only, and indeterminate results may occur. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1. |
| 6<br>RST_DEF | Reset Defaults. Writing a 1 to this bit clears various TRNG registers, and bits within registers, to their default state. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. Reading this bit always produces a 0. |
| 5<br>TRNG_ACC | TRNG Access Mode. If this bit is set to 1, the TRNG will generate an Entropy value that can be read via the ENT0-ENT15 registers. The Entropy value may be read once the ENT VAL bit is asserted. Also see ENTa register descriptions (For a = 0 to 15). |
| 4<br>UNUSED4 | This bit is unused. Always reads zero. |
| 3-2 | Oscillator Divide. Determines the amount of dividing done to the ring oscillator before it is used by the TRNG. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| OSC_DIV | This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1.<br>00b - use ring oscillator with no divide<br>01b - use ring oscillator divided-by-2<br>10b - use ring oscillator divided-by-4<br>11b - use ring oscillator divided-by-8 |
| 1-0<br>SAMP_MODE | Sample Mode. Determines the method of sampling the ring oscillator while generating the Entropy value:<br>This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously with writing this field. This field is cleared to the POR default value by writing the RST_DEF bit to 1.<br>00b - use Von Neumann data into both Entropy shifter and Statistical Checker<br>01b - use raw data into both Entropy shifter and Statistical Checker<br>10b - use Von Neumann data into Entropy shifter. Use raw data into Statistical Checker<br>11b - undefined/reserved. |

## 44.1.5.3 Statistical Check Miscellaneous Register (SCMISC)

### 44.1.5.3.1 Offset

| Register | Offset |
|---|---|
| SCMISC | 4h |

### 44.1.5.3.2 Function

The Statistical Check Miscellaneous Register contains the Long Run Maximum Limit value and the Retry Count value. This register is accessible only when the MCTL[PRGM] bit is 1, otherwise this register will read zeroes, and cannot be written.

### 44.1.5.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | RTY_CT | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | LRUN_MAX | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

#### 44.1.5.3.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19-16<br>RTY_CT | RETRY COUNT. If a statistical check fails during the TRNG Entropy Generation, the RTY_CT value indicates the number of times a retry should occur before generating an error. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-8<br>— | Reserved. |
| 7-0<br>LRUN_MAX | LONG RUN MAX LIMIT. This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the MCTL[RST_DEF] bit to 1. |

### 44.1.5.4 Poker Range Register (PKRRNG)

#### 44.1.5.4.1 Offset

| Register | Offset |
|---|---|
| PKRRNG | 8h |

#### 44.1.5.4.2 Function

The Poker Range Register defines the difference between the TRNG Poker Maximum Limit and the minimum limit. These limits are used during the TRNG Statistical Check Poker Test.

#### 44.1.5.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKR_RNG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

#### 44.1.5.4.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. Always 0. |
| 15-0<br>PKR_RNG | Poker Range. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register, and the minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the minimum allowable Poker result is PKR_MAX - PKR_RNG + 1. |

### 44.1.5.5 Poker Square Calculation Result Register (PKRSQ)

#### 44.1.5.5.1 Offset

| Register | Offset | Description |
|---|---|---|
| PKRSQ | Ch | Accessible at this address when MCTL[PRGM] = 0] |

#### 44.1.5.5.2 Function

The Poker Square Calculation Result Register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed PKRRNG value. Note that this offset (0x0C) is used as PKRMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as PKRSQ readback register, as described here.

#### 44.1.5.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PKR_SQ | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_SQ | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 44.1.5.5.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>PKR_SQ | Poker Square Calculation Result.<br>During the TRNG Statistical Checks, a "Poker Test" is run which starts with the value PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm, and must remain greater than zero, but less than the PKRRNG[PKR_RNG] limit. The resulting value may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register PKRMAX, as defined in the previous section. |

### 44.1.5.6 Poker Maximum Limit Register (PKRMAX)

#### 44.1.5.6.1 Offset

| Register | Offset | Description |
|---|---|---|
| PKRMAX | Ch | Accessible at this address when MCTL[PRGM] = 1] |

#### 44.1.5.6.2 Function

The Poker Maximum Limit Register defines Maximum Limit allowable during the TRNG Statistical Check Poker Test. Note that this offset (0x0C) is used as PKRMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as the PKRSQ readback register.

#### 44.1.5.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PKR_MAX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

#### 44.1.5.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>PKR_MAX | Poker Maximum Limit.<br>During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the PKRMAX[PKR_MAX] register. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the PKRMAX and PKRRNG registers combined are used to define the minimum allowable Poker result, which is PKR_MAX - PKR_RNG + 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Poker Test Square Calculation result in register PKRSQ, as defined in the following section. |

### 44.1.5.7 Seed Control Register (SDCTL)

#### 44.1.5.7.1 Offset

| Register | Offset |
|---|---|
| SDCTL | 10h |

#### 44.1.5.7.2 Function

The Seed Control Register contains two fields. One field defines the length (in system clocks) of each Entropy sample (ENT_DLY), and the other field indicates the number of samples that will taken during each TRNG Entropy generation (SAMP_SIZE).

### 44.1.5.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ENT_DLY | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAMP_SIZE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

### 44.1.5.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>ENT_DLY | Entropy Delay. Defines the length (in system clocks) of each Entropy sample taken. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR. |
| 15-0<br>SAMP_SIZE | Sample Size. Defines the total number of Entropy samples that will be taken during Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.8 Sparse Bit Limit Register (SBLIM)

### 44.1.5.8.1 Offset

| Register | Offset | Description |
|---|---|---|
| SBLIM | 14h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.8.2 Function

The Sparse Bit Limit Register is used when Von Neumann sampling is selected during Entropy Generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated. Note that this address (0x14) is used as SBLIM only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as TOTSAM readback register.

#### 44.1.5.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | SB_LIM | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 44.1.5.8.4 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. Always 0. |
| 9-0<br>SB_LIM | Sparse Bit Limit. During Von Neumann sampling (if enabled by MCTL[SAMP_MODE], samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient Entropy. The Sparse Bit Limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Total Samples count in register TOTSAM, as defined in the following section. |

### 44.1.5.9 Total Samples Register (TOTSAM)

#### 44.1.5.9.1 Offset

| Register | Offset | Description |
|---|---|---|
| TOTSAM | 14h | Accessible at this address when MCTL[PRGM] = 0] |

#### 44.1.5.9.2 Function

The Total Samples Register is a read-only register used to read the total number of samples taken during Entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling. Note that this offset (0x14) is used as SBLIM if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as TOTSAM readback register, as described here.

### 44.1.5.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | TOT_SAM | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOT_SAM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.9.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. Always 0. |
| 19-0<br>TOT_SAM | Total Samples. During Entropy generation, the total number of raw samples is counted. This count is useful in determining how often a sample is used during Von Neumann sampling. The count may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Sparse Bit Limit in register SBLIM, as defined in the previous section. |

## 44.1.5.10 Frequency Count Minimum Limit Register (FRQMIN)

### 44.1.5.10.1 Offset

| Register | Offset |
|---|---|
| FRQMIN | 18h |

### 44.1.5.10.2 Function

The Frequency Count Minimum Limit Register defines the minimum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is less than this programmed minimum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated.

### 44.1.5.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | FRQ_MIN | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FRQ_MIN | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.10.4 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved. Always 0. |
| 21-0<br>FRQ_MIN | Frequency Count Minimum Limit. Defines the minimum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR. |

## 44.1.5.11 Frequency Count Maximum Limit Register (FRQMAX)

### 44.1.5.11.1 Offset

| Register | Offset | Description |
|---|---|---|
| FRQMAX | 1Ch | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.11.2 Function

The Frequency Count Maximum Limit Register defines the maximum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is greater than this programmed maximum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated. Note that this address (001C) is used as FRQMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as FRQCNT readback register.

### 44.1.5.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | FRQ_MAX | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FRQ_MAX | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.11.4 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved. Always 0. |
| 21-0<br>FRQ_MAX | Frequency Counter Maximum Limit. Defines the maximum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Frequency Count result in register FRQCNT, as defined in the FRQ_CT field. |

## 44.1.5.12 Frequency Count Register (FRQCNT)

### 44.1.5.12.1 Offset

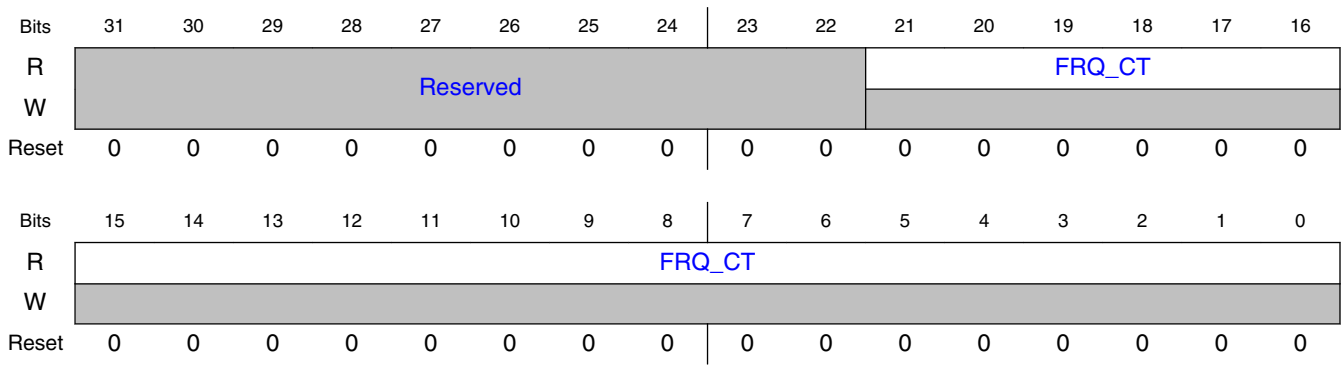
| Register | Offset | Description |
|---|---|---|
| FRQCNT | 1Ch | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.12.2 Function

The Frequency Count Register is a read-only register used to read the frequency counter within the TRNG entropy generator. It will read all zeroes unless PRGM[MCTL] = 0 and MCTL[TRNG_ACC] = 1. Note that this offset (0x1C) is used as FRQMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as live FRQCNT readback register, as described here. This register must be read once, as a full 32-bit. A subsequent

read is only valid when MCTL[FCT_VAL] bit is HIGH. A read clears this register. The next read when MCTL[FCT_VAL] bit is HIGH must be treated as a different value from the previous one, even though the counts might be the same.

### 44.1.5.12.3 Diagram




### 44.1.5.12.4 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved. Always 0. |
| 21-0<br>FRQ_CT | Frequency Count. If MCTL[TRNG_ACC] = 1, Reads a sample frequency count taken during entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Frequency Maximum Limit in register FRQMAX, as defined in the FRQ_MAX field. |

## 44.1.5.13 Statistical Check Monobit Count Register (SCMC)

### 44.1.5.13.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCMC | 20h | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.13.2 Function

The Statistical Check Monobit Count Register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in SCML[MONO_MAX], and is decremented each time a one is sampled. Note that this offset (0x20) is used as SCML if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register, as described here.

### 44.1.5.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MONO_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.13.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. Always 0. |
| 15-0<br>MONO_CT | Monobit Count. Reads the final Monobit count after entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Statistical Check Monobit Limit in register SCML, as defined in the previous section. |

## 44.1.5.14 Statistical Check Monobit Limit Register (SCML)

### 44.1.5.14.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCML | 20h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.14.2 Function

The Statistical Check Monobit Limit Register defines the allowable maximum and minimum number of ones/zero detected during entropy generation. To pass the test, the number of ones/zeroes generated must be less than the programmed maximum value, and the number of ones/zeroes generated must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x20) is used as SCML only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register.

### 44.1.5.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MONO_RNG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MONO_MAX | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

### 44.1.5.14.4 Fields

| Field | Function |
|---|---|
| 31-16<br>MONO_RNG | Monobit Range. The number of ones/zeroes detected during entropy generation must be greater than MONO_MAX - MONO_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-0<br>MONO_MAX | Monobit Maximum Limit. Defines the maximum allowable count taken during entropy generation. The number of ones/zeroes detected during entropy generation must be less than MONO_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.15 Statistical Check Run Length 1 Limit Register (SCR1L)

### 44.1.5.15.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR1L | 24h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.15.2 Function

The Statistical Check Run Length 1 Limit Register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 1 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x24) is used as SCR1L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR1C readback register.

### 44.1.5.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | RUN1_RNG | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | RUN1_MAX | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

### 44.1.5.15.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. Always 0. |
| 30-16<br>RUN1_RNG | Run Length 1 Range. The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX - RUN1_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15<br>— | Reserved. Always 0. |
| 14-0<br>RUN1_MAX | Run Length 1 Maximum Limit. Defines the maximum allowable runs of length 1 (for both 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.16 Statistical Check Run Length 1 Count Register (SCR1C)

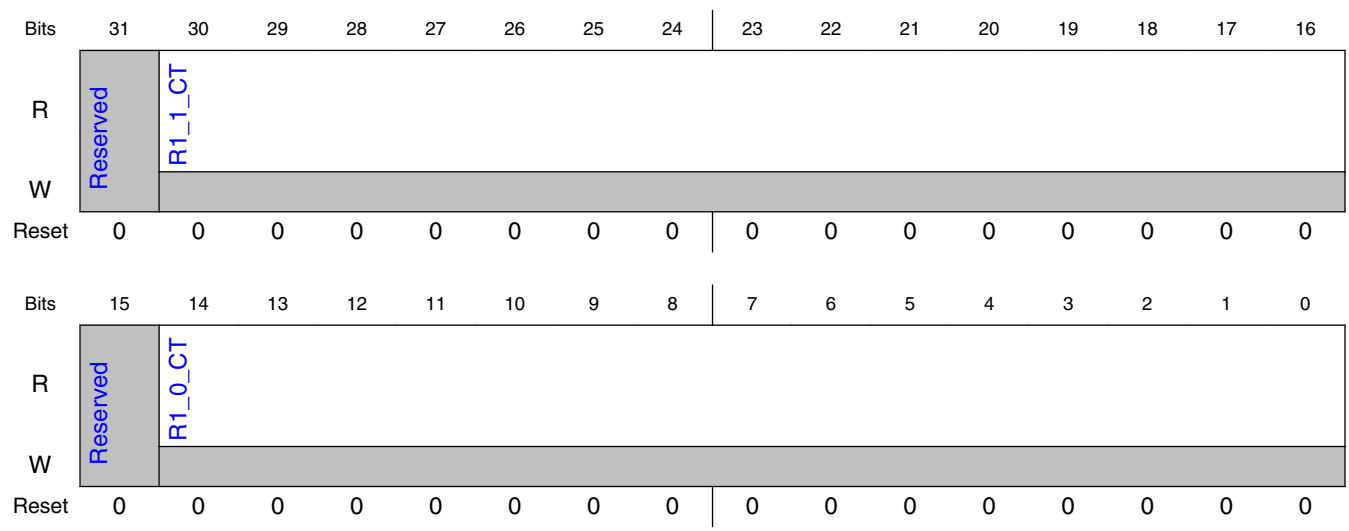
### 44.1.5.16.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR1C | 24h | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.16.2 Function

The Statistical Check Run Length 1 Counters Register is a read-only register used to read the final Run Length 1 counts after entropy generation. These counters start with the value in SCR1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one). Note that this offset (0x24) is used as SCR1L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR1C readback register, as described here.

### 44.1.5.16.3 Diagram




### 44.1.5.16.4 Fields

| Field | Function |
|---|---|
| 31 | Reserved. Always 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 30-16<br>R1_1_CT | Runs of One, Length 1 Count. Reads the final Runs of Ones, length 1 count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15<br>— | Reserved. Always 0. |
| 14-0<br>R1_0_CT | Runs of Zero, Length 1 Count. Reads the final Runs of Zeroes, length 1 count after entropy generation. Requires MCTL[PRGM] = 0. |

## 44.1.5.17 Statistical Check Run Length 2 Count Register (SCR2C)

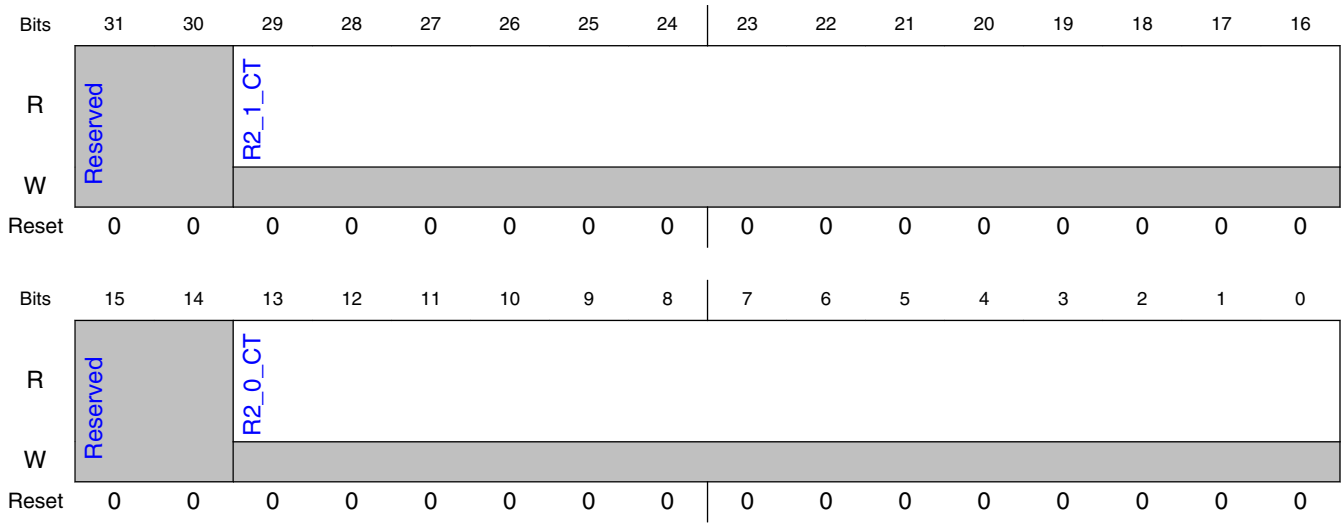
### 44.1.5.17.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR2C | 28h | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.17.2 Function

The Statistical Check Run Length 2 Counters Register is a read-only register used to read the final Run Length 2 counts after entropy generation. These counters start with the value in SCR2L[RUN2_MAX]. The R2_1_CT decrements each time two consecutive ones are sampled (preceded by a zero and followed by a zero). The R2_0_CT decrements each time two consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x28) is used as SCR2L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR2C readback register, as described here.

### 44.1.5.17.3 Diagram




### 44.1.5.17.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. Always 0. |
| 29-16<br>R2_1_CT | Runs of One, Length 2 Count. Reads the final Runs of Ones, length 2 count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15-14<br>— | Reserved. Always 0. |
| 13-0<br>R2_0_CT | Runs of Zero, Length 2 Count. Reads the final Runs of Zeroes, length 2 count after entropy generation. Requires MCTL[PRGM] = 0. |

## 44.1.5.18 Statistical Check Run Length 2 Limit Register (SCR2L)

### 44.1.5.18.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR2L | 28h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.18.2 Function

The Statistical Check Run Length 2 Limit Register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 2 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x28) is used as SCR2L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR2C readback register.

### 44.1.5.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | RUN2_RNG | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | RUN2_MAX | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

### 44.1.5.18.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. Always 0. |
| 29-16<br>RUN2_RNG | Run Length 2 Range. The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX - RUN2_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-14<br>— | Reserved. Always 0. |
| 13-0<br>RUN2_MAX | Run Length 2 Maximum Limit. Defines the maximum allowable runs of length 2 (for both 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.19 Statistical Check Run Length 3 Limit Register (SCR3L)

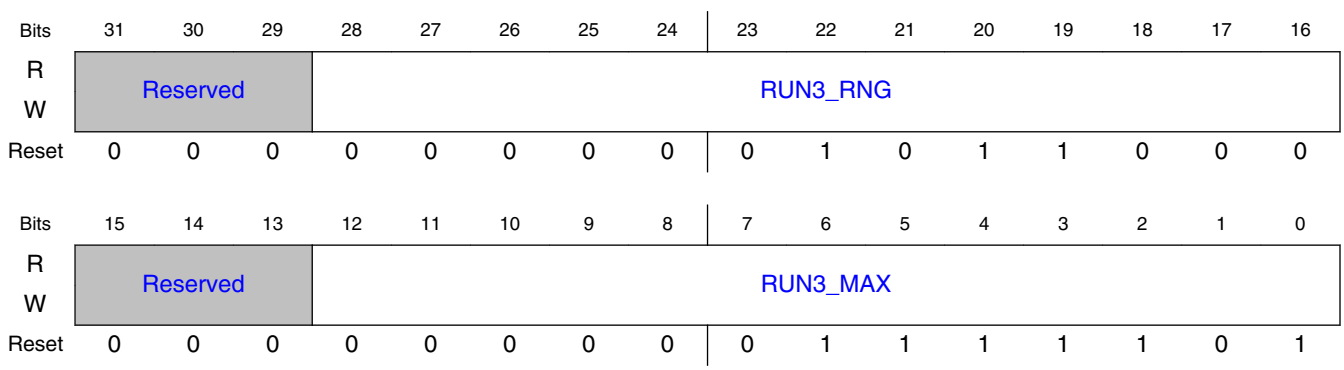
### 44.1.5.19.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR3L | 2Ch | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.19.2 Function

The Statistical Check Run Length 3 Limit Register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 3 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x2C) is used as SCR3L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR3C readback register.

### 44.1.5.19.3 Diagram




### 44.1.5.19.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. Always 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 28-16<br>RUN3_RNG | Run Length 3 Range. The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX - RUN3_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-13<br>— | Reserved. Always 0. |
| 12-0<br>RUN3_MAX | Run Length 3 Maximum Limit. Defines the maximum allowable runs of length 3 (for both 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.20 Statistical Check Run Length 3 Count Register (SCR3C)

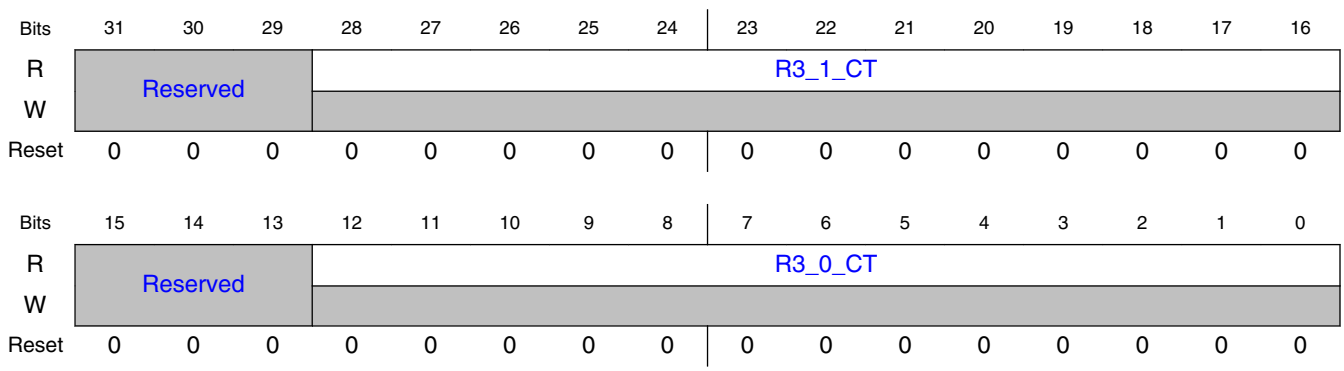
### 44.1.5.20.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR3C | 2Ch | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.20.2 Function

The Statistical Check Run Length 3 Counters Register is a read-only register used to read the final Run Length 3 counts after entropy generation. These counters start with the value in SCR3L[RUN3_MAX]. The R3_1_CT decrements each time three consecutive ones are sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time three consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x2C) is used as SCR3L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR3C readback register, as described here.

### 44.1.5.20.3 Diagram



#### 44.1.5.20.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. Always 0. |
| 28-16<br>R3_1_CT | Runs of Ones, Length 3 Count. Reads the final Runs of Ones, length 3 count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15-13<br>— | Reserved. Always 0. |
| 12-0<br>R3_0_CT | Runs of Zeroes, Length 3 Count. Reads the final Runs of Zeroes, length 3 count after entropy generation. Requires MCTL[PRGM] = 0. |

### 44.1.5.21 Statistical Check Run Length 4 Count Register (SCR4C)

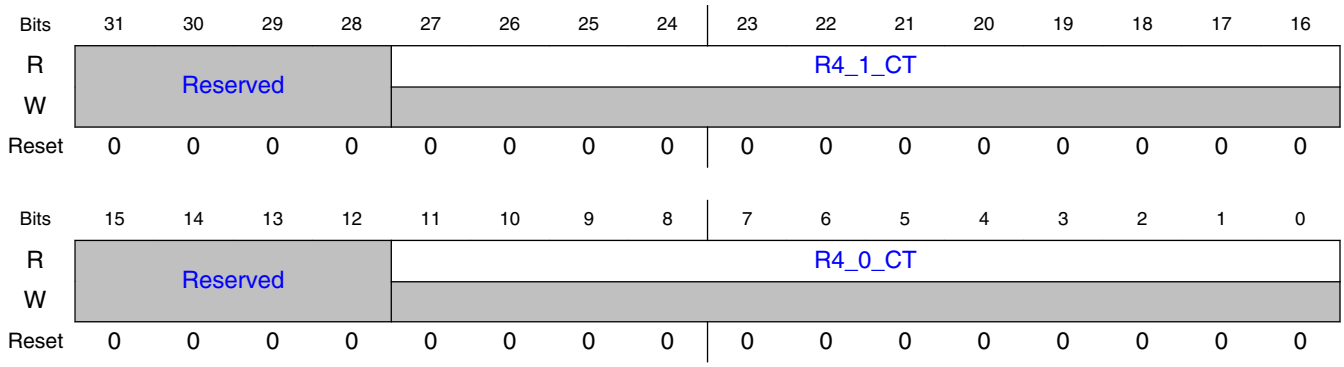
#### 44.1.5.21.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR4C | 30h | Accessible at this address when MCTL[PRGM] = 0] |

#### 44.1.5.21.2 Function

The Statistical Check Run Length 4 Counters Register is a read-only register used to read the final Run Length 4 counts after entropy generation. These counters start with the value in SCR4L[RUN4_MAX]. The R4_1_CT decrements each time four consecutive ones are sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time four consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x30) is used as SCR4L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR4C readback register, as described here.

### 44.1.5.21.3 Diagram




### 44.1.5.21.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. Always 0. |
| 27-16<br>R4_1_CT | Runs of One, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15-12<br>— | Reserved. Always 0. |
| 11-0<br>R4_0_CT | Runs of Zero, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0. |

## 44.1.5.22 Statistical Check Run Length 4 Limit Register (SCR4L)

### 44.1.5.22.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR4L | 30h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.22.2 Function

The Statistical Check Run Length 4 Limit Register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 4 must be greater than

(maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x30) is used as SCR4L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR4C readback register.

### 44.1.5.22.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | RUN4_RNG | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | RUN4_MAX | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

### 44.1.5.22.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. Always 0. |
| 27-16<br>RUN4_RNG | Run Length 4 Range. The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX - RUN4_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-12<br>— | Reserved. Always 0. |
| 11-0<br>RUN4_MAX | Run Length 4 Maximum Limit. Defines the maximum allowable runs of length 4 (for both 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.23 Statistical Check Run Length 5 Count Register (SCR5C)

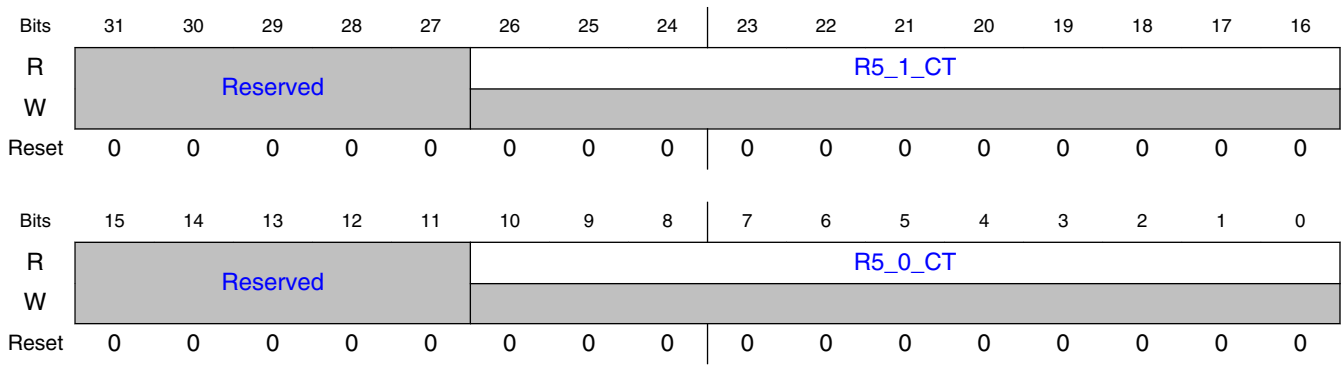
### 44.1.5.23.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR5C | 34h | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.23.2 Function

The Statistical Check Run Length 5 Counters Register is a read-only register used to read the final Run Length 5 counts after entropy generation. These counters start with the value in SCR5L[RUN5_MAX]. The R5_1_CT decrements each time five consecutive ones are sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time five consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x34) is used as SCR5L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR5C readback register, as described here.

### 44.1.5.23.3 Diagram




### 44.1.5.23.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. Always 0. |
| 26-16<br>R5_1_CT | Runs of One, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15-11<br>— | Reserved. Always 0. |
| 10-0<br>R5_0_CT | Runs of Zero, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0. |

## 44.1.5.24 Statistical Check Run Length 5 Limit Register (SCR5L)

### 44.1.5.24.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR5L | 34h | Accessible at this address when MCTL[PRGM] = 1] |

### 44.1.5.24.2 Function

The Statistical Check Run Length 5 Limit Register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 5 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x34) is used as SCR5L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR5C readback register.

### 44.1.5.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | RUN5_RNG | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | RUN5_MAX | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

### 44.1.5.24.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. Always 0. |
| 26-16<br>RUN5_RNG | Run Length 5 Range. The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX - RUN5_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-11<br>— | Reserved. Always 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10-0<br>RUN5_MAX | Run Length 5 Maximum Limit. Defines the maximum allowable runs of length 5 (for both 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.25 Statistical Check Run Length 6+ Count Register (SCR6PC)

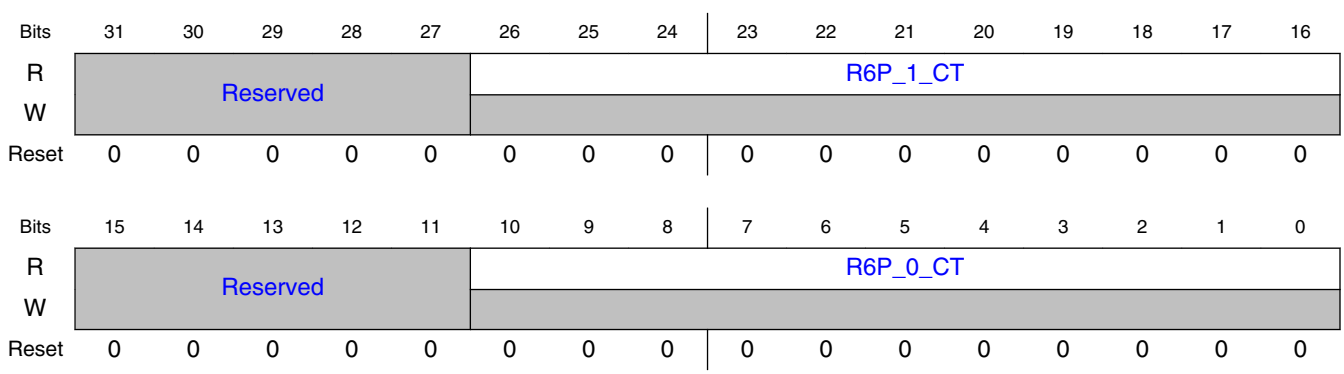
### 44.1.5.25.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR6PC | 38h | Accessible at this address when MCTL[PRGM] = 0] |

### 44.1.5.25.2 Function

The Statistical Check Run Length 6+ Counters Register is a read-only register used to read the final Run Length 6+ counts after entropy generation. These counters start with the value in SCR6PL[RUN6P_MAX]. The R6P_1_CT decrements each time six or more consecutive ones are sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time six or more consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x38) is used as SCR6PL if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register, as described here.

### 44.1.5.25.3 Diagram



#### 44.1.5.25.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. Always 0. |
| 26-16<br>R6P_1_CT | Runs of One, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0. |
| 15-11<br>— | Reserved. Always 0. |
| 10-0<br>R6P_0_CT | Runs of Zero, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0. |

### 44.1.5.26 Statistical Check Run Length 6+ Limit Register (SCR6PL)

#### 44.1.5.26.1 Offset

| Register | Offset | Description |
|---|---|---|
| SCR6PL | 38h | Accessible at this address when MCTL[PRGM] = 1] |

#### 44.1.5.26.2 Function

The Statistical Check Run Length 6+ Limit Register defines the allowable maximum and minimum number of runs of length 6 or more detected during entropy generation. To pass the test, the number of runs of length 6 or more (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 6 or more must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x38) is used as SCR6PL only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register.

### 44.1.5.26.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | RUN6P_RNG | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | RUN6P_MAX | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

### 44.1.5.26.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. Always 0. |
| 26-16<br>RUN6P_RNG | Run Length 6+ Range. The number of runs of length 6 or more (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX - RUN6P_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |
| 15-11<br>— | Reserved. Always 0. |
| 10-0<br>RUN6P_MAX | Run Length 6+ Maximum Limit. Defines the maximum allowable runs of length 6 or more (for both 0 and 1) detected during entropy generation. The number of runs of length 6 or more detected during entropy generation must be less than RUN6P_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. |

## 44.1.5.27 Status Register (STATUS)

### 44.1.5.27.1 Offset

| Register | Offset |
|---|---|
| STATUS | 3Ch |

### 44.1.5.27.2 Function

Various statistical tests are run as a normal part of the TRNG's entropy generation process. The least-significant 16 bits of the STATUS register reflect the result of each of these tests. The status of these bits will be valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the Miscellaneous Control Register.

Note that there is a very small probability that a statistical test will fail even though the TRNG is operating properly. If this happens the TRNG will automatically retry the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is nonzero, a retry is initiated. But if a statistical check fails when the retry count is zero, an error is generated by the RNG. By default RETRY_CT is initialized to 1, but software can increase the retry count by writing to the RTY_CT field in the SCMISC register.

All 0s will be returned if this register address is read while the RNG is in Program Mode (see PRGM field in MCTL register. If this register is read while the RNG is in Run Mode the value returned will be formatted as follows.

### 44.1.5.27.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | RETRY_CT | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TFMB | TFP | TFLR | TFSB | TF6PBR1 | TF6PBR0 | TF5BR1 | TF5BR0 | TF4BR1 | TF4BR0 | TF3BR1 | TF3BR0 | TF2BR1 | TF2BR0 | TF1BR1 | TF1BR0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.27.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. Always 0. |
| 19-16 | RETRY COUNT. This represents the current number of entropy generation retries left before a statistical text failure will cause the RNG to generate an error condition. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RETRY_CT | |
| 15<br>TFMB | Test Fail, Mono Bit. If TFMB=1, the Mono Bit Test has failed. |
| 14<br>TFP | Test Fail, Poker. If TFP=1, the Poker Test has failed. |
| 13<br>TFLR | Test Fail, Long Run. If TFLR=1, the Long Run Test has failed. |
| 12<br>TFSB | Test Fail, Sparse Bit. If TFSB=1, the Sparse Bit Test has failed. |
| 11<br>TF6PBR1 | Test Fail, 6 Plus Bit Run, Sampling 1s. If TF6PBR1=1, the 6 Plus Bit Run, Sampling 1s Test has failed. |
| 10<br>TF6PBR0 | Test Fail, 6 Plus Bit Run, Sampling 0s. If TF6PBR0=1, the 6 Plus Bit Run, Sampling 0s Test has failed. |
| 9<br>TF5BR1 | Test Fail, 5-Bit Run, Sampling 1s. If TF5BR1=1, the 5-Bit Run, Sampling 1s Test has failed. |
| 8<br>TF5BR0 | Test Fail, 5-Bit Run, Sampling 0s. If TF5BR0=1, the 5-Bit Run, Sampling 0s Test has failed. |
| 7<br>TF4BR1 | Test Fail, 4-Bit Run, Sampling 1s. If TF4BR1=1, the 4-Bit Run, Sampling 1s Test has failed. |
| 6<br>TF4BR0 | Test Fail, 4-Bit Run, Sampling 0s. If TF4BR0=1, the 4-Bit Run, Sampling 0s Test has failed. |
| 5<br>TF3BR1 | Test Fail, 3-Bit Run, Sampling 1s. If TF3BR1=1, the 3-Bit Run, Sampling 1s Test has failed. |
| 4<br>TF3BR0 | Test Fail, 3-Bit Run, Sampling 0s. If TF3BR0=1, the 3-Bit Run, Sampling 0s Test has failed. |
| 3<br>TF2BR1 | Test Fail, 2-Bit Run, Sampling 1s. If TF2BR1=1, the 2-Bit Run, Sampling 1s Test has failed. |
| 2<br>TF2BR0 | Test Fail, 2-Bit Run, Sampling 0s. If TF2BR0=1, the 2-Bit Run, Sampling 0s Test has failed. |
| 1<br>TF1BR1 | Test Fail, 1-Bit Run, Sampling 1s. If TF1BR1=1, the 1-Bit Run, Sampling 1s Test has failed. |
| 0<br>TF1BR0 | Test Fail, 1-Bit Run, Sampling 0s. If TF1BR0=1, the 1-Bit Run, Sampling 0s Test has failed. |

## 44.1.5.28 Entropy Read Register (ENT0 - ENT15)

### 44.1.5.28.1 Offset

For a = 0 to 15:

| Register | Offset | Description |
|---|---|---|
| ENTa | 40h + (a × 4h) | Word a |

### 44.1.5.28.2 Function

The RNG TRNG can be programmed to generate an entropy value that is readable via the SkyBlue bus. To do this, set the MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the MCTL[ENT_VAL] bit will be set to 1. At this point, ENT0 through ENT15 may be read to retrieve the 512-bit entropy value. Note that once ENT15 is read, the entropy value will be cleared and a new value will begin generation, so it is important that ENT15 be read last. These registers are readable only when MCTL[PRGM] = 0 (Run Mode) , MCTL[TRNG_ACC] = 1 (TRNG access mode) and MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid ENT15 register value, reading any ENT0 through ENT15 register would return zeros.

### 44.1.5.28.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ENT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ENT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.28.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ENT | Entropy Value. Will be non-zero only if MCTL[PRGM] = 0 Run Mode) and MCTL[ENT_VAL] = 1 (Entropy Valid). The most significant bits of the entropy are read from the lowest offset, and the least significant bits are read from the highest offset. Note that reading the highest offset also clears the entire entropy value, and starts a new entropy generation. |

# 44.1.5.29 Statistical Check Poker Count 1 and 0 Register (PKRCNT10)

## 44.1.5.29.1 Offset

| Register | Offset |
|---|---|
| PKRCNT10 | 80h |

## 44.1.5.29.2 Function

The Statistical Check Poker Count 1 and 0 Register is a read-only register used to read the final Poker test counts of 1h and 0h patterns. The Poker 0h Count increments each time a nibble of sample data is found to be 0h. The Poker 1h Count increments each time a nibble of sample data is found to be 1h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.29.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_1_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_0_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.29.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_1_CT | Poker 1h Count. Total number of nibbles of sample data which were found to be 1h. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_0_CT | Poker 0h Count. Total number of nibbles of sample data which were found to be 0h. Requires MCTL[PRGM] = 0. |

# 44.1.5.30 Statistical Check Poker Count 3 and 2 Register (PKRCNT32)

## 44.1.5.30.1 Offset

| Register | Offset |
|---|---|
| PKRCNT32 | 84h |

## 44.1.5.30.2 Function

The Statistical Check Poker Count 3 and 2 Register is a read-only register used to read the final Poker test counts of 3h and 2h patterns. The Poker 2h Count increments each time a nibble of sample data is found to be 2h. The Poker 3h Count increments each time a nibble of sample data is found to be 3h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.30.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_3_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_2_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.30.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_3_CT | Poker 3h Count. Total number of nibbles of sample data which were found to be 3h. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_2_CT | Poker 2h Count. Total number of nibbles of sample data which were found to be 2h. Requires MCTL[PRGM] = 0. |

# 44.1.5.31 Statistical Check Poker Count 5 and 4 Register (PKRCNT54)

## 44.1.5.31.1 Offset

| Register | Offset |
|---|---|
| PKRCNT54 | 88h |

## 44.1.5.31.2 Function

The Statistical Check Poker Count 5 and 4 Register is a read-only register used to read the final Poker test counts of 5h and 4h patterns. The Poker 4h Count increments each time a nibble of sample data is found to be 4h. The Poker 5h Count increments each time a nibble of sample data is found to be 5h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.31.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_5_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_4_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.31.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_5_CT | Poker 5h Count. Total number of nibbles of sample data which were found to be 5h. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_4_CT | Poker 4h Count. Total number of nibbles of sample data which were found to be 4h. Requires MCTL[PRGM] = 0. |

## 44.1.5.32 Statistical Check Poker Count 7 and 6 Register (PKRCNT76)

### 44.1.5.32.1 Offset

| Register | Offset |
|---|---|
| PKRCNT76 | 8Ch |

### 44.1.5.32.2 Function

The Statistical Check Poker Count 7 and 6 Register is a read-only register used to read the final Poker test counts of 7h and 6h patterns. The Poker 6h Count increments each time a nibble of sample data is found to be 6h. The Poker 7h Count increments each time a nibble of sample data is found to be 7h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

### 44.1.5.32.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_7_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_6_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.32.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_7_CT | Poker 7h Count. Total number of nibbles of sample data which were found to be 7h. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_6_CT | Poker 6h Count. Total number of nibbles of sample data which were found to be 6h. Requires MCTL[PRGM] = 0. |

# 44.1.5.33 Statistical Check Poker Count 9 and 8 Register (PKRCNT98)

## 44.1.5.33.1 Offset

| Register | Offset |
|---|---|
| PKRCNT98 | 90h |

## 44.1.5.33.2 Function

The Statistical Check Poker Count 9 and 8 Register is a read-only register used to read the final Poker test counts of 9h and 8h patterns. The Poker 8h Count increments each time a nibble of sample data is found to be 8h. The Poker 9h Count increments each time a nibble of sample data is found to be 9h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.33.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_9_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_8_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.33.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_9_CT | Poker 9h Count. Total number of nibbles of sample data which were found to be 9h. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_8_CT | Poker 8h Count. Total number of nibbles of sample data which were found to be 8h. Requires MCTL[PRGM] = 0. |

## 44.1.5.34 Statistical Check Poker Count B and A Register (PKRCNTBA)

### 44.1.5.34.1 Offset

| Register | Offset |
|---|---|
| PKRCNTBA | 94h |

### 44.1.5.34.2 Function

The Statistical Check Poker Count B and A Register is a read-only register used to read the final Poker test counts of Bh and Ah patterns. The Poker Ah Count increments each time a nibble of sample data is found to be Ah. The Poker Bh Count increments each time a nibble of sample data is found to be Bh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

### 44.1.5.34.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_B_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_A_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.34.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_B_CT | Poker Bh Count. Total number of nibbles of sample data which were found to be Bh. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_A_CT | Poker Ah Count. Total number of nibbles of sample data which were found to be Ah. Requires MCTL[PRGM] = 0. |

# 44.1.5.35 Statistical Check Poker Count D and C Register (PKRCNTDC)

## 44.1.5.35.1 Offset

| Register | Offset |
|---|---|
| PKRCNTDC | 98h |

## 44.1.5.35.2 Function

The Statistical Check Poker Count D and C Register is a read-only register used to read the final Poker test counts of Dh and Ch patterns. The Poker Ch Count increments each time a nibble of sample data is found to be Ch. The Poker Dh Count increments each time a nibble of sample data is found to be Dh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.35.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_D_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_C_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.35.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_D_CT | Poker Dh Count. Total number of nibbles of sample data which were found to be Dh. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_C_CT | Poker Ch Count. Total number of nibbles of sample data which were found to be Ch. Requires MCTL[PRGM] = 0. |

# 44.1.5.36 Statistical Check Poker Count F and E Register (PKRCNTFE)

## 44.1.5.36.1 Offset

| Register | Offset |
|---|---|
| PKRCNTFE | 9Ch |

## 44.1.5.36.2 Function

The Statistical Check Poker Count F and E Register is a read-only register used to read the final Poker test counts of Fh and Eh patterns. The Poker Eh Count increments each time a nibble of sample data is found to be Eh. The Poker Fh Count increments each time a nibble of sample data is found to be Fh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

## 44.1.5.36.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_F_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PKR_E_CT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 44.1.5.36.4 Fields

| Field | Function |
|---|---|
| 31-16<br>PKR_F_CT | Poker Fh Count. Total number of nibbles of sample data which were found to be Fh. Requires MCTL[PRGM] = 0. |
| 15-0<br>PKR_E_CT | Poker Eh Count. Total number of nibbles of sample data which were found to be Eh. Requires MCTL[PRGM] = 0. |

# 44.1.5.37 Security Configuration Register (SEC_CFG)

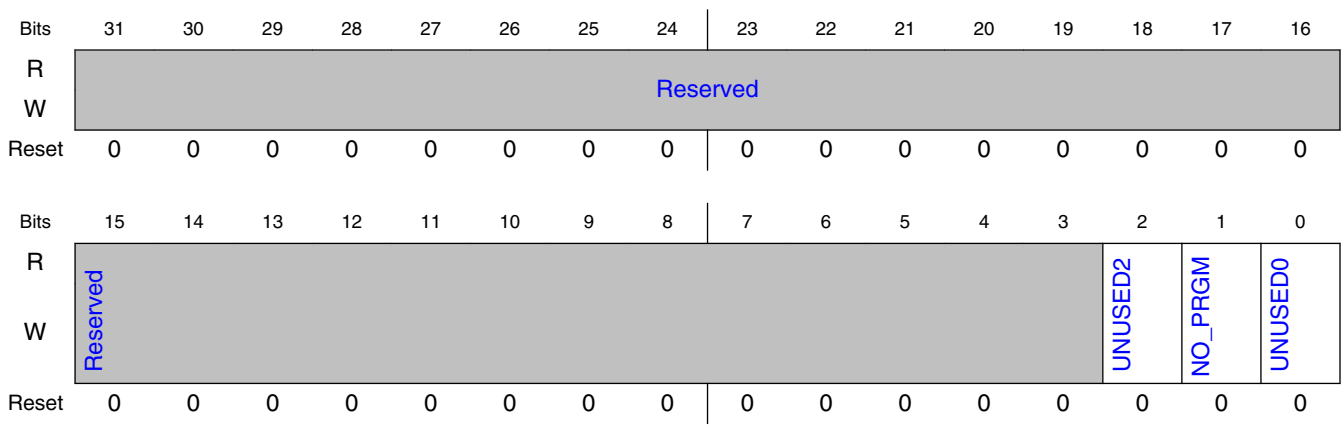
## 44.1.5.37.1 Offset

| Register | Offset |
|---|---|
| SEC_CFG | A0h |

## 44.1.5.37.2 Function

The Security Configuration Register is a read/write register used to control the test mode, programmability and state modes of the TRNG. Clears on asynchronous system hard reset or POR.

## 44.1.5.37.3 Diagram




## 44.1.5.37.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2<br>UNUSED2 | This bit is unused. Ignore. |
| 1<br>NO_PRGM | If set, the TRNG registers cannot be programmed. That is, regardless of the TRNG access mode in the TRNG Miscellaneous Control Register.<br>0b - Programability of registers controlled only by the Miscellaneous Control Register's access mode bit.<br>1b - Overides Miscellaneous Control Register access mode and prevents TRNG register programming. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>UNUSED0 | This bit is unused. Ignore. |

## 44.1.5.38 Interrupt Control Register (INT_CTRL)

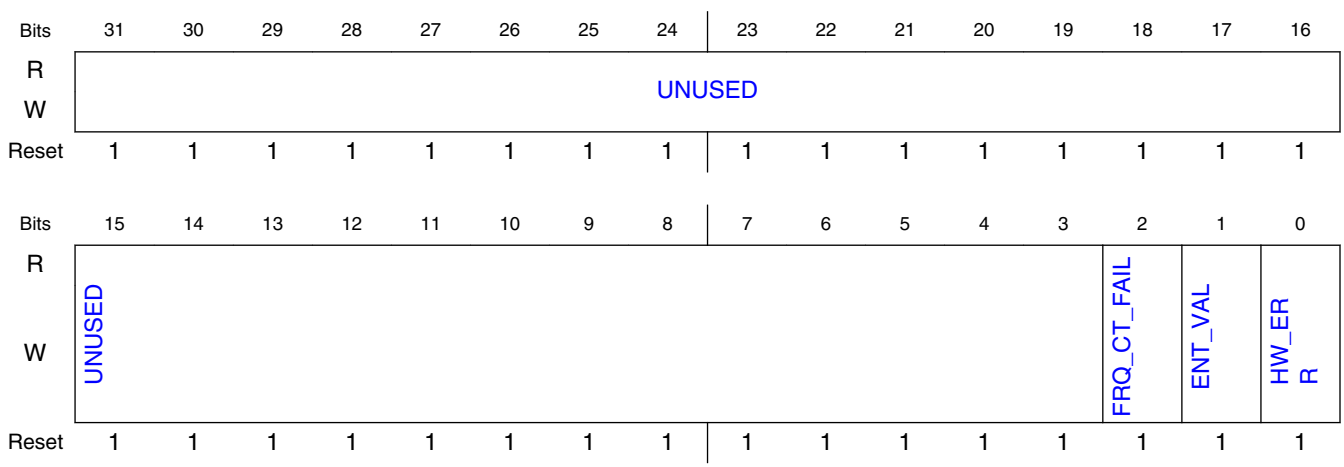
### 44.1.5.38.1 Offset

| Register | Offset |
|---|---|
| INT_CTRL | A4h |

### 44.1.5.38.2 Function

The Interrupt Control Register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG. See INT_STATUS register description. Each interrupt can be cleared by de-asserting the corresponding bit in the INT_CTRL register. Only a new interrupt will reassert the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

### 44.1.5.38.3 Diagram



#### 44.1.5.38.4 Fields

| Field | Function |
|---|---|
| 31-3<br>UNUSED | Reserved but writeable. |
| 2<br>FRQ_CT_FAIL | Same behavior as bit 0 of this register.<br>0b - Same behavior as bit 0 of this register.<br>1b - Same behavior as bit 0 of this register. |
| 1<br>ENT_VAL | Same behavior as bit 0 of this register.<br>0b - Same behavior as bit 0 of this register.<br>1b - Same behavior as bit 0 of this register. |
| 0<br>HW_ERR | Bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted.<br>0b - Corresponding bit of INT_STATUS register cleared.<br>1b - Corresponding bit of INT_STATUS register active. |

### 44.1.5.39 Mask Register (INT_MASK)

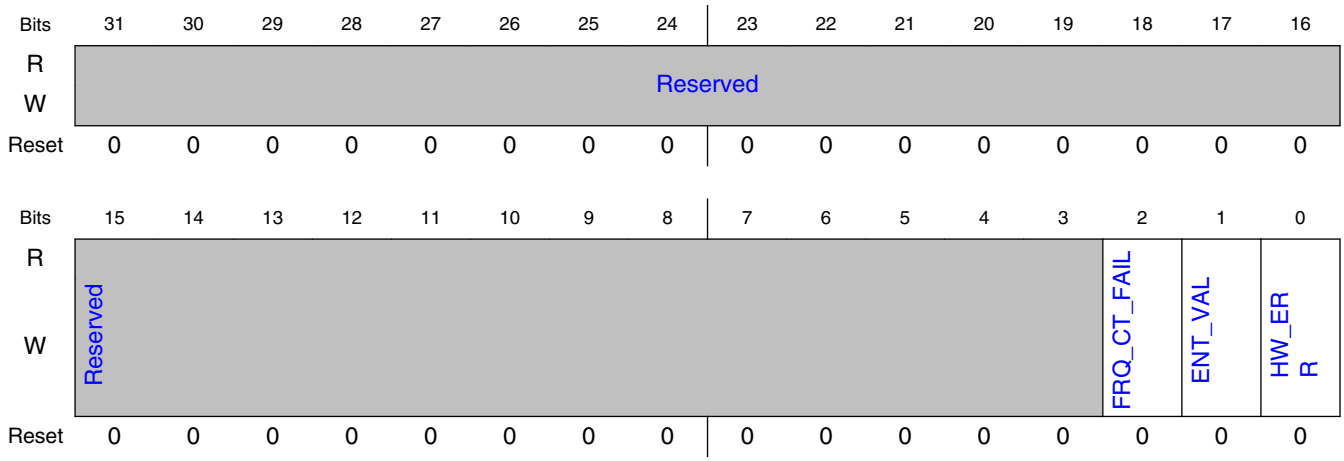
#### 44.1.5.39.1 Offset

| Register | Offset |
|---|---|
| INT_MASK | A8h |

#### 44.1.5.39.2 Function

The Interrupt Mask Register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG. See INT_STATUS register descrption. Each interrupt can be masked/disabled by de-asserting the corresponding bit in the INT_MASK register. Only setting this bit high will re-enable the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

### 44.1.5.39.3 Diagram




### 44.1.5.39.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2<br>FRQ_CT_FAIL | Same behavior as bit 0 of this register.<br>0b - Same behavior as bit 0 of this register.<br>1b - Same behavior as bit 0 of this register. |
| 1<br>ENT_VAL | Same behavior as bit 0 of this register.<br>0b - Same behavior as bit 0 of this register.<br>1b - Same behavior as bit 0 of this register. |
| 0<br>HW_ERR | Bit position that can be cleared if corresponding bit of INT_STATUS has been asserted.<br>0b - Corresponding interrupt of INT_STATUS is masked.<br>1b - Corresponding bit of INT_STATUS is active. |

## 44.1.5.40 Interrupt Status Register (INT_STATUS)

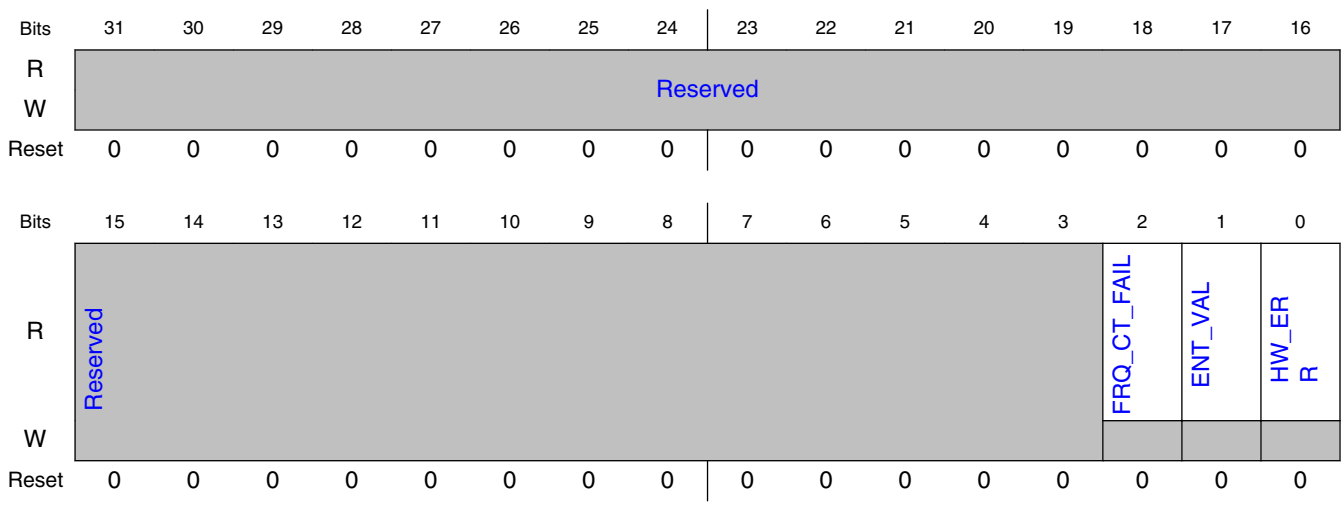
### 44.1.5.40.1 Offset

| Register | Offset |
|---|---|
| INT_STATUS | ACh |

### 44.1.5.40.2 Function

The Interrupt Status Register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The TRNG interrupt signal alerts that TRNG has either generated a Frequency Count Fail, Entropy Valid or Error Interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the INT_STATUS register. Each interrupt can be temporarily cleared by de-asserting the corresponding bit in the INT_CTRL register. To mask the interrupts, clear the corresponding bits in the INT_MASK register. The description of each of the 3 interrupts is defined in the Block Guide under the MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

### 44.1.5.40.3 Diagram




### 44.1.5.40.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2<br>FRQ_CT_FAIL | Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator.<br>0b - No hardware nor self test frequency errors.<br>1b - The frequency counter has detected a failure. |
| 1<br>ENT_VAL | Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared when ENT15 is read. (ENT0 through ENT14 should be read before reading ENT15).<br>0b - Busy generation entropy. Any value read is invalid. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - TRNG can be stopped and entropy is valid if read. |
| 0<br>HW_ERR | Read: Error status. 1 = error detected. 0 = no error. Any HW error in the TRNG will trigger this interrupt.<br>0b - no error<br>1b - error detected. |

## 44.1.5.41 Version ID Register (MS) (VID1)

### 44.1.5.41.1 Offset

| Register | Offset |
|---|---|
| VID1 | F0h |

### 44.1.5.41.2 Function

The Version ID Register (VID1) is a MSW read only register used to identify the version of the TRNG in use. This register as well as VID2 should both be read to verify the expected version.

### 44.1.5.41.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IP_ID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJ_REV | | | | | | | | MIN_REV | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.41.4 Fields

| Field | Function |
|---|---|
| 31-16<br>IP_ID | Shows the IP ID.<br>0000000000110000b - ID for TRNG. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-8<br>MAJ_REV | Shows the IP's Major revision of the TRNG.<br>00000001b - Major revision number for TRNG. |
| 7-0<br>MIN_REV | Shows the IP's Minor revision of the TRNG.<br>00000000b - Minor revision number for TRNG. |

## 44.1.5.42 Version ID Register (LS) (VID2)

### 44.1.5.42.1 Offset

| Register | Offset |
|---|---|
| VID2 | F4h |

### 44.1.5.42.2 Function

The Version ID Register LSB is a read only register used to identify the architecture of the TRNG in use. This register as well as VID1 should both be read to verify the expected version.

### 44.1.5.42.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERA | | | | | | | | INTG_OPT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ECO_REV | | | | | | | | CONFIG_OPT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 44.1.5.42.4 Fields

| Field | Function |
|---|---|
| 31-24<br>ERA | Shows the compile options for the TRNG.<br>00000000b - COMPILE_OPT for TRNG. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>INTG_OPT | Shows the integration options for the TRNG.<br>00000000b - INTG_OPT for TRNG. |
| 15-8<br>ECO_REV | Shows the IP's ECO revision of the TRNG.<br>00000000b - TRNG_ECO_REV for TRNG. |
| 7-0<br>CONFIG_OPT | Shows the IP's Configuaration options for the TRNG.<br>00000000b - TRNG_CONFIG_OPT for TRNG. |

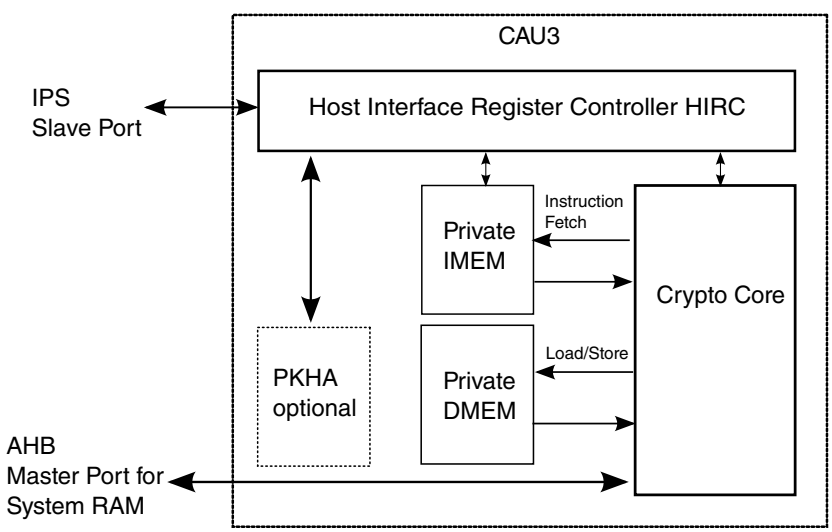
# Chapter 45
# Cryptographic Acceleration Unit (CAU3)

## 45.1 Introduction

The Version 3 Cryptographic Acceleration Unit (CAU3) is a bus mastering IP module that provides hardware acceleration of a variety of cryptographic symmetric key and secure hash algorithms including DES, 3DES, AES-{128,192,256}, SHA-{1,256,512} as well as acceleration of basic public key cryptography including Elliptical Curve Cryptography (ECC). The execution of these algorithms is controlled by optimized firmware developed by NXP that executes on the CAU3 module.

The CAU3 includes four major components: a NXP-programmable 4-stage pipeline CryptoCore, two private memories for local instruction (IMEM) and data (DMEM) storage and a host interface register controller (HIRC) that connects the system IPS slave peripheral bus to the CAU3. The figure bellow describes these components.

**Figure 45-1. CAU3 Major Components**



From the system perspective, the CAU3 is a two-terminal device with a bus master connection to the system AHB bus fabric (for accessing crypto data in system RAM and specific slave peripheral modules) and a slave connection to the IPS bus (for the host to program and control the CAU3).

The host interface is optimized to provide a streamlined mechanism to quickly initialize the CAU3 with minimal overhead, while the CryptoCore is a single-issue 4-stage pipelined processor. The CryptoCore's instruction set architecture includes a combination of standard integer instructions needed for general processing as well as a set of crypto instructions specifically defined to accelerate the supported cryptographic algorithms.

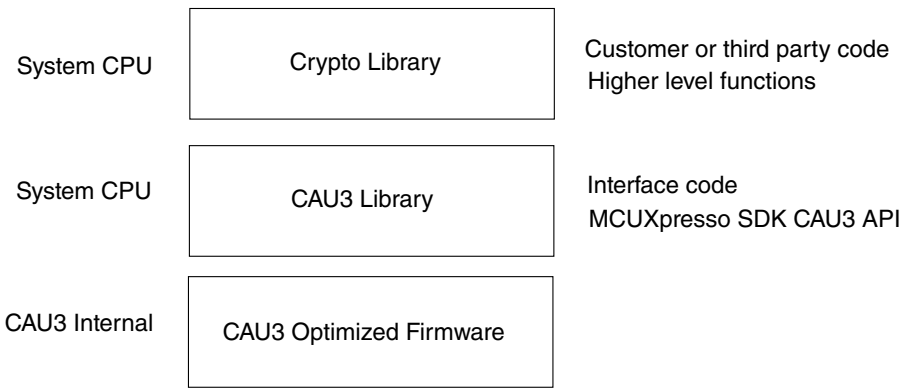
The general flow of operations contains the following steps:

1. Since the CAU3 is a shared system module, its "ownership" must first be claimed by the host processor. This is accomplished via a hardware semaphore implemented in the HIRC submodule. Once ownership has been acquired, the host processor continues with the subsequent steps.
2. The host programs the needed input arguments by writing their values directly into the CryptoCore's programming model including a starting PC value which indexes into a branch table based at address 0x0 of the IMEM.IRAM memory. The final operation is a write to the CryptoCore's CMD register which initiates execution.
3. The CryptoCore follows a "run to completion" model, and following a write to its CMD register, executes the selected firmware routine(s) to perform the desired cryptographic function(s).
4. Once the function's execution is complete, the CAU3 provides a user- programmable signaling mechanism. The task completion can be configured to: generate an interrupt back to the host, signal an event completion to the host, generate a DMA request to the appropriate DMA, and simply change the state of its RUN/STOP/ HALT status (to allow host polling).
5. After receiving the appropriate notification, the host processor releases the CAU3 semaphore, making the module available again.

Within the CAU3's micro-architecture, the HIRC submodule contains a number of local control, status and configuration registers and supports read/write access to the CryptoCore's programming model (General-purpose registers R[0-31], PC, CMD, CF) and the ability to read/write the CAU3's private memories (IMEM, DMEM).

NXP provides an optimized C-function library that implements the appropriate software building blocks to execute the high-level security functions as well as an optimized firmware library containing the CryptoCore code. The figure below describes the CAU3 software stack.

**Figure 45-2. CAU3 Software Stack**




## 45.2 Block Diagram

A more detailed view of the CAU3 micro-architecture is presented below. The CryptoCore's design is partitioned into three major functions: an instruction fetch pipeline for controlling accesses of firmware instructions from the private, local IMEM, both IRAM and IROM, the execution unit pipeline for register-to-register operations and the load/store unit for accessing system RAM using the AHB master bus port and private accesses to the local DMEM.DRAM.

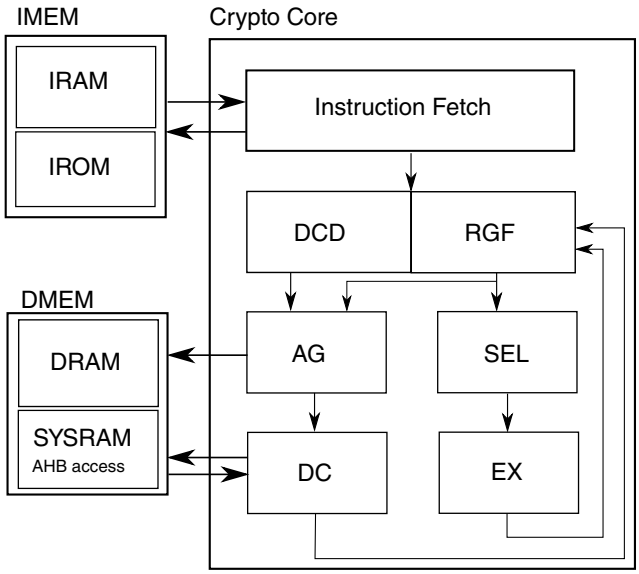



**Figure 45-3. CryptoCore Highest Level block diagram**

The CAU3's local private memories (IMEM and DMEM.DRAM) provide zero wait-state response to instruction fetches and data loads/stores from the CryptoCore. These memories can be initialized indirectly under control of the host processor, but once the firmware is written into the IMEM.IRAM, the host processor access port is typically disabled. Once disabled, the local memories are only accessible by the CryptoCore. The IMEM.IRAM storage includes a function branch table based at address 0 followed by the specific CAU3 firmware functions needed in a given system. The IMEM.IROM includes a small number of firmware functions, mostly involving CAU3 initialization and/or handling of a security violation detected at the SoC level. These functions include zeroing out the local DMEM.DRAM and CryptoCore registers to protect any key information. The DMEM.DRAM is used for local private storage, for both temporary variables and secret key management. It is also used to store data used during public key calculations.

## 45.3 Overview

The CAU3 is a cryptographic sub-system that includes a dedicated core, a dedicated instruction memory (IRAM and IROM) and a dedicated data RAM for autonomously implement symmetric keys algorithms, such as 3DES, AES-128/192/256, AES-CBC, AES-CCM, AES-GCM as well as secure hash functions including SHA-1, SHA-256 and SHA-512. Hardware acceleration for public key algorithms including RSA and ECC is also provided.

The set of implemented algorithms provides excellent support for network security standards, such as SSL and IPsec.

The cryptographic algorithms are implemented in the CAU3 firmware using dedicated instructions that are executed by a cryptographic core to efficiently execute the cryptographic algorithms. Hardware acceleration is provided for the algorithms that are not completely implemented in the CAU3 module, allowing fine-grained partitioning of functions between hardware and software:

- Implement the basic security kernel functions using the cryptographic core processor instructions
- Implement higher level functions in software by using the standard processor core instructions in addition to the calls to CAU3 firmware with its cryptographic core instructions

The CAU3 implements a set of cryptographic functions programmed in the module local private instruction memory (IRAM and/or IROM) . These firmware instructions are executed by the cryptographic core. The module can access cipher text or plain text stored in the system RAM without the host CPU intervention using a crossbar master AHB interface. A private RAM (DMEM.DRAM) provides local storage of temporary data variables and secret key management, to avoid exposing the data in any system memory.

## 45.4 Features

The CAU3 includes the following distinctive features:

- Backwards compatible with MMCAU
  - Symmetric key functions: DES, 3DES, AES-{128,192,256}
  - Secure hash functions: SHA-1, SHA-256
- Added SHA-512 secure hash acceleration
- AES-CBC, AES-CCM, AES-GCM
- RSA and ECC acceleration
- Programmable task completion signaling: interrupt, event completion, DMA request
- Significantly improved performance and power efficiency
- Common, portable CAU3 library written in C
- ARM embedded TLS library reference design
- Hardware size appropriate for low cost CM0+ and CM4 MCUs
- Flexible architecture

## 45.5 CAU3 register descriptions

This section describes the CAU3 registers. Certain register resources of the CAU3 programming model can only be accessed while in secure privileged mode. Additionally, unless noted otherwise, the programming model registers can be accessed via 8-, 16- or 32-bit reads and 32-bit write references. Attempted accesses in a different operating mode, using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error unless noted otherwise. Due to security concerns, in general, any IPS read access that is error terminated must respond with zeroes as the data. Refer to Access to Memory Mapped Registers for details.

### 45.5.1 CAU3 Memory map

CAU3 base address: 4102_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Processor Core Type (PCT) | 32 | RO | 4B41_6000h |
| 4h | Memory Configuration (MCFG) | 32 | RO | 0A04_0700h |
| 10h | Control Register (CR) | 32 | RW | 8000_0000h |
| 14h | Status Register (SR) | 32 | W1C | 0102_0900h |
| 20h | Debug Control/Status Register (DBGCSR) | 32 | RW | 4000_0000h |
| 24h | Debug PC Breakpoint Register (DBGPBR) | 32 | RW | 0000_0000h |
| 30h | Debug Memory Command Register (DBGMCMD) | 32 | RW | 8800_0000h |
| 34h | Debug Memory Address Register (DBGMADR) | 32 | RW | 0000_0000h |
| 38h | Debug Memory Data Register (DBGMDR) | 32 | RW | 0000_0000h |
| F0h | Semaphore Register (SEMA4) | 32 | WORZ | 0000_0000h |
| F4h | Semaphore Ownership Register (SMOWNR) | 32 | RO | 8000_0000h |
| FCh | Address Remap Register (ARR) | 32 | RW | 0000_0000h |
| 180h - 1F4h | CryptoCore General Purpose Registers (CC_R0 - CC_R29) | 32 | RW | 0000_0000h |
| 1F8h | General Purpose R30 (CC_R30) | 32 | RW | 0000_0000h |
| 1FCh | General Purpose R31 (CC_R31) | 32 | RW | 0000_0000h |
| 200h | Program Counter (CC_PC) | 32 | RW | 0008_0000h |
| 204h | Start Command Register (CC_CMD) | 32 | WO | 0000_0000h |
| 208h | Condition Flag (CC_CF) | 32 | RO | 0000_0000h |
| 400h | Mode Register (PublicKey) (MDPK) | 32 | RW | 0000_0000h |
| 430h | Command Register (COM) | 32 | WO | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 434h | Control Register (CTL) | 32 | RW | 0000_0000h |
| 440h | Clear Written Register (CW) | 32 | WO | 0000_0000h |
| 448h | Status Register (STA) | 32 | W1C | 0000_0000h |
| 44Ch | Error Status Register (ESTA) | 32 | RO | 0000_0000h |
| 480h | PKHA A Size Register (PKASZ) | 32 | RW | 0000_0000h |
| 488h | PKHA B Size Register (PKBSZ) | 32 | RW | 0000_0000h |
| 490h | PKHA N Size Register (PKNSZ) | 32 | RW | 0000_0000h |
| 498h | PKHA E Size Register (PKESZ) | 32 | RW | 0000_0000h |
| 4F0h | PKHA Revision ID 1 (PKHA_VID1) | 32 | RO | 0034_0100h |
| 4F4h | PKHA Revision ID 2 (PKHA_VID2) | 32 | RO | 0000_0202h |
| 4F8h | CHA Revision ID (CHA_VID) | 32 | RO | 0047_0000h |
| 600h | PKHA Clock Control Register (PKHA_CCR) | 32 | RW | See description. |
| 604h | Global Status Register (GSR) | 32 | RO | See description. |
| 608h | Clock Linear Feedback Shift Register (CKLFSR) | 32 | RW | 0000_0000h |
| 800h - 87Ch | PKHA A0 Register (PKA0_0 - PKA0_31) | 32 | RW | 0000_0000h |
| 880h - 8FCh | PKHA A1 Register (PKA1_0 - PKA1_31) | 32 | RW | 0000_0000h |
| 900h - 97Ch | PKHA A2 Register (PKA2_0 - PKA2_31) | 32 | RW | 0000_0000h |
| 980h - 9FCh | PKHA A3 Register (PKA3_0 - PKA3_31) | 32 | RW | 0000_0000h |
| A00h - A7Ch | PKHA B0 Register (PKB0_0 - PKB0_31) | 32 | RW | 0000_0000h |
| A80h - AFCh | PKHA B1 Register (PKB1_0 - PKB1_31) | 32 | RW | 0000_0000h |
| B00h - B7Ch | PKHA B2 Register (PKB2_0 - PKB2_31) | 32 | RW | 0000_0000h |
| B80h - BFCh | PKHA B3 Register (PKB3_0 - PKB3_31) | 32 | RW | 0000_0000h |
| C00h - C7Ch | PKHA N0 Register (PKN0_0 - PKN0_31) | 32 | RW | 0000_0000h |
| C80h - CFCh | PKHA N1 Register (PKN1_0 - PKN1_31) | 32 | RW | 0000_0000h |
| D00h - D7Ch | PKHA N2 Register (PKN2_0 - PKN2_31) | 32 | RW | 0000_0000h |
| D80h - DFCh | PKHA N3 Register (PKN3_0 - PKN3_31) | 32 | RW | 0000_0000h |
| E00h - FFCh | PKHA E Register (PKE_0 - PKE_127) | 32 | WO | 0000_0000h |

## 45.5.2 Processor Core Type (PCT)

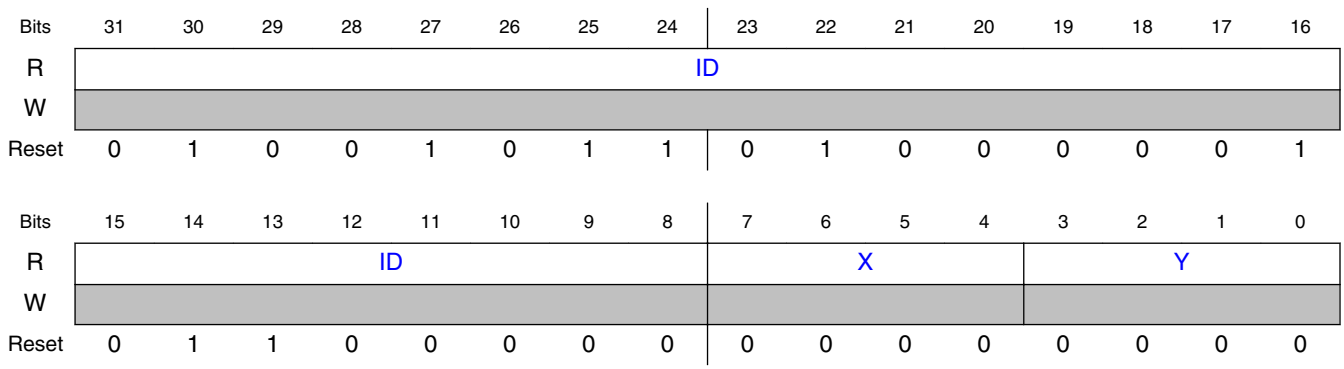
### 45.5.2.1 Offset

| Register | Offset |
|---|---|
| PCT | 0h |

### 45.5.2.2 Function

This read-only register contains the CAU3 Processor Core Type information. This register can only be read by a bus master operating in secure, privileged mode.

### 45.5.2.3 Diagram




### 45.5.2.4 Fields

| Field | Function |
|---|---|
| 31-8<br>ID | Module ID number<br>The Module Identification Number read only field indicates the configuration of the CAU3 module.<br>01001011010000010110000 0b - ID number for basic configuration<br>01001011010000010110000 1b - ID number for PKHA configuration |
| 7-4<br>X | Major version number<br>This read only field returns the major version number.<br>0000b - Major version number |
| 3-0<br>Y | Minor version number<br>This read only field returns the minor version number.<br>0000b - Minor version number |

## 45.5.3 Memory Configuration (MCFG)

### 45.5.3.1 Offset

| Register | Offset |
|---|---|
| MCFG | 4h |

### 45.5.3.2 Function

This read-only register contains the local private memory configuration information. This register can only be read by a bus master operating in secure, privileged mode.

### 45.5.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | IRAM_SZ | | | | Reserved | | | | IROM_SZ | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | DRAM_SZ | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.3.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>IRAM_SZ | Instruction RAM Size<br>Private Instruction RAM size. The 4 bits field should be seen as xxx.y<br>0000b - No memory module<br>0100b - 2K bytes<br>0101b - 3K bytes<br>0110b - 4K bytes<br>0111b - 6K bytes<br>1000b - 8K bytes<br>1001b - 12K bytes<br>1010b - 16K bytes<br>1011b - 24K bytes<br>1100b - 32K bytes |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1101b - 48K bytes<br>1110b - 64K bytes<br>1111b - 96K bytes |
| 23-20<br>— | Reserved |
| 19-16<br>IROM_SZ | Instruction ROM Size<br>Private Instruction ROM size. The 4 bits field should be seen as xxx.y<br>0000b - No memory module<br>0100b - 2K bytes<br>0101b - 3K bytes<br>0110b - 4K bytes<br>0111b - 6K bytes<br>1000b - 8K bytes<br>1001b - 12K bytes<br>1010b - 16K bytes<br>1011b - 24K bytes<br>1100b - 32K bytes<br>1101b - 48K bytes<br>1110b - 64K bytes<br>1111b - 96K bytes |
| 15-12<br>— | Reserved |
| 11-8<br>DRAM_SZ | Data RAM Size<br>Private Data RAM size. The 4 bits field should be seen as xxx.y, where xxx are power of 2 and y is a middle value to increase the resolution of the bit field.<br>0000b - No memory module<br>0100b - 2K bytes<br>0101b - 3K bytes<br>0110b - 4K bytes<br>0111b - 6K bytes<br>1000b - 8K bytes<br>1001b - 12K bytes<br>1010b - 16K bytes<br>1011b - 24K bytes<br>1100b - 32K bytes<br>1101b - 48K bytes<br>1110b - 64K bytes<br>1111b - 96K bytes |
| 7-0<br>— | Reserved |

## 45.5.4 Control Register (CR)

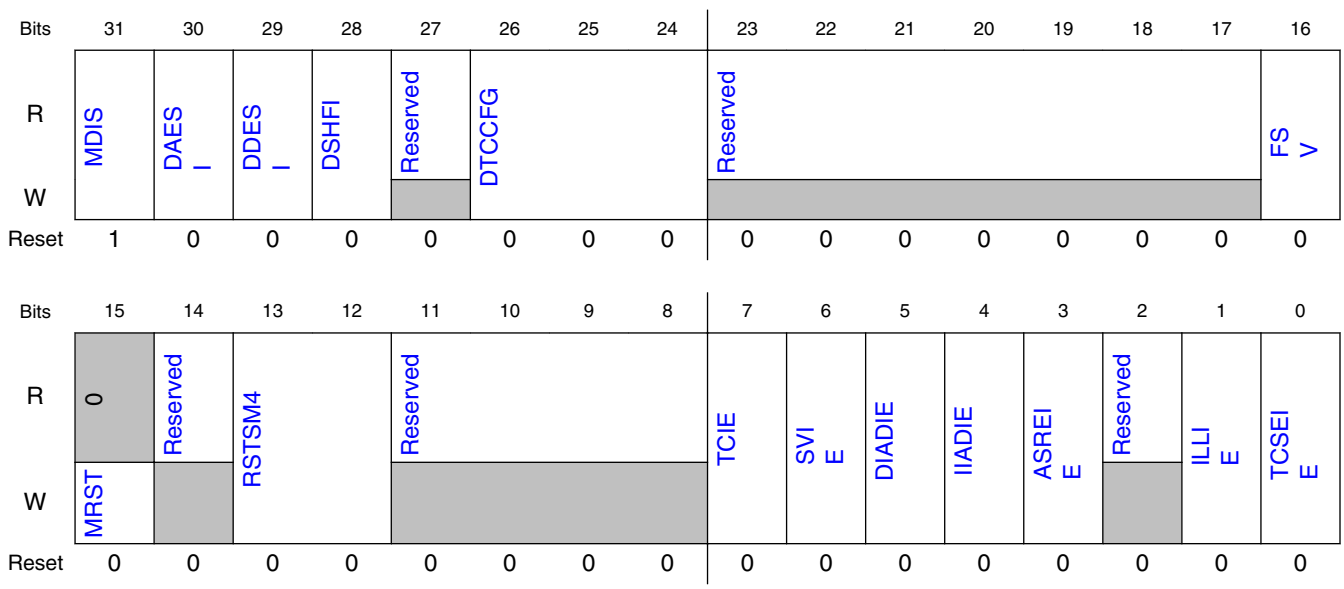
### 45.5.4.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 45.5.4.2 Function

This register provides control bits for CAU3 operation. This register can only be accessed by a bus master operating in secure, privileged mode. Accesses to this register are also allowed when the CryptoCore is in low power mode.

### 45.5.4.3 Diagram




### 45.5.4.4 Fields

| Field | Function |
|---|---|
| 31<br>MDIS | Module Disable<br>If set, this bit disables the CAU3 program execution and then puts the CryptoCore into low power mode by disabling the module clocks and consequently disabling all functions of the module. While disabled, accesses to CAU3 register resources are not allowed with the exception of PCT, MCFG, CR and SR registers which are accessible. In case of SR, it is only accessible for read operations. Furthermore, |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | blocked accesses return a transfer error. MDIS should be set only when CAU3 is at STOP state. If MDIS is negated, the CryptoCore exits from the low power mode by restoring the clocks. See section Access to Memory Mapped Registers for details.<br>0b - CAU3 exits from low power mode<br>1b - CAU3 enters low power mode |
| 30<br>DAESI | Disable AES Instructions<br>If set, this bit disables all AES crypto instructions (AESS, AESIS, AESSC, AESISC, AESR and AESIR). If the execution of an AES instruction is attemped and this bit is set, that instruction is treated as an illegal instruction and is terminated with a stop-with-error condition. This bit field is sticky, meaning that once it is written to logic 1 it cannot be cleared by a subsequent write access. It is only cleared by a hardware reset. This bit should be changed only when the CCore is in STOP or HALT states.<br>0b - AES instructions are enabled<br>1b - AES instructions are disabled |
| 29<br>DDESI | Disable DES Instructions<br>If this bit is set, the DES instructions are disabled (DESR and DESK), meaning that they are considered invalid instructions. Any attempt to execute those instructions when this bit is set generates an stop-with-error program termination. This bit is sticky, meaning that once written to logic 1 it cannot be changed by a subsequent write access. Only a hardware reset clears this bit. This bit should be changed only when the CCore is in STOP or HALT states.<br>0b - DES instructions are enabled<br>1b - DES instructions are disabled |
| 28<br>DSHFI | Disable Secure Hash Function Instructions<br>This control bit disables the Secure Hash Functions instructions (HFP, HFC, HFM, HF2A, HF2B, HF2C, and HF2D), meaning that these functions are interpreted as invalid instructions. Any attempt to execute Secure Hash instructions when this bit is set generates a stop-with-error program termination. This bit is sticky, meaning that once written to logic 1 it cannot be changed by a subsequent write access. Only a hardware reset clears this bit. This bit should be changed only when the CCore is in STOP or HALT states.<br>0b - Secure Hash Functions are enabled<br>1b - Secure Hash Functions are disabled |
| 27<br>— | Reserved |
| 26-24<br>DTCCFG | Default Task Completion Configuration<br>This control field defines the action to be taken after a task is completed. Note that the task completion report status can be modified by the CMD register bit field [2:0]. In case CMD[2:0]=0 the DTCCFG is used to report the task completion status. If CMD[2:0]!=0 then the CMD[2:0] is used in the place of DTCCFG. The decoding of CMD[2:0] and DTCCFG bits have the same meaning. This bit should be changed only when the CCore is in STOP or HALT states.<br>000b - no explicit action<br>001b - Issue an Interrupt Request<br>010b - Assert Event Completion Signal<br>100b - Issue a DMA request |
| 23-17<br>— | Reserved |
| 16<br>FSV | Force Security Violation Test<br>The force security violation bit simulates a system security violation event. This bit is used for software test of the security violation counter measures implemented in the crypto core. This bit is sticky. Once set it remains set until a Module (MRST) or hardware reset occurs.<br>0b - no violation is forced<br>1b - force security violation |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>MRST | Module Reset<br>The Module reset is self negated by the crypto core hardware. It is always read as logic zero. This control bit is used to force a task to end as if a STOP instruction is executed by the CCore. If a task is running it is terminated. If the CCore is at Security Violation Stop state as a result of a force security violation, setting FSV bit, it exits this state and transitions to the STOP state. If the CCore is at Security Violation Run (SEVR) or Initialization Run (INIR) or STOP state, setting MRST has no effect. The memory mapped registers, IMEM and DMEM, are not affected by this bit.<br>**NOTE:** MRST must not set along with FSV or when FSV is already set.<br>0b - no action<br>1b - reset |
| 14<br>— | Reserved |
| 13-12<br>RSTSM4 | Reset Semaphore<br>The Reset Semaphore field is used to release a semaphore lock. In order to clear the semaphore lock, thus releasing the semaphore, the master must have Secure and Priority access attributes and write to this bit field the sequence: "01" followed by "10". After completing this sequence the semaphore is released and these bits are read as "00". If the previously described write sequence is not executed the value on these bits correspond to the latest written value. Thus reading "00" after the "01","10" write sequence indicates that the semaphore was released. If the master does not have Secure and Privilege attributes these bits will not change value and a transfer error is indicated. Note that this procedure can be executed by any master with Secure and Privilege attributes, regardless of the master ID and DID.<br>00b - Idle state<br>01b - Wait for second write<br>10b - Clears semaphore if previous state was "01"<br>11b - Reserved |
| 11-8<br>— | Reserved |
| 7<br>TCIE | Task completion with no error interrupt enable<br>This bit enables the CAU3 to issue an interrupt request in case a task was completed with no error, which means a STOP instruction is executed and TKCS= "1001".<br>0b - Disables task completion with no error to generate an interrupt request<br>1b - Enables task completion with no error to generate an interrupt request |
| 6<br>SVIE | Security Violation Interrupt Enable<br>This bit enables the crypto core to issue an error interrupt request in case a security violation occurs.<br>0b - Security violation interruption is not enabled<br>1b - Security violation interruption is enabled |
| 5<br>DIADIE | DMEM Illegal Address Interrupt Enable<br>This bit enables the crypto core to issue an interrupt request in case of an illegal address is used for a DMEM access.<br>0b - DMEM illegal address interruption is not enabled<br>1b - DMEM illegal address interruption is enabled |
| 4<br>IIADIE | IMEM Illegal Address Interrupt Enable<br>This bit enables the crypto core to issue an error interrupt request in case an illegal address is used for an IMEM access. If the origin of the access is the IPS bus, then a transfer error is returned independent of the state of this enable bit.<br>0b - IMEM illegal address interruption is not enabled<br>1b - IMEM illegal address interruption is enabled |
| 3<br>ASREIE | AHB Slave Response Error Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This bit enables the crypto core to issue an error interrupt request in case an AHB slave response error is detected (HRESP != 0) in an AHB access.<br>0b - AHB slave response error interruption is not enabled<br>1b - AHB slave response error interruption is enabled |
| 2<br>— | Reserved |
| 1<br>ILLIE | Illegal Instruction Interrupt Enable<br>This bit enables the crypto core to issue an error interrupt request when an illegal instruction is detected.<br>0b - Illegal instruction interrupt requests are disabled<br>1b - illegal Instruction interrupt requests are enabled |
| 0<br>TCSEIE | Task completion with software error interrupt enable<br>This bit enables the CAU3 to issue an interrupt request in case a task was completed with software indicated error, which means a STOP instruction is executed and TKCS= "1010".<br>0b - Disables task completion with software error to generate an interrupt request<br>1b - Enables task completion with software error to generate an interrupt request |

## 45.5.5 Status Register (SR)

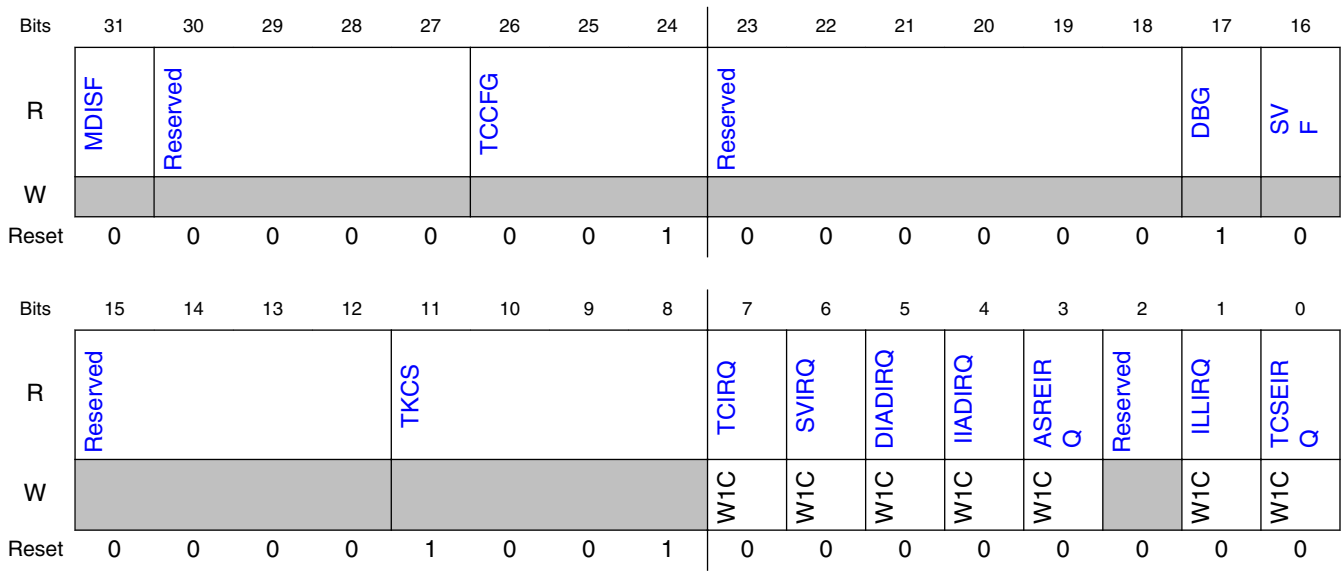
### 45.5.5.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 45.5.5.2 Function

This register contains the Status bits of the CAU3. In low power mode i.e. MDISF= 1, reads are allowed but writes are not effective and are terminated with error.

### 45.5.5.3 Diagram




### 45.5.5.4 Fields

| Field | Function |
|---|---|
| 31<br>MDISF | Module disable flag<br>This Flag indicates that the CCore is in low power mode, meaning that the clocks to the module are allowed to be gated off. This flag is set due to the MDIS bit in the control register being set and the CCore being at STOP state or when the IPG STOP request is acknowledged. Note that immediately out of reset the CryptoCore is running the initialization routine, thus this bit remains 0x0 until that routine ends, when it transitions to 0x1 indicating the low power mode.<br>0b - CCore is not in low power mode<br>1b - CCore is in low power mode |
| 30-27<br>— | Reserved |
| 26-24<br>TCCFG | Task completion configuration<br>The Task Completion Configuration shows the selected task completion status for the task being executed. The value in this field is a copy of the CMD.CMD[2:0] field if CMD.CMD[2:0]!=0. Else it is a copy of the CR.DTCCFG field. It is updated when a task starts execution.<br>000b - No action<br>001b - Assert an interrupt request<br>010b - Assert the Event Completion Signal<br>100b - Issue a DMA request |
| 23-18<br>— | Reserved |
| 17<br>DBG | Debug mode<br>This flag indicates that CAU3 is in debug mode. The debug mode is active out of reset and remains active until DBGCSR[DDBG]= 1, when it is disabled. Since the access to the DBGCSR debug register is |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | blocked out of debug mode, this flag provides the information to the host controller that the CAU3 debug features and access to debug registers are not blocked.<br>0b - CAU3 is not in debug mode<br>1b - CAU3 is in debug mode |
| 16<br>SVF | Security violation flag<br>The security violation flag, if set, indicates that the external SoC security violation signal, input to the CAU3, was asserted during at least 2 clock cycles, or that the force security violation bit, CR[FSV], was set . This flag is cleared only by a hardware reset if an SoC security violation occurred. In case of a forced security violation it can also be cleared by setting the software reset bit CR[MRST], but the security violation routine is not aborted in this scenario.<br>0b - SoC security violation is not asserted<br>1b - SoC security violation was asserted |
| 15-12<br>— | Reserved |
| 11-8<br>TKCS | Task completion status<br>The task Completion Status field signals to the software the current status of the CAU3 core. The low order three bits [2:0] are sourced from the STOP instruction operand, TKCS[3:0] = {1, STOP operand[2:0]} or from the HALT instruction imm16 data in debug mode TKCS[3:0] = {0, HALT imm16[2:0]}. Values of this field other than the shown below are reserved.<br>0000b - Initialization RUN<br>0001b - Running<br>0010b - Debug Halted<br>1001b - Stop - Error Free<br>1010b - Stop - Error<br>1110b - Stop - Security Violation, assert security violation output signal and set SVIRQ<br>1111b - Stop - Security Violation and set SVIRQ |
| 7<br>TCIRQ | Task completion with no error interrupt request<br>This flag indicates the end of a task execution with no error. It is set due to a STOP instruction execution and SR[TKCS]= "1001" and SR[TCCFG]= "001". It is cleared by the Host CPU application software by writing 1 to this bit.<br>0b - Task not finished or finished with error<br>1b - Task execution finished with no error |
| 6<br>SVIRQ | Security violation interrupt request<br>The Security Violation Interrupt Request indicates that there was an attempt to break the security measures of the system. This flag is set by a task with the completion status (TKCS) indicating the codes "1110" or "1111". In particular it is set by the Security Violation routine which is automaticaly called when the system wide security violation indication is active, also when the CR[FSV] is asserted or when there is an attempt to access the CAU3 Private DRAM content with the debug mode disabled DBGCSR[DDBGMC] = 1 or DBGCSR[DDBG] = 1. These routines ends with a STOP or STOPI instruction with operand [2:0]= "111" or "110". See Security Violation for details.<br>0b - No security violation<br>1b - Security violation |
| 5<br>DIADIRQ | DMEM illegal access interrupt request<br>The DMEM illegal access interrupt request is set when the CAU3 software executes an access (read or write) to an address of the DMEM which is greater than the maximum DMEM address, if an unaligned address is used, or when a 64-bit access is done to system memory region. In this case the CAU3 halts the software execution and enters the STOP state with a completion error indication. This flag is not set in case the Host CPU software access the DMEM through the IPS interface.<br>0b - no illegal address<br>1b - illegal address |
| 4 | IMEM Illegal address interrupt request |

*Table continues on the next page...*

| Field | Function |
|---|---|
| IIADIRQ | The IMEM Illegal Address Interrupt Request indicates if the CAU3 software tried to execute an instruction beyond the maximum allowed address of the IMEM RAM. In this case the CAU3 halts the software execution and enters the STOP state with a completion error indication. This flag is not set in case the Host CPU software access the IMEM through the IPS interface.<br>0b - no error<br>1b - illegal IMEM address detected |
| 3<br>ASREIRQ | AHB slave response error interrupt Request<br>The AHB Slave Response Error Interrupt Request indicates that the HRESP AHB Bus signal was set during an AHB access initiated by the CCore. In this case the CAU3 halts the software execution and enters the STOP state with a completion error indication. This bit is not set if the AHB access was initiated by the Host CPU through the IPS bus.<br>0b - no error<br>1b - AHB slave response error detected |
| 2<br>— | Reserved. |
| 1<br>ILLIRQ | Illegal instruction interrupt request<br>The Illegal Instruction Interrupt request is asserted when an illegal instruction is executed. This illegal instruction is not actually executed and the CAU3 core halts at (before) this instruction and enters the STOP state with a completion error indication.<br>0b - no error<br>1b - illegal instruction detected |
| 0<br>TCSEIRQ | Task completion with software error interrupt request<br>This flag indicates the end of a task execution that terminated with an error condition. This error is referred to as a software indicated error. It is set due to a STOP instruction execution with SR[TKCS]= "1010". It is cleared by the Host CPU application software by writing 1 to this bit.<br>0b - Task not finished or finished with no software error<br>1b - Task execution finished with software error |

## 45.5.6 Debug Control/Status Register (DBGCSR)

### 45.5.6.1 Offset

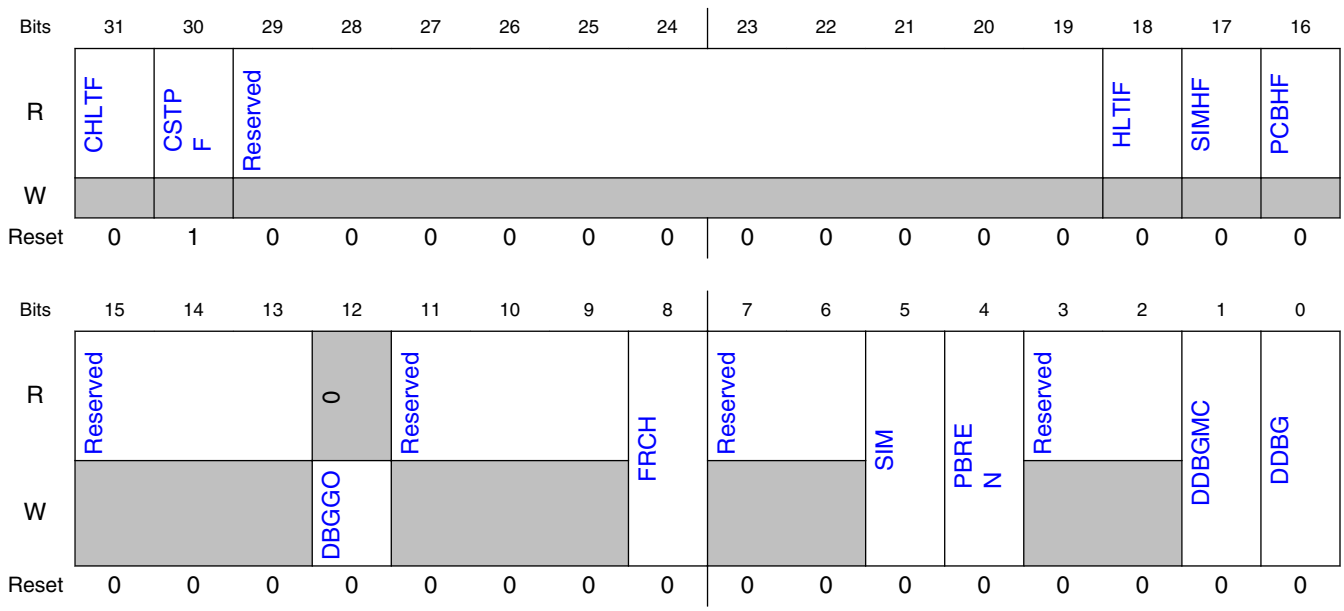
| Register | Offset |
|---|---|
| DBGCSR | 20h |

### 45.5.6.2 Function

This register contains the Debug Control and Status bits. These bits control the operation of the CAU3 CryptoCore in debug mode. This mode of operation provides software debug capabilities which means it allows access to the local private IMEM and DMEM as well as special configuration options in the CryptoCore. This mode does not provide

protection for secure information such as Cryptographic Keys. Any attempt to access this register out of debug mode, DBGCSR[DDBG] = 1, is not allowed and is terminated with error, therefore setting DBGCSR[DDBG] locks CAU3 out of debug mode. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1, or if the CryptoCore is running the initialization routine, i.e. it is in the initialization run state, when TKCS= '0000'.

## 45.5.6.3 Diagram




## 45.5.6.4 Fields

| Field | Function |
|---|---|
| 31<br>CHLTF | CryptoCore is Halted Status Flag<br>This flag bit indicates that the CryptoCore is halted.<br>0b - CryptoCore is not halted<br>1b - CryptoCore is halted |
| 30<br>CSTPF | CryptoCore is Stopped Status Flag<br>This flag bit indicates if the CryptoCore is stopped<br>0b - CryptoCore is not stopped<br>1b - CryptoCore is stopped |
| 29-19<br>— | Reserved |
| 18<br>HLTIF | CryptoCore is Halted due to HALT Instruction |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This flag bit indicates if the CryptoCore is halted due to a HALT instruction execution (also known as software breakpoint).<br>0b - CryptoCore is not in software breakpoint<br>1b - CryptoCore is in software breakpoint |
| 17<br>SIMHF | CryptoCore is Halted due to Single Instruction Step<br>This bit indicates if the CryptoCore is halted due to a single step execution in debug mode.<br>0b - CryptoCore is not in a single step halt<br>1b - CryptoCore is in a single step halt |
| 16<br>PCBHF | CryptoCore is Halted due to Hardware Breakpoint<br>This flag bit indicates that the CryptoCore is halted due to a hardware breakpoint. The hardware breakpoint is caused by a match between the Program Counter (PC) and the DBGPBR[PCBKPT] value. The instruction has not yet been executed.<br>0b - CryptoCore is not halted due to a hardware breakpoint<br>1b - CryptoCore is halted due to a hardware breakpoint |
| 15-13<br>— | Reserved |
| 12<br>DBGGO | Debug Go<br>This bit controls if the CryptoCore exits Halt state and resumes program execution. This bit is always read as zero.<br>**NOTE:** DBGGO must not set along with FRCH or when FRCH is already set.<br>**NOTE:** For single step execution, make sure SIM=1 before setting DBGGO.<br>0b - No action<br>1b - Resume program execution |
| 11-9<br>— | Reserved |
| 8<br>FRCH | Force Debug Halt<br>This bit forces the CryptoCore to enter Halt state. The processor halts instruction execution at the next instruction boundary.<br>0b - Halt state not forced<br>1b - Force halt state |
| 7-6<br>— | Reserved |
| 5<br>SIM | Single Instruction Mode<br>After HALT state is reached and single instruction is selected, SIM=1, and DBGGO is set, the CryptoCore executes one instruction and then enter HALT state again. Normal execution is resumed by clearing SIM and setting DBGGO bit.<br>**NOTE:** SIM only takes effective when the CCore is in HALT state and DBGGO is set.<br>0b - Single instruction mode is disabled<br>1b - Single instruction mode is enabled |
| 4<br>PBREN | PC Breakpoint Register Enable<br>This bit enables the PC breakpoint register. When set and the CryptoCore's program counter matches the value in the DBGPBR register, the CryptoCore halts execution and sets the PCBHF indicator.<br>**NOTE:** PBREN should only be set or cleared in STOP or HALT states.<br>0b - PC breakpoint register (DBGPBR) is disabled<br>1b - PC breakpoint register (DBGPBR) is enabled |
| 3-2<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 1<br>DDBGMC | Disable Debug Memory Commands<br>This control bit disables IPS access to the local, private IMEM and DMEM. Any attempt to access these memories with this bit set has no effect and returns a transfer error. In particular, attempted accesses of the DBGMCMD, DBGMADR and DBGMDR registers are disabled when this indicator is set and are error terminated. This bit is sticky, meaning that once set to logic 1 it cannot be cleared by a subsequent write access. Only a hardware reset clears this bit. Even though not restricted by the hardware, this bit should only be changed when the CCore is in STOP state.<br>0b - IPS access to IMEM and DMEM are enabled<br>1b - IPS access to IMEM and DMEM are disabled |
| 0<br>DDBG | Debug Disable<br>This control bit, if set, disables all debug operations of the CAU3. If set to logic 1 it blocks the accesses to debug registers DBGCSR, DBGPBR, DBGMCMD, DBGMADR and DBGMDR. Attempts to access those registers when DBGCSR[DDBG] = 1 are terminated with an error. Even though not restricted by the hardware, this bit should only be changed when the CCore is in STOP state. This bit is sticky, meaning that once set to logic 1 it cannot be cleared since the DBGCSR register becomes unaccessible.<br>The CAU3's debug and non-debug modes of operation are defined by the combined DDBG and DDBGMC bits:<br>00 - debug mode: accesses for DBGMCMD, DBGMADR and DBGMDR registers are allowed;<br>01 - debug mode: accesses are blocked for DBGMCMD, DBGMADR and DBGMDR registers;<br>1x - non-debug: accesses are blocked for DBGMCMD, DBGMADR, DBGMDR, DBGPBR and DBGCSR registers<br>In non-debug mode PC breakpoints are disabled and HALT instructions are treated as invalid instructions. See section Debug Resources for details.<br>0b - debug is enabled<br>1b - debug is disabled |

## 45.5.7 Debug PC Breakpoint Register (DBGPBR)

### 45.5.7.1 Offset

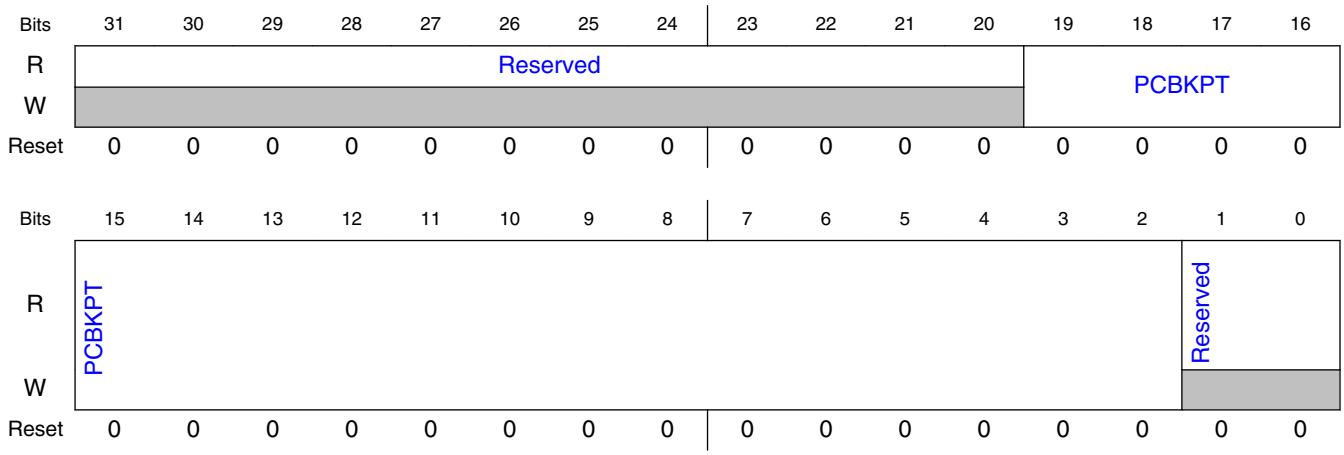
| Register | Offset |
|---|---|
| DBGPBR | 24h |

### 45.5.7.2 Function

This register contains the Debug PC Breakpoint. If enabled by the DBGCSR[PBREN] control bit, a match between the Program Counter (PC) and PCBKPT forces the CryptoCore into halt state. The breakpoint recognition and subsequent entry into halt state occurs after the previous instruction has completed execution and before the

breakpointed instruction executes. Write accesses are not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1.

### 45.5.7.3 Diagram




### 45.5.7.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-2<br>PCBKPT | PC Breakpoint<br>This bit field indicates the PC word address where a breakpoint, if enabled, occurs.<br>**NOTE:** PCBKPT should only be changed in STOP or HALT states. |
| 1-0<br>— | Reserved |

## 45.5.8 Debug Memory Command Register (DBGMCMD)

### 45.5.8.1 Offset

| Register | Offset |
|---|---|
| DBGMCMD | 30h |

### 45.5.8.2 Function

This register contains the Memory Debug Command. This command is to be used for indirect read/write accesses to the local private IMEM and DMEM. After configuring this register and DBGMADR with the address of the target memory block, the host performs writes or reads to the DBGDR register. Read or write accesses are not allowed if DBGCSR[DDBG]= 1 or DBGCSR[DDBGMC]= 1 or the CAU3 module is in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. Particularly if {DDBG,DDBGMC}!= "00" accesses to this register are interpreted as a security violation, see section Security Violation for details.

### 45.5.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 0 | Reserved | BV | 1 | IA | Reserved | DM | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.8.4 Fields

| Field | Function |
|---|---|
| 31<br>R_1 | Read always as 1 |
| 30<br>R_0 | Read always as 0 |
| 29<br>— | Reserved |
| 28<br>BV | Byte Reversal Control<br>For accesses to DMEM, this bit controls if the read or write data has its bytes reversed during the accesses. Byte reversal means that byte[0] is swapped with byte[3] and byte [1] is swapped with byte[2]. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - DMEM bytes are not reversed<br>1b - DMEM bytes are reversed |
| 27<br>Rb_1 | Read always as 1 |
| 26<br>IA | Increment Address<br>This bit controls if the address in the DBGMADR register is automatically incremented after the access to the memory is done.<br>0b - Address is not incremented<br>1b - Address is incremented after the access |
| 25<br>— | Reserved |
| 24<br>DM | Instruction/Data Memory Selection<br>This bit selects the memory (IMEM or DMEM) for the access.<br>0b - IMEM is selected<br>1b - DMEM is selected |
| 23-0<br>— | Reserved |

## 45.5.9 Debug Memory Address Register (DBGMADR)

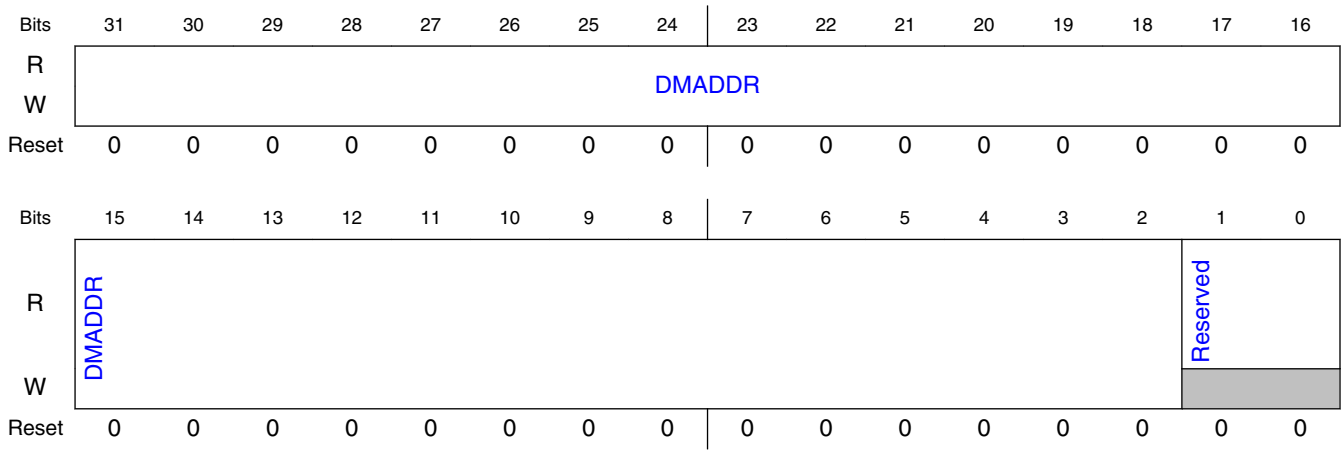
### 45.5.9.1 Offset

| Register | Offset |
|---|---|
| DBGMADR | 34h |

### 45.5.9.2 Function

This register contains the memory word address to be used for debug accesses. This address may be routed to the local private IMEM or DMEM depending on the DBGMCMD[DM] bit. Additionally, data accesses can also be made to the system RAM based on the address. The address is forced to be aligned on a 0-mod-4 byte boundary to match the access size of the CAU3. Read or write accesses are not allowed if DBGCSR[DDBG]=1 or DBGCSR[DDBGMC]= 1 or the CAU3 module is in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. Particularly if {DDBG,DDBGMC}!= "00" accesses to this register are interpreted as a security violation, see section Security Violation for details.

### 45.5.9.3 Diagram




### 45.5.9.4 Fields

| Field | Function |
|---|---|
| 31-2<br>DMADDR | Debug Memory Address<br>This address is used for write and read accesses to/from IMEM and DMEM as defined by the DBGMCMD[DM] control bit. If DBGMCMD[IA] is set then DMADDR is incremented by 1 after the access is done. If this address is used to access the DMEM.AHB space, the address range from 0x4000_0000 up to 0x5FFF_FFFF is blocked, meaning the access is terminated with an error. This restriction prevents IPS accesses of peripheral address space, which includes CAU3 address space, through the AHB master interface.<br>**NOTE:** DMADDR should only be changed in STOP or HALT states. |
| 1-0<br>— | Reserved |

## 45.5.10 Debug Memory Data Register (DBGMDR)

### 45.5.10.1 Offset

| Register | Offset |
|---|---|
| DBGMDR | 38h |

### 45.5.10.2 Function

This register contains the data related to the debug memory access. This register does not physically exist in the CAU3 module. Writes to this register are mapped to the memory location specified by DBGMCMD and DBGMADR registers. Reads of this register return the data from the memory address specified by the same two registers. Read or write accesses are not allowed if DBGCSR[DDBG]=1 or DBGCSR[DDBGMC]=1, or the CAU3 module is in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. Particularly if {DDBG,DDBGMC}!= "00" accesses to this register are interpreted as a security violation, see section Security Violation for details.

### 45.5.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.10.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DMDATA | Debug Memory Data |

## 45.5.11 Semaphore Register (SEMA4)

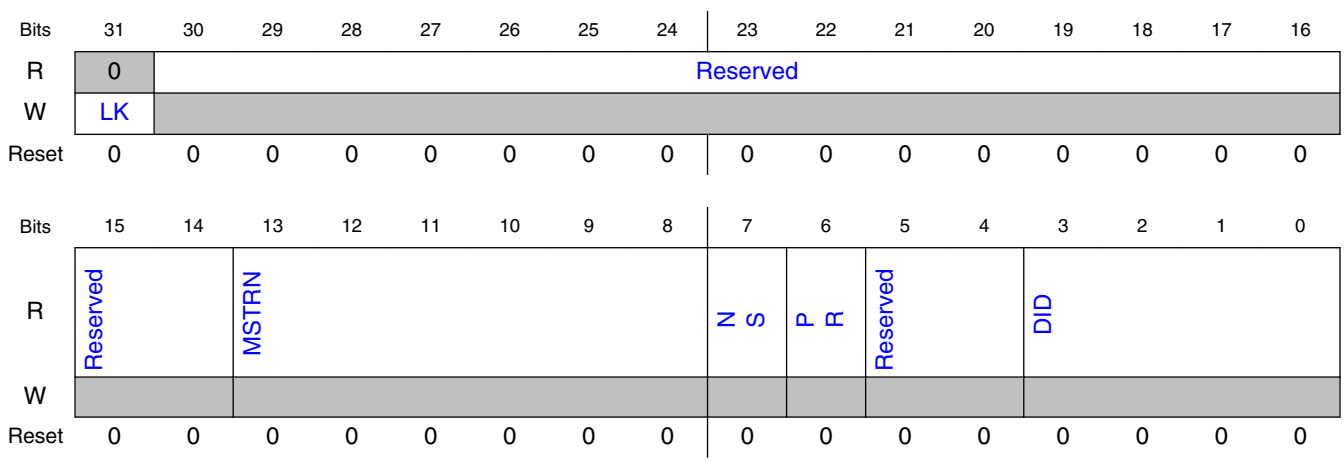
### 45.5.11.1 Offset

| Register | Offset |
|---|---|
| SEMA4 | F0h |

### 45.5.11.2 Function

The Semaphore Register is used when multiple host processors or associated bus mastering modules, for example, a DMA, need to use the shared CAU3 resources. The Semaphore Register is used by the software to take ownership of the CAU3. After writing 1 to the LK bit indicating that a host is trying to take ownership of the semaphore, the Semaphore Ownership Register, SMOWNR, can be read to verify that master is indeed the owner of the semaphore. In order to release the semaphore, the semaphore owner master writes 0 to the LK bit. Once semaphore ownership has been established, only host interface accesses from the owning master are allowed. Attempted accesses from any other master are blocked but are NOT terminated with an error. The MSTRN, NS, PR, DID fields are only visible by a secure, privileged bus master, else these fields are returned as zeros. In addition, accesses to this register are only allowed in STOP CCore state. Any attempt to write this register in a different CryptoCore state returns a transfer error. See section Semaphore Operation for details about the semaphore operation.

### 45.5.11.3 Diagram




### 45.5.11.4 Fields

| Field | Function |
|---|---|
| 31<br>LK | Semaphore Lock and Release Control |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This bit is write only and always reads as zero. If written as logic 1, this bit requests the locking of the semaphore for the accessing master. Writing logic zero means the master is releasing the semaphore, but only if it is the owning master.<br>0b - Semaphore release<br>1b - Semaphore lock |
| 30-14<br>— | Reserved |
| 13-8<br>MSTRN | Master Number of Locked Semaphore Owner<br>This bit field indicates the attribute Master Number of the locked semaphore. It is only visible for accesses with secure and privilege attributes, otherwise they are read as zero. |
| 7<br>NS | Non Secure Attribute of the Locked Semaphore Owner<br>This bit indicates the attribute Non Secure of the locked semaphore. It is only visible by a master with secure and privilege attributes, otherwise it is read as zero.<br>0b - If semaphore is locked, owner is operating in secure mode<br>1b - If semaphore is locked, owner is operating in nonsecure mode |
| 6<br>PR | Privilege Attribute of Locked Semaphore Owner<br>This bit indicates the attribute Privilege of the locked semaphore. It is only visible for accesses with secure and privilege attributes, otherwise it is read as zero.<br>0b - If semaphore is locked, then owner is operating in user mode<br>1b - If semaphore is locked, then owner is operating in privileged mode |
| 5-4<br>— | Reserved |
| 3-0<br>DID | Domain ID of Locked Semaphore Owner<br>This bit field indicates the attribute ID of the locked semaphore. It is only visible for accesses with secure and privilege attributes, otherwise they are read as zero. |

## 45.5.12 Semaphore Ownership Register (SMOWNR)

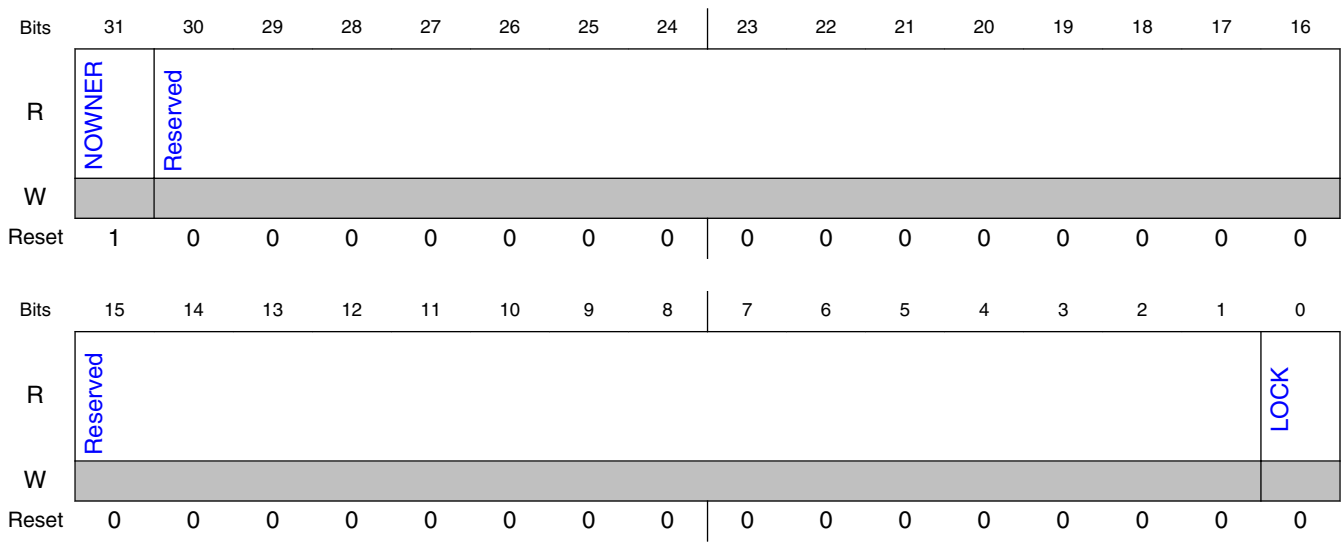
### 45.5.12.1 Offset

| Register | Offset |
|---|---|
| SMOWNR | F4h |

## 45.5.12.2 Function

This read-only register indicates if the lock request was successful on locking the CAU3's semaphore. After attempting to claim ownership of the semaphore by seting the SEMA4[LK] bit, the host device reads the ownership register. The read data returns the semaphore state with 3 distinct values, allowing an easy single conditional branch resolution.See section Semaphore Operation for details about the semaphore operation.

## 45.5.12.3 Diagram




## 45.5.12.4 Fields

| Field | Function |
|---|---|
| 31<br>NOWNER | Semaphore Ownership<br>This bit, if set, indicates that the host making the current read access to this register is NOT the owner of the semaphore.<br>0b - The host making the current read access is the semaphore owner<br>1b - The host making the current read access is NOT the semaphore owner |
| 30-1<br>— | Reserved |
| 0<br>LOCK | Semaphore Locked<br>The LOCK bit, if set, indicates that the semaphore is locked.<br>0b - Semaphore not locked<br>1b - Semaphore locked |

## 45.5.13 Address Remap Register (ARR)

### 45.5.13.1 Offset

| Register | Offset |
|---|---|
| ARR | FCh |

### 45.5.13.2 Function

The Address Remap Register contains a register list for address remapping of host accesses to the CryptoCore's general-purpose registers. Once programmed with any non-zero data it becomes active. Each non-zero bit in the register corresponds to one of the CC_Rn registers, such that bit 0 corresponds to R0, bit 1 corresponds to R1 and so forth up to bit 31 corresponding to R31. For accesses in the CC_Rn (180h - 1FCh) address space with at least one non-zero ARR bit, instead of the register referenced by the actual bus address, a remapped address is generated based on the least significant non-zero bit in ARR. After the access is done, the corresponding ARR bit is cleared. The next access uses the next non-zero LSB in ARR for the remapping. The remapping is done while ARR is non-zero. Read or write accesses are not allowed if the CAU3 module is in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1.

### 45.5.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ARRL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ARRL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.13.4 Fields

| Field | Function |
|---|---|
| 31-0<br>ARRL | Address Remap Register List<br>ARRL[i] corresponds to CC_R[i]. |

## 45.5.14 CryptoCore General Purpose Registers (CC_R0 - CC_R 29)

### 45.5.14.1 Offset

For a = 0 to 29:

| Register | Offset |
|---|---|
| CC_Ra | 180h + (a × 4h) |

### 45.5.14.2 Function

These registers are mapped to the CryptoCore R[0:29] registers. Reads and writes are only allowed in STOP or HALT states . Any attempt to read or write this register in a different CryptoCore state returns a transfer error. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. In the error case, the register write is inhibited and the register read returns zeroes.

### 45.5.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.14.4 Fields

| Field | Function |
|---|---|
| 31-0<br>R | CryptoCore general purpose register R |

## 45.5.15 General Purpose R30 (CC_R30)

### 45.5.15.1 Offset

| Register | Offset |
|---|---|
| CC_R30 | 1F8h |

### 45.5.15.2 Function

This register is mapped to the CryptoCore R30 register. Reads and writes are only allowed in STOP or HALT states. Any attempt to read or write this register in a different CryptoCore state returns a transfer error. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. In the error case, the register write is inhibited and the register read returns zeroes. This register is also used as the CryptoCore Stack Pointer.

### 45.5.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.15.4 Fields

| Field | Function |
|---|---|
| 31-0<br>SP | Stack Pointer |

## 45.5.16 General Purpose R31 (CC_R31)

### 45.5.16.1 Offset

| Register | Offset |
|---|---|
| CC_R31 | 1FCh |

### 45.5.16.2 Function

This register is mapped to the CryptoCore R31 register. Reads and writes are only allowed in STOP or HALT states. Any attempt to read or write this register in a different CryptoCore state returns a transfer error. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. In the error case, the register write is inhibited and the register read returns zeroes. This register is also used as the Core Link Register.

### 45.5.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.16.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LR | Link |

## 45.5.17 Program Counter (CC_PC)

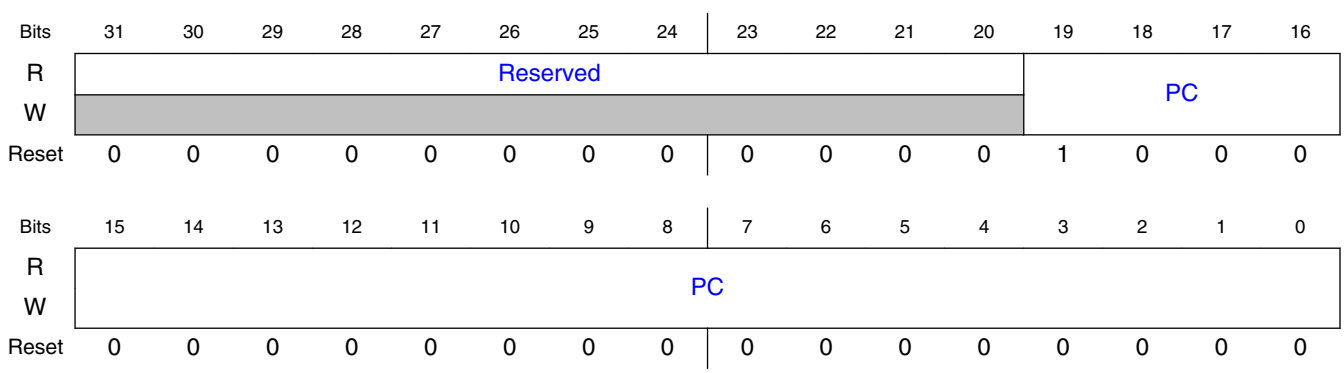
### 45.5.17.1 Offset

| Register | Offset |
|---|---|
| CC_PC | 200h |

### 45.5.17.2 Function

This register is mapped to the CryptoCore's Program Counter (PC) register. Reads and writes are only allowed in STOP or HALT states. Any attempt to read or write this register in a different CryptoCore state returns a transfer error. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. In the error case, the register write is inhibited and the register read returns zeroes.

### 45.5.17.3 Diagram



### 45.5.17.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-0<br>PC | Program Counter<br>The PC field is written prior to task execution with the appropriate index into the branch table based at address 0 of the IMEM. After task completion with a STOPPED status, the PC points to the address of the executed STOP instruction. For any type of illegal operation termination, the PC points to the instruction after, that is, PC+4 of the offending instruction. Since the CryptoCore implements a fixed 32-bit instruction length, the low-order 2 bits of the PC[19:0] field are hardwired to 00b, PC[1:0] = 00, making the PC a word address. Readings from PC[1:0] always return 00b. The PC reset value points to the IMEM.IROM base address: 0x80000. |

## 45.5.18 Start Command Register (CC_CMD)

### 45.5.18.1 Offset

| Register | Offset |
|---|---|
| CC_CMD | 204h |

### 45.5.18.2 Function

Accesses to this register are only allowed in STOP state. Any attempt to write this register in a different CryptoCore state returns a transfer error. Accesses are also not allowed in low power, SR[MDISF]= 1, or if a security violation had occurred, SR[SVF]=1. In the error case, the register write is ignored. In response to the register write, the CryptoCore performs two operations: first, it clears all pipeline state information, and second, it forces an instruction fetch to begin execution at the address programmed into the Program Counter register (CC_PC). Read accesses are terminated with an error.

### 45.5.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | CMD | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.18.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>CMD | Command<br>The Command field starts the execution of a cryptographic routine by the CryptoCore and defines the task completion configuration if it is not zero. In this case the CMD field replaces the task completion configured by the CR[DTCCFG] bit field. Attempted use of an undefined encoding produces unpredictable behavior.<br>000b - Use CR[DTCCFG] for task completion configuration<br>001b - Issue an interrupt request<br>010b - Assert Event Completion Signal<br>100b - Issue a DMA request |
| 15-0<br>— | Reserved |

## 45.5.19 Condition Flag (CC_CF)

### 45.5.19.1 Offset

| Register | Offset |
|---|---|
| CC_CF | 208h |

### 45.5.19.2 Function

This read-only register is mapped to the CryptoCore's Condition Flags. Reads are only allowed in STOP or HALT states. Any attempt to read this register in a different CryptoCore state returns a transfer error. In the error case, the register read returns zeroes.

### 45.5.19.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | N | Z | V | C |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.19.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>N | Negative flag |
| 2<br>Z | Zero flag |
| 1<br>V | Overflow flag |
| 0<br>C | Carry flag |

## 45.5.20 Mode Register (PublicKey) (MDPK)

### 45.5.20.1 Offset

| Register | Offset |
|---|---|
| MDPK | 400h |

### 45.5.20.2 Function

The Mode Register is used to define the public key operation that is being requested. The interpretation of this register is for public key operations (PKHA) only, and is described herein.

The bit assignments for the PKHA_MODE will be different depending on which of the four types of PKHA functions is being called. The four function types are: 1) Clear Memory, 2) Modular Arithmetic, 3) Elliptic Curve, and 4) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions, Table PKHA MODE: Elliptic Curve Functions, and Table PKHA MODE: copy memory functions.

### 45.5.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | ALG | | | | PKHA_MODE_MS | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | PKHA_MODE_LS | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.20.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20 | Algorithm |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ALG | This field specifies which algorithm is being selected.<br>1000b - PKHA |
| 19-16<br>PKHA_MODE_MS | PKHA_MODE most-significant 4 bits<br>The format of the PKHA_MODE field differs depending on which of the four types of PKHA functions is being executed. The four function types are: 1) Clear Memory, 2) Modular Arithmetic, 3) Elliptic Curve, and 4) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions, Table PKHA MODE: Elliptic Curve Functions, and Table PKHA MODE: copy memory functions. |
| 15-12<br>— | Reserved. |
| 11-0<br>PKHA_MODE_LS | PKHA_MODE least significant 12 bits<br>The format of the PKHA_MODE field differs depending on which of the four types of PKHA functions is being executed. The four function types are: 1) Clear Memory, 2) Modular Arithmetic, 3) Elliptic Curve, and 4) Copy Memory. Detailed descriptions of their mode formats can be found in Table PKHA MODE: clear memory function, Table PKHA MODE: Arithmetic Functions, Table PKHA MODE: Elliptic Curve Functions, and Table PKHA MODE: copy memory functions. |

## 45.5.21 Command Register (COM)

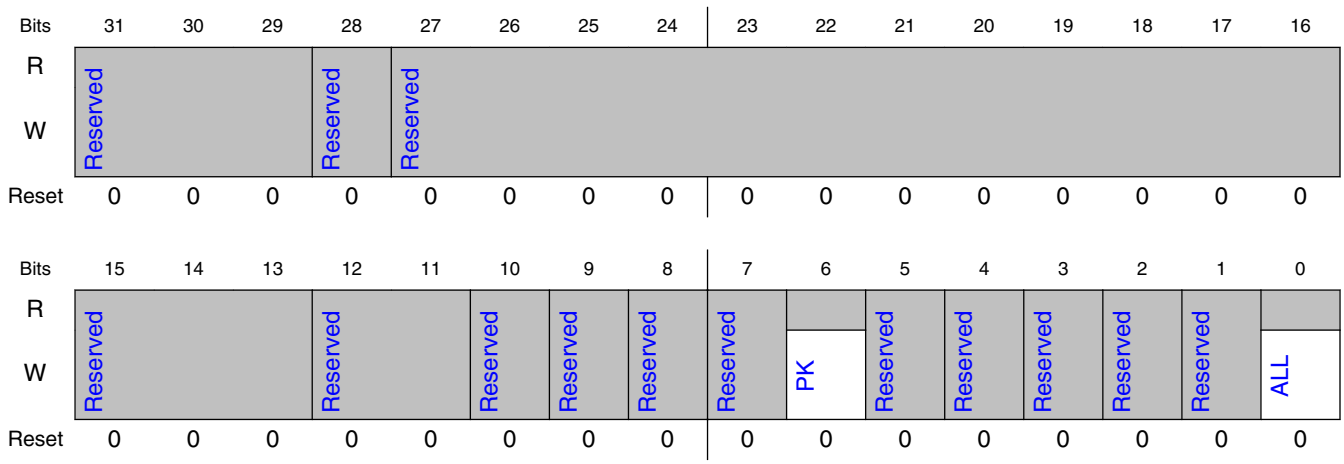
### 45.5.21.1 Offset

| Register | Offset |
|---|---|
| COM | 430h |

### 45.5.21.2 Function

The Command Register is used to send control signals to the Crypto Engines.

## 45.5.21.3 Diagram




## 45.5.21.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28<br>— | Reserved. |
| 27-13<br>— | Reserved. |
| 12-11<br>— | Reserved. |
| 10<br>— | Reserved. |
| 9<br>— | Reserved. |
| 8<br>— | Reserved. |
| 7<br>— | Reserved. |
| 6<br>PK | Reset PKHA<br>Writing a 1 to this bit resets the Public Key Hardware Accelerator.<br>0b - Do Not Reset<br>1b - Reset Public Key Hardware Accelerator |
| 5<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 4<br>— | Reserved. |
| 3<br>— | Reserved. |
| 2<br>— | Reserved. |
| 1<br>— | Reserved. |
| 0<br>ALL | Reset All Internal Logic<br>Writing to this bit will reset all PKHA accelerator engines and as well as all PKHA internal registers.<br>0b - Do Not Reset<br>1b - Reset PKHA engine and registers |

## 45.5.22 Control Register (CTL)

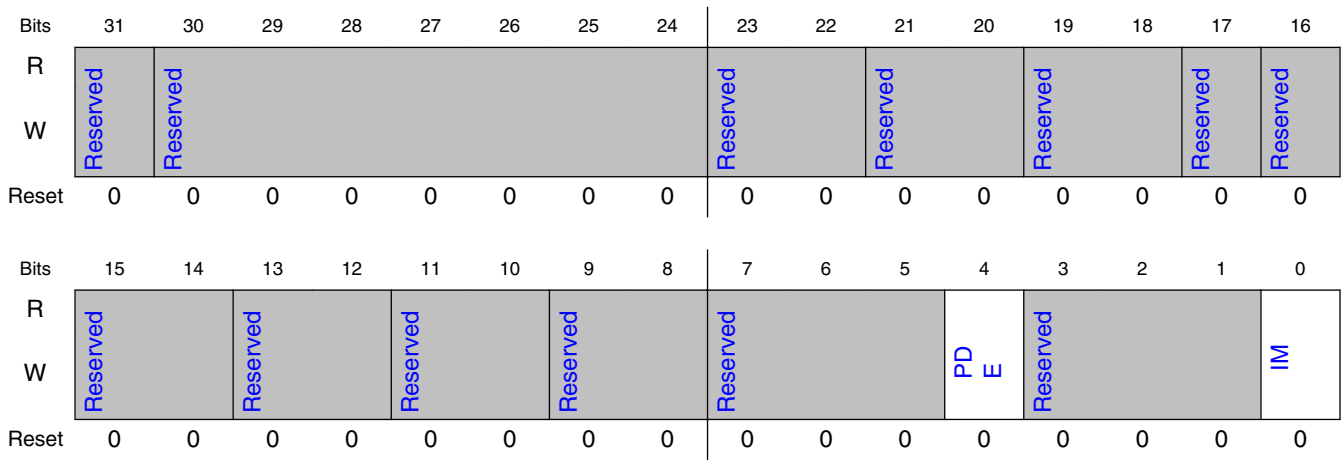
### 45.5.22.1 Offset

| Register | Offset |
|---|---|
| CTL | 434h |

### 45.5.22.2 Function

This register is used for some of the internal controls of the PKHA accelerator, such as DMA and Interrupt Mask.

## 45.5.22.3 Diagram




## 45.5.22.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>— | Reserved. |
| 23-22<br>— | Reserved. |
| 21-20<br>— | Reserved. |
| 19-18<br>— | Reserved. |
| 17<br>— | Reserved. |
| 16<br>— | Reserved. |
| 15-14<br>— | Reserved. |
| 13-12<br>— | Reserved. |
| 11-10<br>— | Reserved. |
| 9-8 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 7-5<br>— | Reserved. |
| 4<br>PDE | PKHA Register DMA Enable<br>0b - DMA Request and Done signals disabled for the PKHA Registers.<br>1b - DMA Request and Done signals enabled for the PKHA Registers. |
| 3-1<br>— | Reserved. |
| 0<br>IM | Interrupt Mask<br>This control bit masks the module interrupt request.<br>0b - Interrupt not masked.<br>1b - Interrupt masked |

## 45.5.23 Clear Written Register (CW)

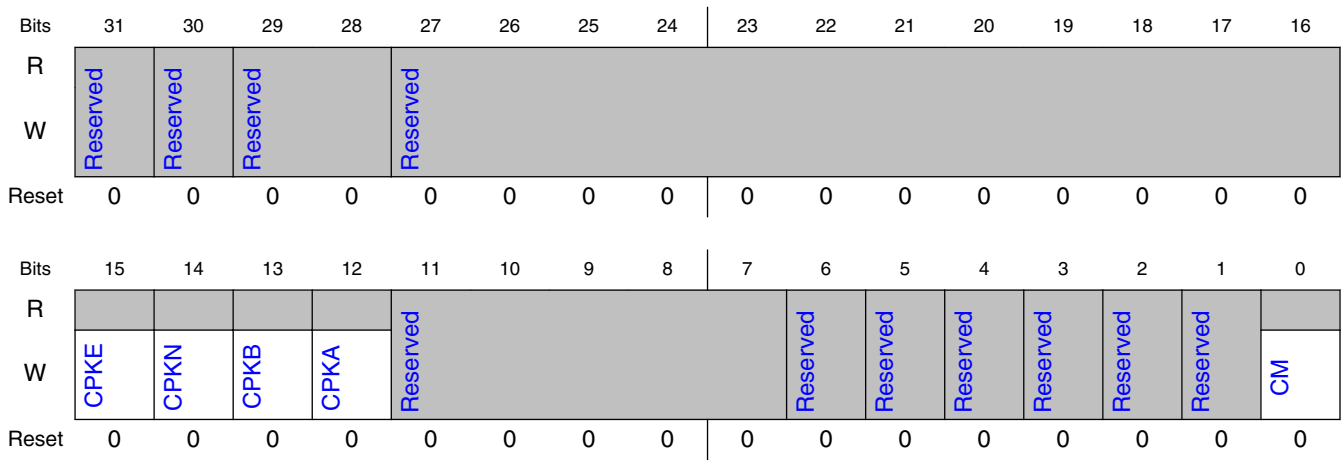
### 45.5.23.1 Offset

| Register | Offset |
|---|---|
| CW | 440h |

### 45.5.23.2 Function

The Clear Written Register is used to clear many of the internal registers. All fields of this register are self-clearing.

### 45.5.23.3 Diagram




### 45.5.23.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30<br>— | Reserved. |
| 29-28<br>— | Reserved. |
| 27-16<br>— | Reserved. |
| 15<br>CPKE | Clear the PKHA E Size Register<br>Writing a one to this bit causes the PKHA E Size Register to be cleared. |
| 14<br>CPKN | Clear the PKHA N Size Register<br>Writing a one to this bit causes the PKHA N Size Register to be cleared. |
| 13<br>CPKB | Clear the PKHA B Size Register<br>Writing a one to this bit causes the PKHA B Size Register to be cleared. |
| 12<br>CPKA | Clear the PKHA A Size Register<br>Writing a one to this bit causes the PKHA A Size Register to be cleared. |
| 11-7<br>— | Reserved. |
| 6<br>— | Reserved. |
| 5 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 4<br>— | Reserved. |
| 3<br>— | Reserved. |
| 2<br>— | Reserved. |
| 1<br>— | Reserved. |
| 0<br>CM | Clear the Mode Register<br>Writing a one to this bit causes the Mode Register to be cleared. |

## 45.5.24 Status Register (STA)

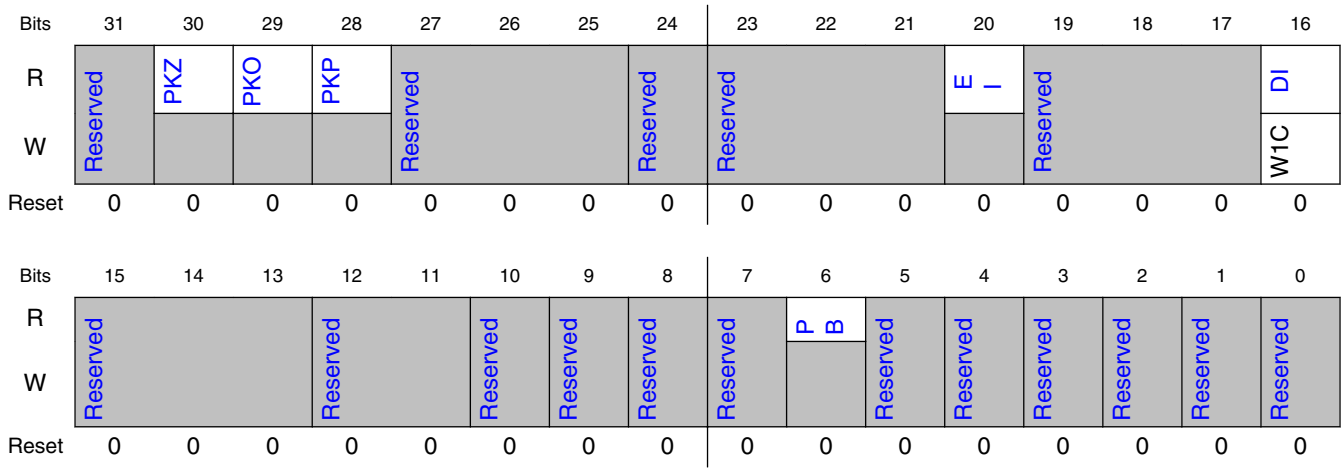
### 45.5.24.1 Offset

| Register | Offset |
|---|---|
| STA | 448h |

### 45.5.24.2 Function

The Status Register shows the status of the internal Crypto engine and its internal registers.

## 45.5.24.3 Diagram




## 45.5.24.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30<br>PKZ | Public Key Operation is Zero<br>For Finite Field operations the result of a Public Key operation is zero. For ECC operations, the result is Point at infinity. |
| 29<br>PKO | Public Key Operation is One<br>The greatest common divisor of two numbers is one (that is, the two numbers are relatively prime) for GCD operation, or PKHA result = 1 for Finite Field operations. |
| 28<br>PKP | Public Key is Prime<br>The given number is probably prime (that is, it passes the Miller-Rabin primality test). |
| 27-25<br>— | Reserved. |
| 24<br>— | Reserved. |
| 23-21<br>— | Reserved. |
| 20<br>EI | Error Interrupt<br>The Error Interrupt has been asserted. This error can only be cleared by resetting PKHA using the ALL bit in the Command Register, COM.<br>0b - Not Error.<br>1b - Error Interrupt. |
| 19-17 | Reserved. |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>—</td><td></td></tr>
<tr><td>16<br>DI</td><td>Done Interrupt<br>The Done Interrupt has been asserted.<br><table><tr><th>Value</th><th>Read</th><th>Write</th></tr><tr><td>0</td><td>No Done Interrupt</td><td>No change</td></tr><tr><td>1</td><td>Done Interrupt asserted</td><td>Clear the Done Interrupt</td></tr></table></td></tr>
<tr><td>15-13<br>—</td><td>Reserved.</td></tr>
<tr><td>12-11<br>—</td><td>Reserved.</td></tr>
<tr><td>10<br>—</td><td>Reserved.</td></tr>
<tr><td>9<br>—</td><td>Reserved.</td></tr>
<tr><td>8<br>—</td><td>Reserved.</td></tr>
<tr><td>7<br>—</td><td>Reserved.</td></tr>
<tr><td>6<br>PB</td><td>PKHA Busy<br>This bit indicates that the Public Key Hardware Accelerator is busy. The PKHA can either be busy processing data or resetting.<br>0b - PKHA Idle<br>1b - PKHA Busy.</td></tr>
<tr><td>5<br>—</td><td>Reserved.</td></tr>
<tr><td>4<br>—</td><td>Reserved.</td></tr>
<tr><td>3<br>—</td><td>Reserved.</td></tr>
<tr><td>2<br>—</td><td>Reserved.</td></tr>
<tr><td>1<br>—</td><td>Reserved.</td></tr>
<tr><td>0<br>—</td><td>Reserved.</td></tr>
</table>

# 45.5.25 Error Status Register (ESTA)

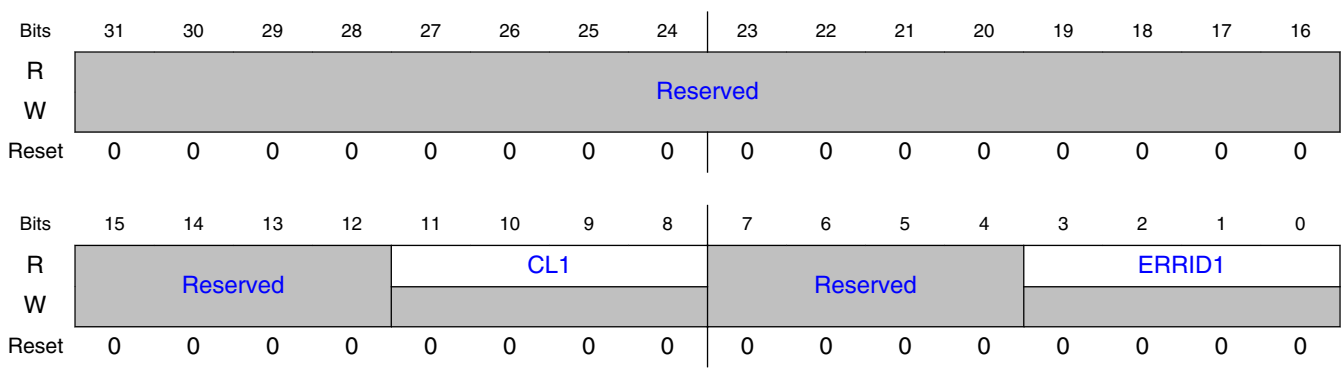
## 45.5.25.1 Offset

| Register | Offset |
|---|---|
| ESTA | 44Ch |

## 45.5.25.2 Function

The Error Register shows the status of the internal Crypto Engine and its internal registers.

## 45.5.25.3 Diagram




## 45.5.25.4 Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved. |
| 11-8<br>CL1 | algorithms<br>The algorithms field indicates which algorithm is asserting an error.<br>Others reserved<br>0000b - General Error<br>1000b - Public Key |
| 7-4 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 3-0<br>ERRID1 | Error ID 1<br>These bits indicate the type of error that was found while processing the Descriptor. The Algorithm that is associated with the error can be found in the CL1 field.<br>Others reserved.<br>0001b - Mode Error<br>0010b - PKHA N Register Size Error<br>0011b - PKHA E Register Size Error<br>0100b - PKHA A Register Size Error<br>0101b - PKHA B Register Size Error<br>0110b - PKHA C input (as contained in the PKHA B0 quadrant) is Zero<br>0111b - PKHA Divide by Zero Error<br>1000b - PKHA Modulus Even Error<br>1111b - Invalid Crypto Engine Selected |

## 45.5.26 PKHA A Size Register (PKASZ)

### 45.5.26.1 Offset

| Register | Offset |
|---|---|
| PKASZ | 480h |

### 45.5.26.2 Function

The PKHA A Size Register is used to indicate the number of bytes of valid data in the PKHA A Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA A Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

### 45.5.26.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | PKASZ | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.26.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>PKASZ | PKHA A Size<br>This is the size of the numeric value, in bytes, contained within the PKHA A Register. |

## 45.5.27 PKHA B Size Register (PKBSZ)

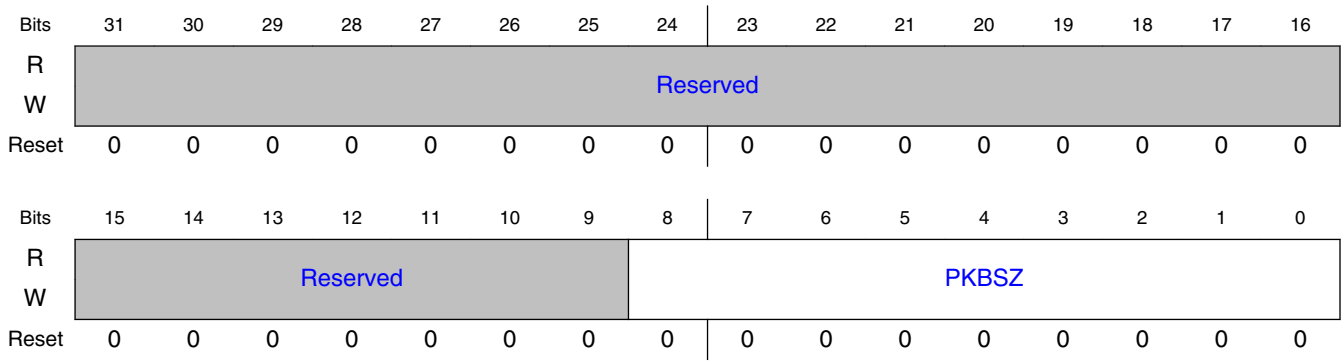
### 45.5.27.1 Offset

| Register | Offset |
|---|---|
| PKBSZ | 488h |

### 45.5.27.2 Function

The PKHA B Size Register is used to indicate the number of bytes of valid data in the PKHA B Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA B Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

### 45.5.27.3 Diagram




### 45.5.27.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>PKBSZ | PKHA B Size<br>This is the size of the numeric value, in bytes, contained within the PKHA B Register. |

## 45.5.28 PKHA N Size Register (PKNSZ)

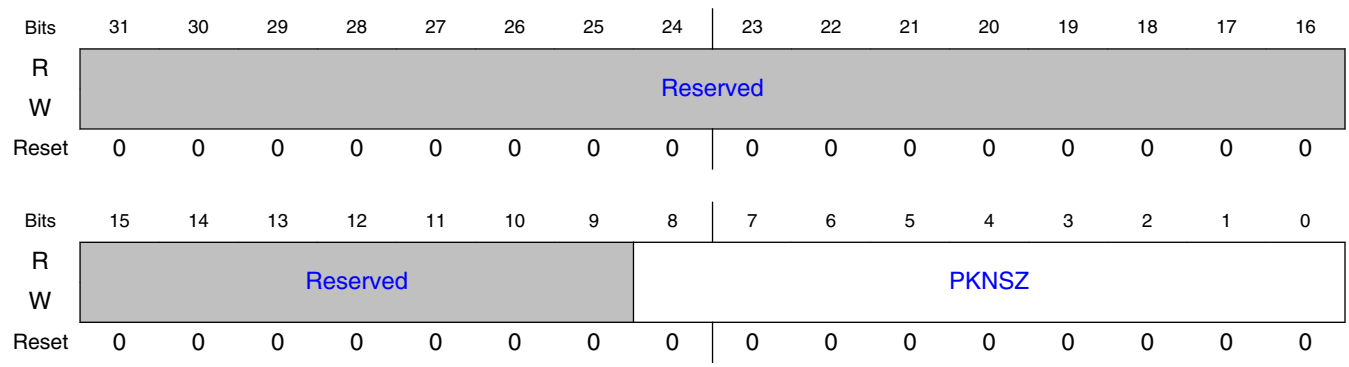
### 45.5.28.1 Offset

| Register | Offset |
|---|---|
| PKNSZ | 490h |

### 45.5.28.2 Function

The PKHA N Size Register is used to indicate the number of bytes of valid data in the PKHA N Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA N Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

### 45.5.28.3 Diagram




### 45.5.28.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>PKNSZ | PKHA N Size<br>This is the size of the numeric value, in bytes, contained within the PKHA N Register. |

## 45.5.29 PKHA E Size Register (PKESZ)

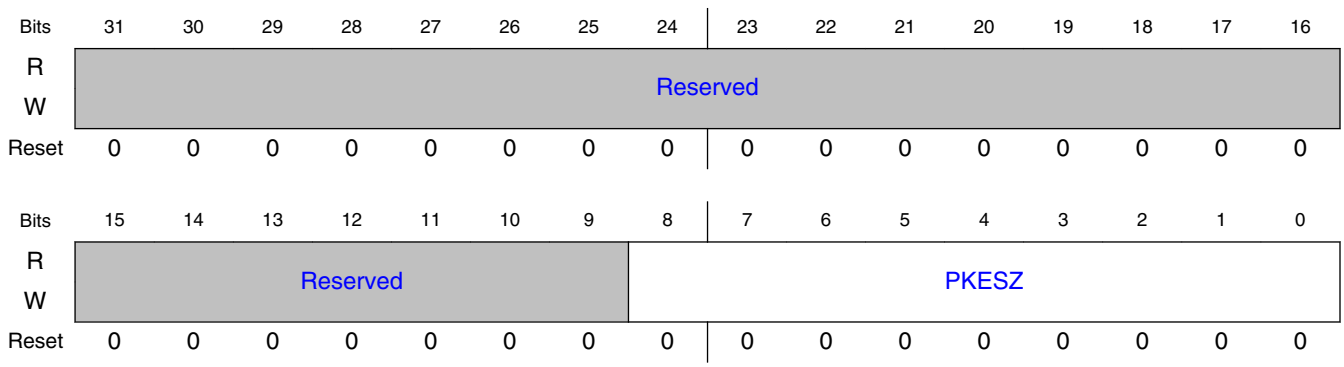
### 45.5.29.1 Offset

| Register | Offset |
|---|---|
| PKESZ | 498h |

### 45.5.29.2 Function

The PKHA E Size Register is used to indicate the number of bytes of valid data in the PKHA E Register. That is, it indicates the size, in bytes, of the numeric value contained in the PKHA E Register. Depending on the PKHA operation to be run, this register may need to be loaded prior to starting the PKHA operation. Note that some PKHA operations update this value at the completion of the operation.

### 45.5.29.3 Diagram




### 45.5.29.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>PKESZ | PKHA E Size<br>This is the size of the numeric value, in bytes, contained within the PKHA E Register. |

## 45.5.30 PKHA Revision ID 1 (PKHA_VID1)

### 45.5.30.1 Offset

| Register | Offset |
|---|---|
| PKHA_VID1 | 4F0h |

### 45.5.30.2 Function

This register contains the ID for PKHA sub-module, major and minor revision numbers.

### 45.5.30.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IP_ID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJ_REV | | | | | | | | MIN_REV | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.30.4 Fields

| Field | Function |
|---|---|
| 31-16<br>IP_ID | Hardware Revision ID |
| 15-8<br>MAJ_REV | Major Revision Number |
| 7-0<br>MIN_REV | Minor Revision Number |

## 45.5.31 PKHA Revision ID 2 (PKHA_VID2)

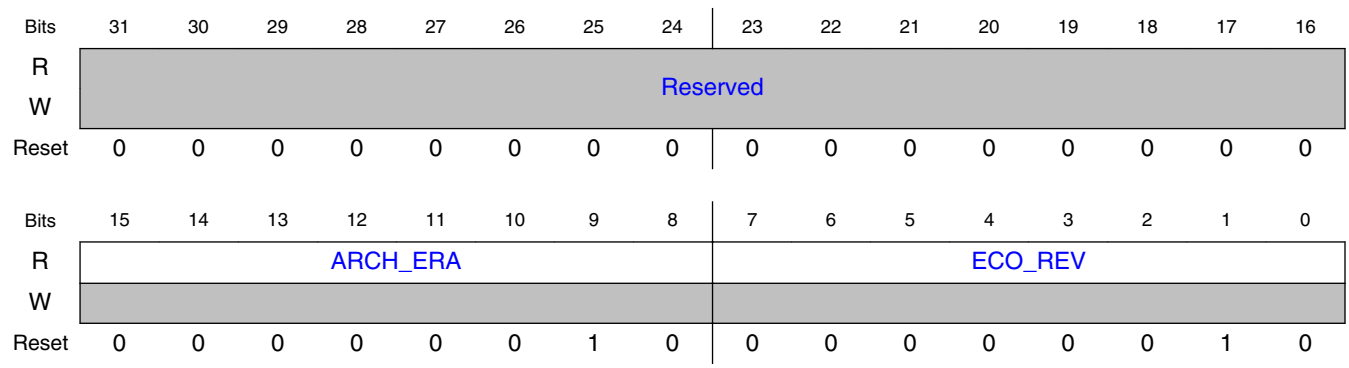
### 45.5.31.1 Offset

| Register | Offset |
|---|---|
| PKHA_VID2 | 4F4h |

### 45.5.31.2 Function

This register contains the architectural era and eco revision numbers of the PKHA sub-module.

### 45.5.31.3 Diagram




### 45.5.31.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8<br>ARCH_ERA | Architecture ERA |
| 7-0<br>ECO_REV | ECO Revision Number |

## 45.5.32 CHA Revision ID (CHA_VID)

### 45.5.32.1 Offset

| Register | Offset |
|---|---|
| CHA_VID | 4F8h |

### 45.5.32.2 Function

This register contains the Version ID and Revision Number for the CHAs contained within the PKHA sub-module.

### 45.5.32.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | PKHAVID | | | | PKHAREV | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.32.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-20<br>PKHAVID | PK Version ID |
| 19-16<br>PKHAREV | PK Revision NUmber |
| 15-0<br>— | Reserved. |

## 45.5.33 PKHA Clock Control Register (PKHA_CCR)

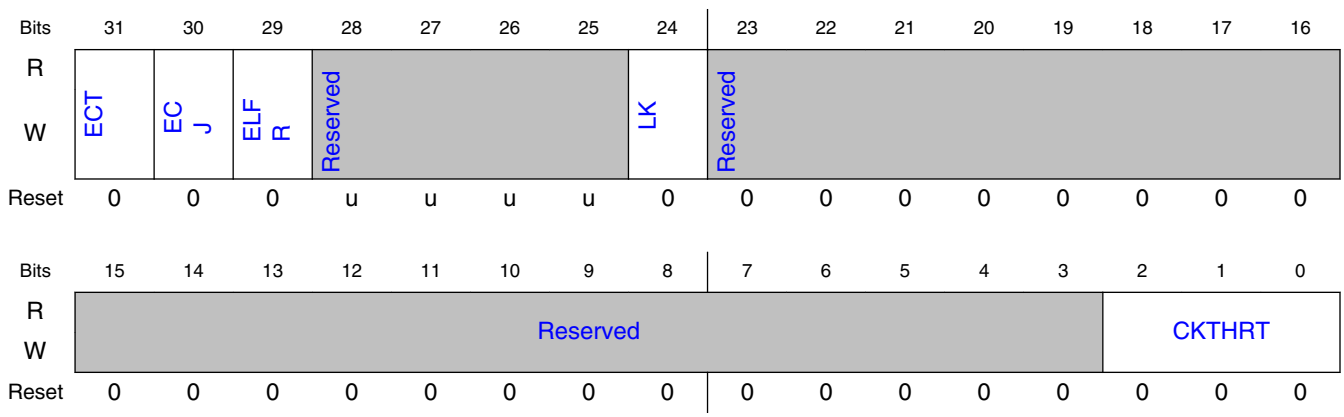
## 45.5.33.1 Offset

| Register | Offset |
|---|---|
| PKHA_CCR | 600h |

## 45.5.33.2 Function

The PKHA Clock Control register implements several PKHA clock control signals to reduce the frequency and add optional jitter to the clock input of this module.

## 45.5.33.3 Diagram




## 45.5.33.4 Fields

| Field | Function |
|---|---|
| 31<br>ECT | Enable Clock Throttle<br>The Clock Throttle Enable bit enables the PKHA clock to be divided therefore controlling the Public Key module processing speed.<br>0b - PKHA clock throttle disabled meaning that PKHA is operatiing at full speed<br>1b - PKHA clock throttle enabled |
| 30<br>ECJ | Enable Clock Jitter<br>Enable PKHA clock jitter according to CKTHRT field. This feature adds jitter to the PKHA clock, providing a hardware countermeasure against side channel attacks. Note that the jitter only applies if the clock is not set to full speed, i.e ECT = 0 or CKTHRT = 0.<br>0b - Clock Jitter is disabled<br>1b - Clock jitter is enabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29<br>ELFR | Enable Linear Feedback Shift Register<br>This control bit provides an independent control over the clocking of the LFSR. It can be used as a hardware countermeasure against side channel attacks by generating pseudo-randon background noise when ECT = 0 or ECJ = 0. In this background noise generation application the LFSR has neither impact on functionality nor PKHA throughput.<br>0b - LFSR is only enabled if ECT = 1 and ECJ = 1<br>1b - LFSR is enabled independently of ECT and ECJ |
| 28-25<br>— | Reserved. |
| 24<br>LK | Register Lock<br>The register lock bit locks the values of the other fields in this register. Once this bit is set, software cannot write to this register. A hardware reset clears the register lock bit.<br>0b - Register is unlocked<br>1b - Register is locked |
| 23-3<br>— | Reserved. |
| 2-0<br>CKTHRT | Clock Throttle selection<br>The Clock Throttle selection field selects the clock division applied to the PKHA sub-module clock input, therefore reducing the execution throughput of this module. It is intended to provide a mechanism to match the execution speed of future size/performance reduced PKHA implementations.<br>000b - PKHA clock division rate is 8/8 - full speed<br>001b - PKHA clock division rate is 1/8<br>010b - PKHA clock division rate is 2/8<br>011b - PKHA clock division rate is 3/8<br>100b - PKHA clock division rate is 4/8<br>101b - PKHA clock division rate is 5/8<br>110b - PKHA clock division rate is 6/8<br>111b - PKHA clock division rate is 7/8 |

## 45.5.34 Global Status Register (GSR)

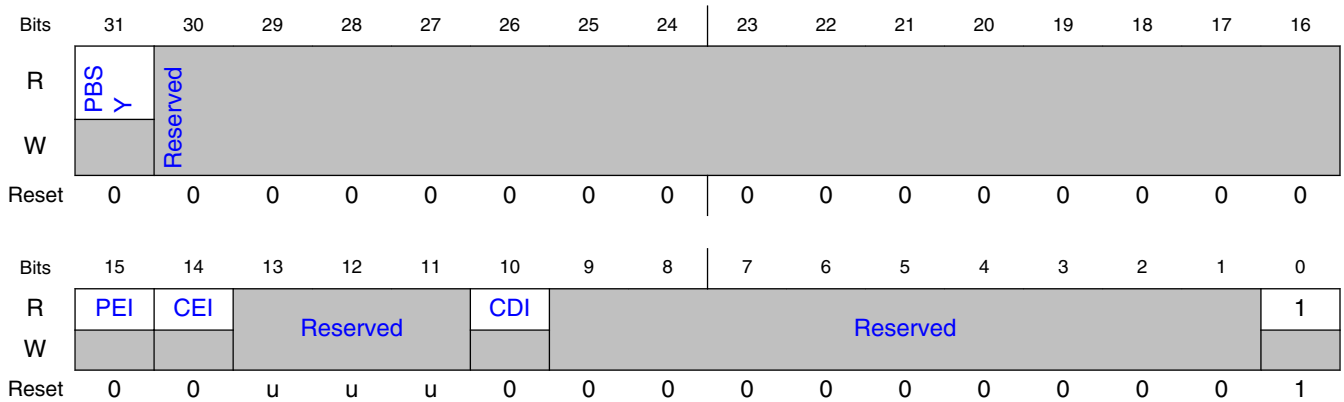
### 45.5.34.1 Offset

| Register | Offset |
|---|---|
| GSR | 604h |

### 45.5.34.2 Function

The Global Status register provides status information for the CAU3 and the public key PKHA sub-module.

## 45.5.34.3 Diagram




## 45.5.34.4 Fields

| Field | Function |
|---|---|
| 31<br>PBSY | PKHA Busy<br>The PKHA Busy bit provides status information of the PKHA sub-module, indicating it is busy executing an operation.<br>0b - PKHA not busy<br>1b - PKHA busy |
| 30-16<br>— | Reserved. |
| 15<br>PEI | PKHA Done or Error Interrupt<br>The PEI bit indicates that an interrupt occurred indicating that the PKHA module have finished the operation. The operation could have either finished with error or without error.<br>0b - PKHA interrupt did not occur<br>1b - PKHA interrupt had occurred |
| 14<br>CEI | CAU3 Error Interrupt<br>The CEI bit indicates that a CAU3 Error Interrupt had occurred.<br>0b - CAU3 Error Interrupt did not occur<br>1b - CAU3 Error Interrupt occurred |
| 13-11<br>— | Reserved. |
| 10<br>CDI | CAU3 Done Interrupt occurred<br>The CDI bit indicates when a done interrupt was issued by the CAU3.<br>0b - CAU3 Done Interrupt did not occur<br>1b - CAU3 Done Interrupt occurred |
| 9-1<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>— | Reserved. |

## 45.5.35 Clock Linear Feedback Shift Register (CKLFSR)

### 45.5.35.1 Offset

| Register | Offset |
|---|---|
| CKLFSR | 608h |

### 45.5.35.2 Function

The Clock Linear Feedback Shift register provides access to a 32-bit LFSR used to add jitter to the PKHA clock signal. Initially software must write a 32-bit non-zero value random number into this register before using it, otherwise the LFSR remains locked at 0. Additionaly software may write a non-zero ramdom number into this register at any timer.

### 45.5.35.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LFSR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LFSR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.35.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LFSR | Linear Feedback Shift Register<br>Data value for the 32-bit LFSR. |

## 45.5.36 PKHA A0 Register (PKA0_0 - PKA0_31)

### 45.5.36.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKA0_a | 800h + (a × 4h) |

### 45.5.36.2 Function

The PKHA A0 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA A0 register is 1024 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 4096 bits, with A0 containing the least-significant 128 bytes of this value. The value in the PKHA A0 Register (or {A3, A2, A1, A0}, if used as a single value) is limited by the PKHA A Size Register (PKASZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.36.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.36.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_A0 | A0 VALUE |

## 45.5.37 PKHA A1 Register (PKA1_0 - PKA1_31)

### 45.5.37.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKA1_a | 880h + (a × 4h) |

### 45.5.37.2 Function

The PKHA A1 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA A1 register is 1024 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 4096 bits, with A0 containing the least-significant 128 bytes of this value. The value in the PKHA A1 Register, if used as a separate value, is limited by the PKHA A Size Register (PKASZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.37.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.37.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_A1 | A1 VALUE |

## 45.5.38 PKHA A2 Register (PKA2_0 - PKA2_31)

### 45.5.38.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKA2_a | 900h + (a × 4h) |

### 45.5.38.2 Function

The PKHA A2 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA A2 register is 1024 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 4096 bits, with A0 containing the least-significant 128 bytes of this value. PKHA A3 and A2 Registers may be similarly combined. The value in the PKHA A2 Register, if used as a separate value, is limited by the PKHA A Size Register (PKASZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.38.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.38.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_A2 | A2 VALUE |

## 45.5.39 PKHA A3 Register (PKA3_0 - PKA3_31)

### 45.5.39.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKA3_a | 980h + (a × 4h) |

### 45.5.39.2 Function

The PKHA A3 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA A3 register is 1024 bits in length. Note that PKHA A3, A2, A1 and A0 Registers may be combined to contain a value of up to 4096 bits, with A0 containing the least-significant 128 bytes of this value. The value in the PKHA A3 Register, if used as a separate value, is limited by the PKHA A Size Register (PKASZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.39.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_A3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.39.4 Fields

| Field | Function |
|---|---|
| 31-0 | A3 VALUE |

| Field | Function |
|---|---|
| PKHA_A3 | |

## 45.5.40 PKHA B0 Register (PKB0_0 - PKB0_31)

### 45.5.40.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKB0_a | A00h + (a × 4h) |

### 45.5.40.2 Function

The PKHA B0 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA B0 register is 1024 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 4096 bits, with B0 containing the least-significant 128 bytes of this value. The value in the PKHA B0 Register (or in {B3, B2, B1, B0} if used as a single value) is limited by the PKHA B Size Register (PKBSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the Mode Register (PublicKey) (MDPK) contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.40.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.40.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_B0 | B0 VALUE |

## 45.5.41 PKHA B1 Register (PKB1_0 - PKB1_31)

### 45.5.41.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKB1_a | A80h + (a × 4h) |

### 45.5.41.2 Function

The PKHA B1 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA B1 register is 1024 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 4096 bits, with B0 containing the least-significant 128 bytes of this value. The value in the PKHA B1 Register, if used as a separate value, is limited by the PKHA B Size Register (PKBSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MRPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.41.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.41.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_B1 | B1 VALUE |

## 45.5.42 PKHA B2 Register (PKB2_0 - PKB2_31)

### 45.5.42.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKB2_a | B00h + (a × 4h) |

### 45.5.42.2 Function

The PKHA B2 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA B2 register is 1024 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 4096 bits, with B0 containing the least-significant 128 bytes of this value. PKHA B3 and B2 Registers may be similarly combined. The value in the PKHA B2 Register, if used as a separate value, is limited by the PKHA B Size Register (PKBSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MRPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.42.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.42.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_B2 | B2 VALUE |

## 45.5.43 PKHA B3 Register (PKB3_0 - PKB3_31)

### 45.5.43.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKB3_a | B80h + (a × 4h) |

### 45.5.43.2 Function

The PKHA B3 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA B3 register is 1024 bits in length. Note that PKHA B3, B2, B1 and B0 Registers may be combined to contain a value of up to 4096 bits, with B0 containing the least-significant 128 bytes of this value. The value in the PKHA B3 Register, if used as a separate value, is limited by the PKHA B Size Register (PKBSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MRPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.43.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_B3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.43.4 Fields

| Field | Function |
|---|---|
| 31-0 | B3 VALUE |

| Field | Function |
|---|---|
| PKHA_B3 | |

## 45.5.44 PKHA N0 Register (PKN0_0 - PKN0_31)

### 45.5.44.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKN0_a | C00h + (a × 4h) |

### 45.5.44.2 Function

The PKHA N0 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA N0 register is 1024 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 4096 bits, with N0 containing the least-significant 128 bytes of this value. The value in the PKHA N0 Register (or {N3, N2, N1, N0}, if used as a single value) is limited by the PKHA N Size Register (PKNSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.44.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 45.5.44.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_N0 | N0 VALUE |

### 45.5.45 PKHA N1 Register (PKN1_0 - PKN1_31)

#### 45.5.45.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKN1_a | C80h + (a × 4h) |

#### 45.5.45.2 Function

The PKHA N1 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA N1 register is 1024 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 4096 bits, with N0 containing the least-significant 128 bytes of this value. The value in the PKHA N1 Register, if used as a separate value, is limited by the PKHA N Size Register (PKNSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.45.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.45.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_N1 | N1 VALUE |

## 45.5.46 PKHA N2 Register (PKN2_0 - PKN2_31)

### 45.5.46.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKN2_a | D00h + (a × 4h) |

### 45.5.46.2 Function

The PKHA N2 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA N2 register is 1024 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 4096 bits, with N0 containing the least-significant 128 bytes of this value. PKHA N3 and N2 Registers may be similarly combined. The value in the PKHA N2 Register, if used as a separate value, is limited by the PKHA N Size Register (PKNSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.46.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.46.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_N2 | N2 VALUE |

## 45.5.47 PKHA N3 Register (PKN3_0 - PKN3_31)

### 45.5.47.1 Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| PKN3_a | D80h + (a × 4h) |

### 45.5.47.2 Function

The PKHA N3 Register contains a numeric value of up to 128 bytes, right-justified. The LSByte is in the base offset of the register. The PKHA N3 register is 1024 bits in length. Note that PKHA N3, N2, N1 and N0 Registers may be combined to contain a value of up to 4096 bits, with N0 containing the least-significant 128 bytes of this value. The value in the PKHA N3 Register, if used as a separate value, is limited by the PKHA N Size Register (PKNSZ).

Note that the PKHA Registers can be read/written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MDPK contains a PKHA operation mode value, then writes to PKHA Registers will be ignored, and reads of PKHA Registers will return a value of zero.

### 45.5.47.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKHA_N3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.47.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_N3 | N3 VALUE |

## 45.5.48 PKHA E Register (PKE_0 - PKE_127)

### 45.5.48.1 Offset

For a = 0 to 127:

| Register | Offset |
|---|---|
| PKE_a | E00h + (a × 4h) |

### 45.5.48.2 Function

The PKHA E Register contains a numeric value of up to 512 bytes, right-justified. The LSByte is in offset E00h, in the right-most byte. The PKHA E register is 4096 bits in length. The value in the PKHA E Register is limited by the PKHA E Size Register (PKESZ).

Note that the PKHA E Register can be written only when the Mode Register (PublicKey) (MDPK) does not contain a value enabling a PKHA operation. If the MRPK contains a PKHA operation mode value, then writes to the PKHA E Register will be ignored.

### 45.5.48.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | PKHA_E | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | PKHA_E | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 45.5.48.4 Fields

| Field | Function |
|---|---|
| 31-0<br>PKHA_E | E VALUE |

## 45.6 Functional description

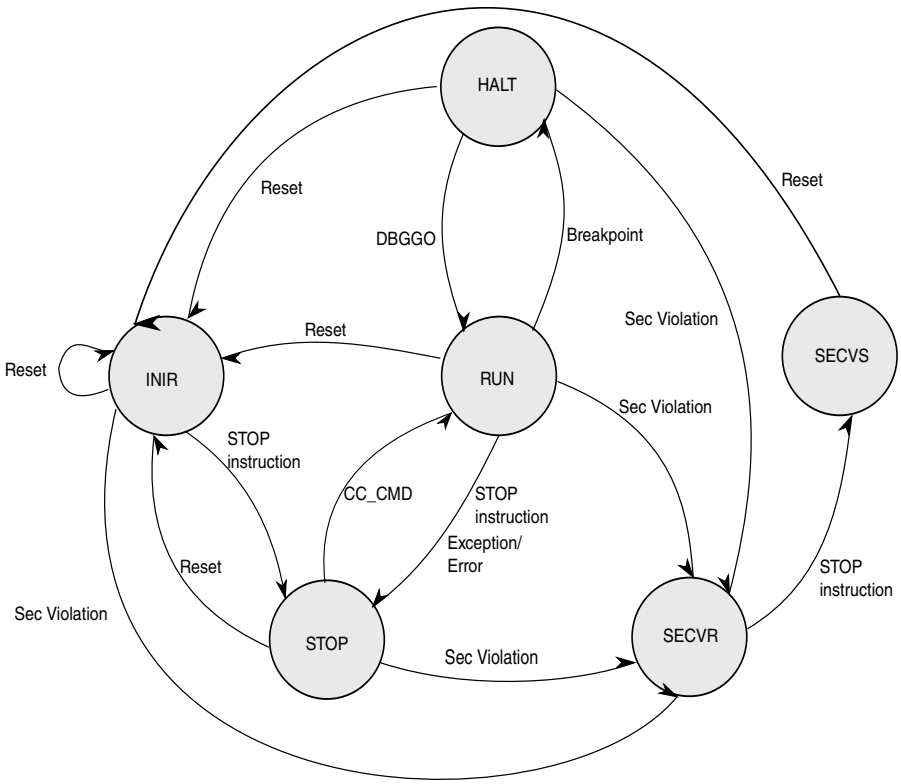
This section describes the programming model and operation of the CAU3 cryptographic core.

### 45.6.1 CryptoCore States

The CAU3 CryptoCore comes out of reset in the initialization run state (INIR). In this state it executes the initialization routine which initializes the DMEM.DRAM and the register file CC_Rn. TKCS='0000' indicates the INIR state. Once the CCore have transitioned to the STOP state indicated by TKCS='1001' it is ready to be programmed. After configuring all host interface (HIRC) registers and CryptoCore registers, a cryptographic routine is executed with a write to the CryptoCore Command (CC_CMD) register. At this point the CryptoCore transitions to the RUN state which blocks several HIRC registers from host CPU accesses. In RUN state the semaphore cannot be changed, from locked to unlocked and vice-versa. The CCore may then transition to STOP or HALT states: STOP state is reached when a STOP instruction is executed and HALT state is reached when a debug breakpoint occurs. In STOP or HALT states the access to HIRC registers are allowed and the semaphore can be locked and unlocked. The transition to HALT state is only allowed if the CryptoCore debug mode is enabled by the DBGCSR[DDBG] bit.

The figure below describes the CryptoCore states transitions.

**Figure 45-4. CryptoCore State Transition**




In case of a security violation, independent of the CCore state, it transitions to the security violation run state, SECVR, and after executing a special task stored in the IMEM.IROM it transitions to the security violation stop state, SECVS, staying at this state until a reset occurs, when it transitions to the INIR state. A hardware reset always re-initializes the CCore at INIR state. See specific section for use cases of the module reset CR[MRST].

## 45.6.2 Memory Map

The CAU3 is fully configured and programmed through the IPS interface. It provides access to the module registers within the range from (Base addr + 0x0) up to (Base addr + 0x1000). The CryptoCore resources directly accessed from the IPS interface are the configuration, debug, Register File and control/status registers. The CAU3 memory modules, IRAM, IROM and DRAM are accessed in an indirect manner, using the DBGMADR register.

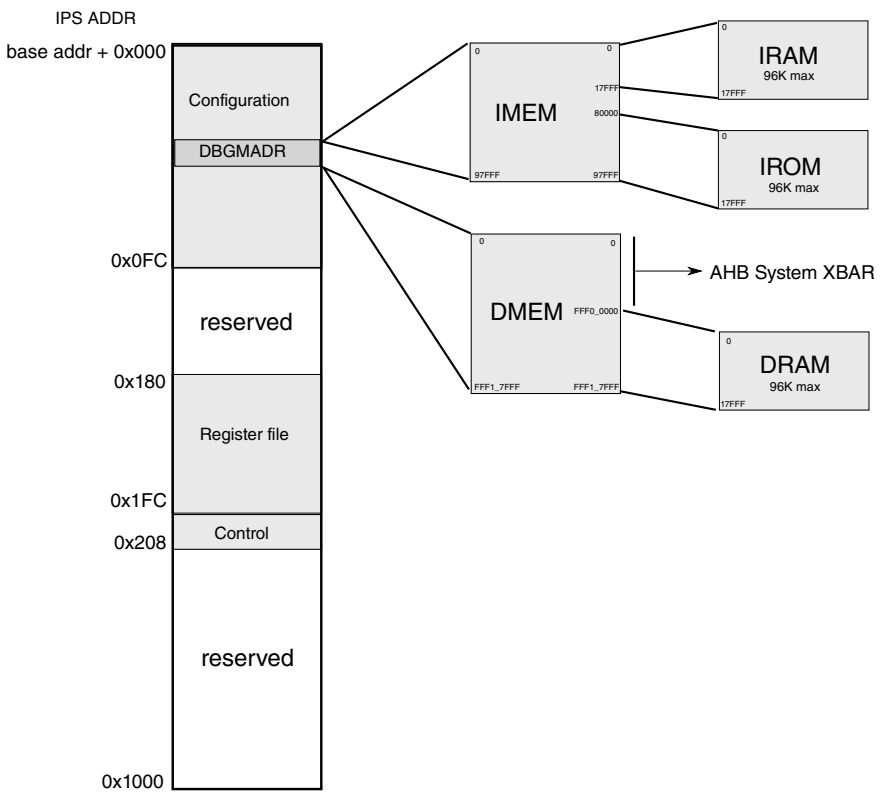
The DBGMADR is a 32-bit register that provides the address for the IMEM and DMEM memory regions. The selection of the target memory module from the IPS address bus is done by the Debug Memory Command Register (DBGMCMD). Before the access to the

memory is executed, the host programs the DBGMADR with the memory address. The DBGMADR is automatically incremented by the hardware after each access, if DBGMCMD[IA] bit is set, providing sequential addresses and allowing the programming of a contiguous memory region without the need for address reconfiguration in the DBGMADR register.

The physical memory modules in the IMEM address range are the IRAM and IROM. The DMEM address range is translated into the DRAM and AHB addresses.

The figure below depicts the address translation from the IPS address bus to the IRAM, IROM, DRAM and AHB addresses.

**Figure 45-5. IPS to physical memory address mapping**




## 45.6.3 Access to Cryptographic Routines

The CAU3 cryptographic routines are stored in the private IMEM. In order to execute those routines the host application software writes to the Program Counter (PC) register with the value corresponding to one element of a table stored at the beginning of the IMEM. The table elements are pointers to routines also stored in the IMEM. After that

the CMD register is written which causes the CryptoCore to start fetching instructions from the IMEM at the address specified by PC. Once initiated, the cryptographic routine runs until a STOP instruction is reached or until an exception occurs. See Handling Exceptions for details. The CryptoCore remains stopped allowing the host application software to access the results generated by the executed routine. These results are stored in the CryptoCore General Purpose Registers File (RGF). The process repeats with a new PC and CMD registers writes. The CryptoCore RGF is also initialized by the host application software before the cryptographic routine is executed.

### 45.6.4 Integrity checks

If an IPS access to a nonexistent memory location is attempted, the IPS bus cycle is terminated with a transfer error and no access is performed.

In case an illegal instruction is executed by the CryptoCore an error flag is set and the program execution is halted and the core transition to the STOP state. An illegal instruction is also provided in the CAU3 instruction set in order to allow the host application software to check if the illegal instruction behavior is correct.

IPS accesses are not allowed when a cryptographic routine is being executed. If a routine is being executed and an IPS access is attempted, the access is ignored and a termination error is indicated. This is to prevent security violations by the host application software when trying to access intermediary results stored in the CryptoCore registers.

The CAU3 provides a hardware semaphore to allow a master with locked semaphore to gain exclusive control of several CryptoCore registers. When the semaphore is locked, any attempt from another master, other than the one with matching master number (MSTRN), DID and with equal or higher privilege attributes, NS and PR, to access the CryptoCore semaphore protected registers is blocked and signaled with a transfer error. See Semaphore Operation for details.

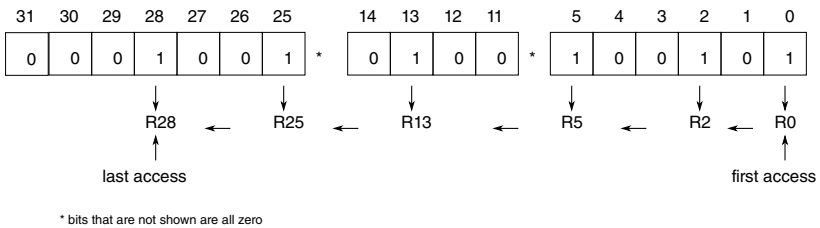
### 45.6.5 DMA and Address Remapping

The CAU3 implements a register address remapping in order to streamline the programming of the CryptoCore General Purpose Registers (CC_Rn). If the Address Remap Register (ARR) content is non-zero, accesses to CC_Rn address space, from 180h to 1FCh, are remapped to the CC_Rn pointed by ARR. After the access is done, the content of ARR is automatically modified by the CAU3 hardware in order to point to the next CC_Rn for the subsequent access. The remapping is done while ARR is non-zero.

Each ARR bit corresponds to one of the 32 CC_Rn registers. The least significant ARR bit , ARR[0], maps to the CC_R0. ARR[1] maps to CC_R1 and so forth. If ARR[0] = 1 then accessing any CC_Rn register will cause the access to be redirected to CC_R0 register. After the access is done, ARR[0] is cleared. If there is at least one non-zero bit in ARR, then the next access to CC_Rn is also remapped. The criteria for ARR bit selection is from the least to the most significant non-zero bit. If all ARR bits are zero then there is no address remapping and the IPS address is used to access the CC_Rn register.

The figure below describes the use of ARR remapping .

**Figure 45-6. ARR to CC_Rn Remapping**



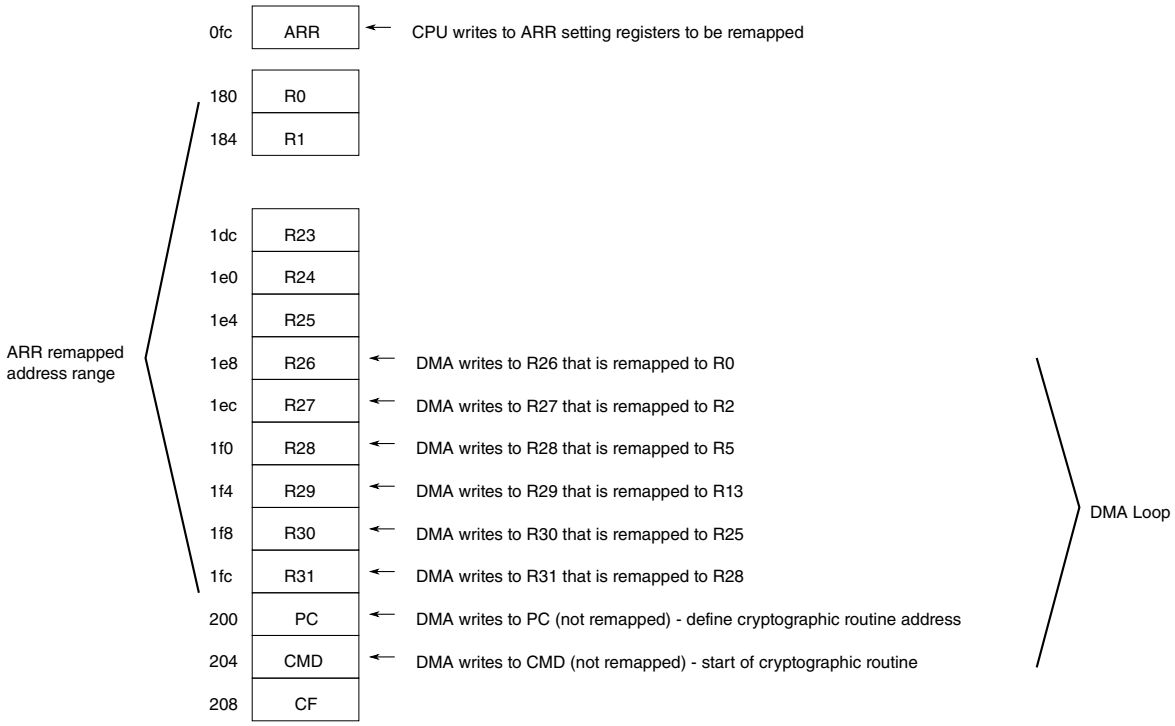

Consider the case where 6 register arguments must be written to the CryptoCore before beginning task execution. The host writes the bit map of the affected CC_Rn values into the ARR. Next, a sequential data transfer can be performed writing the appropriate register values into the CC_Rn registers using a starting offset address defined as 200h - 4*n, where n is the number of argument registers to be written.

If n = 6, then the starting offset address is 1E8h. The first writes at offset address {1E8h, 1ECh, 1F0h, 1F4h, 1F8h, 1FCh} are remapped to the appropriate CC_Rn registers based on the ARR.

The sequential CC_Rn programming allowed by using the ARR remapping is convenient for DMA transfers. After the CC_Rn registers are programmed the ARR content should be zero. The DMA then writes to the PC register defining the address of the selected cryptographic routine in the branch table to be executed and then writes to the CMD register to initiate the execution of the cryptographic routine.

The figure below describes an use case which considers the ARR programming of the previous figure.

**Figure 45-7. DMA ARR Use Case**

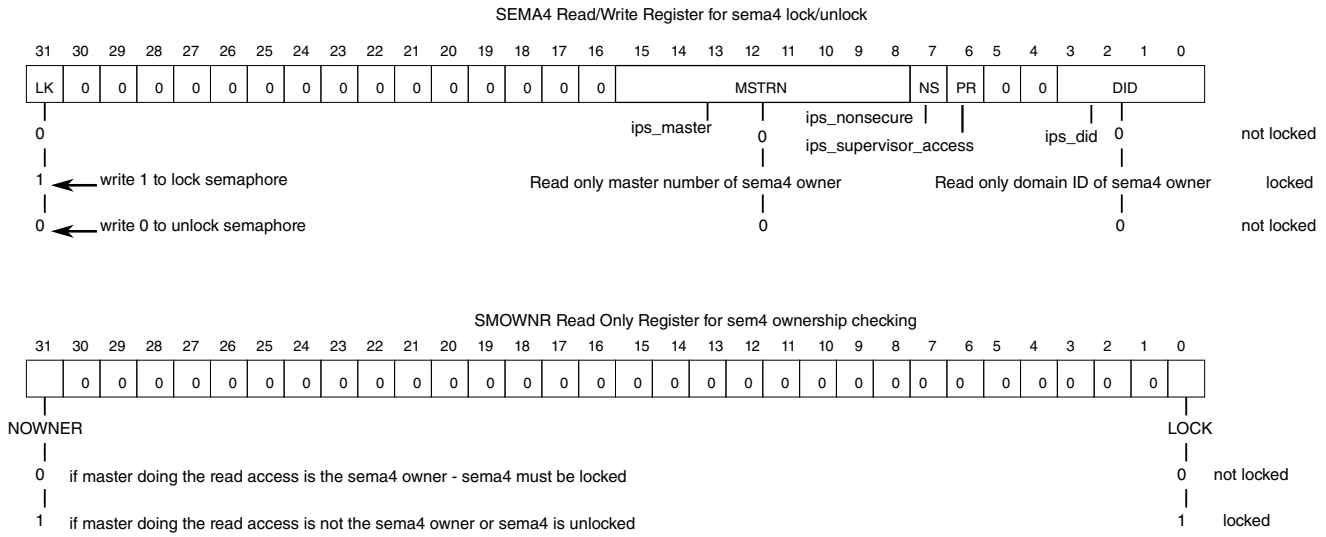



## 45.6.6 Semaphore Operation

The CAU3 module provides a hardware semaphore that allows multiple host processors or associated bus mastering modules, for example, a DMA, to take ownership of shared CAU3 resources. After locking the semaphore, by writing 1 to the SEMA4[LK] lock bit, the master host becomes the semaphore owner and then can access several CAU3 resources privately, meaning that, only host interface accesses with matching Master Number (MSTRN), Domain ID (DID) and with same or higher privilege as defined by the privilege and secure attribute signals NS and PR, compared to the semaphore register (SEMA4) fields, are allowed to access the semaphore protected registers. See section Access to Memory Mapped Registers for details about registers protected by semaphore.

Attempted accesses to the protected registers from any master other than the semaphore owner are terminated with an error. In order to release the semaphore, the semaphore owner master writes 0 to the SEMA4[LK] bit.

**Figure 45-8. Semaphore Operation**




The read data on the SMOWNR register returns the semaphore state with 3 distinct values, allowing an easy single conditional branch resolution:

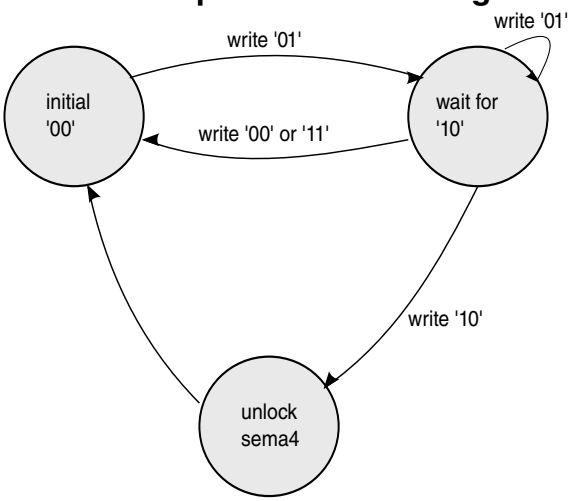
```
If (SMOWNR == 0x8000_0000) then
      semaphore is unlocked
else if (SMOWNR == 1) then
      semaphore is locked and master is the owner
else if (SMOWNR == 0x8000_0001) then
      semaphore is locked and master is not the owner
```

A master is considered the owner of a locked semaphore if the access attributes, MSTRN and DID, match with the SEMA4 register fields and the privilege/secure attributes, NS and PR, are the same or higher than the values stored in the SEMA4 register. See Privilege and Secure attributes precedence below for details.

A locked semaphore can be unlocked by the semaphore owner or by any other master using the CR[RSTSM4] bit, provided it has secure and privilege access attributes and the CryptoCore is in Stop state. In general the semaphore state can only be changed in Stop

state. The figure below describes a state machine based on CR[RSTSM4] values. It is initially in the '00' state. A write '01' moves the state to "wait for 10". Once in this state if '10' is written the "unlock sema4" state is reached and the semaphore is unlocked.

**Figure 45-9. Semaphore unlock using RSTSM4**




The Master and DID IPS sideband signals are also used for mastering of CAU3 AHB accesses. If the semaphore is locked, the Master Number and DID of the locked semaphore are used in the CAU3 AHB accesses. If the semaphore is not locked, the Master and DID of the IPS access of the Start Command (CC_CMD) register are used in the AHB accesses executed during the task execution.

### Semaphore operation and debug control signals

The DBGCSR[DDBG] and DBGCSR[DDBGMC] debug bits control the semaphore behavior:

- {DDBG,DDBGMC} = '00': semaphore is disabled
- {DDBG,DDBGMC} != '00': semaphore is enabled

### Privilege and Secure attributes precedence

The *Privileged* and *Nonsecure* attributes, PR and NS sideband signals of the IPS bus, define the privilege level of the master on accessing peripheral registers. From the highest to the lowest priority level the access privileges defined by {PR,NS} are as follows:

- PrivSecure (10) > PrivNonsecure (11) > UserSecure (00) > UserNonsecure (01)

## 45.6.7 Debug Resources

The CAU3 CryptoCore implements debug features to help the development of cryptographic routines. The debug features are:

- Breakpoint at a defined instruction address
- Code execution in step mode
- Resume program execution from breakpoint

There are two types of breakpoints: the hardware breakpoint using the Debug PC Breakpoint (DBGPBR) and the software breakpoint implemented by the HALT instruction.

### Hardware Breakpoint

Assuming the debug resources are enabled by DBGCSR[DDBG]=0, after the hardware breakpoint is enabled by DBGCSR[PBREN]=1, if the Program Counter (PC) matches the DBGPBR[PCBKPT] field, the program execution is halted before the instruction of the matching PC is executed. The program execution is resumed by setting the DBGCSR[DBGGO] bit. If the DBGCSR[SIM] bit is set, the next instruction is executed and the CryptoCore enters halt state again, implementing a step by step execution. If SIM is cleared and DBGGO is set, the program execution is resumed until the next breakpoint or a STOP instruction is executed.

### Software Breakpoint

The software breakpoint is implemented by replacing a regular instruction by the HALT instruction. After reaching a HALT instruction the program execution is halted and the program variables can be inspected or modified. After that the debug software replaces the HALT instruction with the actual instruction and resumes the program execution by setting the DBGCSR[DBGGO] bit. The instruction that replaced the HALT instruction is the first instruction to be executed after DBGGO is set. Note that if DBGGO is set and the HALT instruction is not replaced by any other instruction, the CryptoCore does not exit the breakpoint trap and remains at the same breakpoint address. The debug resources need to be enabled by DBGCSR[DDBG]=0. If a HALT instruction is reached and the DBGCSR[DDBG]=1, the HALT instruction is executed as an invalid instruction.

Once reaching a breakpoint the software has access to the CC_Rn registers, IMEM and DMEM through the IPS bus interface. Accesses to these resources while the CryptoCore is running are not allowed and return no valid data for IPS reads. Write accesses are not executed. In both cases a transfer error is issued.

In debug mode, if the CryptoCore is halted, the IMEM and DMEM can be accessed using the DBGMDR, DBGMCMD and DBGADR registers. If DBGCSR[DDBG] is set or if the CryptoCore is running, any attempt to access the IMEM and DMEM is blocked and the access is terminated with an error. The private memories content are not modified.

There are three status flags in the DBGCSR register that indicate the CryptoCore halt condition. The core could be halted due to a hardware breakpoint, HALT instruction or single step. In case there is more than one reason for halt, only one of the following flags are set at a time, highest priority on top:

1. SIMHF, single step
2. HLTIF, HALT instruction
3. PCBHF, hardware breakpoint

### 45.6.8 Virtual External Registers

SoC devices with the CAU3 include a 256-bit secret key as an input to the accelerator core. The 256-bit input vector can be treated as two separate 128-bit keys or a single 256-bit key. These inputs can be used as a Key Encryption Key (KEK) for unwrapping user-defined key blobs containing additional keys, or they can be used for encryption and decryption operations directly.

Regardless of the intended use case(s), care must be taken in the management of these SoC input keys. Note there can be various implementations at the SoC level for storing the KEK values. For example, they might be stored in the flash IFR, or implemented in eFuses, etc. In any case, the 256 bits of input key information is routed to the CAU3 over direct point-to-point connections. This input vector must not be generally accessible in any user-accessible programming model or on-chip memory.

Within the CAU3, the handling of the SoC input key(s) is made via the definition of an alternate read-only external (as viewed from the CryptoCore) “virtual” registers. This read-only external register space is only accessible via the MVXTR (MoVe eXternal To Register) instruction. This virtual register space includes the 256 bits of SoC input key as well as specific HIRC programming model registers to provide information on CAU3 configuration, control and status.

The memory map for this virtual read-only register space is defined as:

| RegID | Register |
|---|---|
| 0 | Processor Core Type (PCT) |
| 1 | Memory Configuration (MCFG) |
| 2 | Control Register (CR) |
| 3 | Status Register (SR) |
| 4 | RAZ, RFU |
| 5 | RAZ, RFU |
| 6 | RAZ, RFU |
| 7 | RAZ, RFU |

*Table continues on the next page...*

| RegID | Register |
|---|---|
| 8 | key0_w0 in BE format, i.e., byte-reversed input0[ 31: 0] |
| 9 | key0_w1 in BE format, i.e., byte-reversed input0[ 63: 32] |
| 10 | key0_w2 in BE format, i.e., byte-reversed input0[ 95: 64] |
| 11 | key0_w3 in BE format, i.e., byte-reversed input0[127: 96] |
| 12 | key1_w0 in BE format, i.e., byte-reversed input1[ 31: 0] |
| 13 | key1_w1 in BE format, i.e., byte-reversed input1[ 63: 32] |
| 14 | key1_w2 in BE format, i.e., byte-reversed input1[ 95: 64] |
| 15 | key1_w3 in BE format, i.e., byte-reversed input1[127: 96] |

It should be noted that the SoC key inputs are defined as a 16- or 32-element little-endian byte array. As a result, the logical mapping of the byte array to the input vectors is defined as:

| |
|---|
| soc_input_key0[ 31: 0] = key0_w0 = {k0[ 3], k0[ 2], k0[ 1], k0[ 0]} |
| soc_input_key0[ 63: 32] = key0_w1 = {k0[ 7], k0[ 6], k0[ 5], k0[ 4]} |
| soc_input_key0[ 95: 64] = key0_w2 = {k0[11], k0[10], k0[ 9], k0[ 8]} |
| soc_input_key0[127: 96] = key0_w3 = {k0[15], k0[14], k0[13], k0[12]} |

The second 128 bits of input key can be viewed as either a continuation of the k0 byte array or as a second key (k1) byte array:

| |
|---|
| soc_input_key1[ 31: 0] = key0_w4 = {k0[19], k0[18], k0[17], k0[16]} |
| soc_input_key1[ 63: 32] = key0_w5 = {k0[23], k0[22], k0[21], k0[20]} |
| soc_input_key1[ 95: 64] = key0_w6 = {k0[27], k0[26], k0[25], k0[24]} |
| soc_input_key1[127: 96] = key0_w7 = {k0[31], k0[30], k0[29], k0[28]} |

Or

| |
|---|
| soc_input_key1[ 31: 0] = key1_w0 = {k1[ 3], k1[ 2], k1[ 1], k1[ 0]} |
| soc_input_key1[ 63: 32] = key1_w1 = {k1[ 7], k1[ 6], k1[ 5], k1[ 4]} |
| soc_input_key1[ 95: 64] = key1_w2 = {k1[11], k1[10], k1[ 9], k1[ 8]} |
| soc_input_key1[127: 96] = key1_w3 = {k1[15], k1[14], k1[13], k1[12]} |

The MVXTR instruction has two operand arguments: the destination RGF register in which the external register content is going to be loaded and the pointer to the external register to select one of the 16 possible inputs.

### 45.6.9 Security Violation

The CAU3 detects and responds to security threats by the assertion of an interrupt request and optionally the assertion of the *cau3_security_violation* output signal. The SoC is responsible for processing the security violation indication and in response generate the appropriate actions in order to protect the system against intruders. In particular the CAU3 receives a *soc_security_violation* input signal that, once asserted, forces the CCore to branch to a *security violation routine* stored in the IMEM.IROM. The execution of this routine clears all data stored in the DMEM.DRAM and CC_Rn registers in order to avoid the intruder to steal information stored in these memory locations.

There are specific events within the CCore that cause the assertion of the *cau3_security_violation* signal:

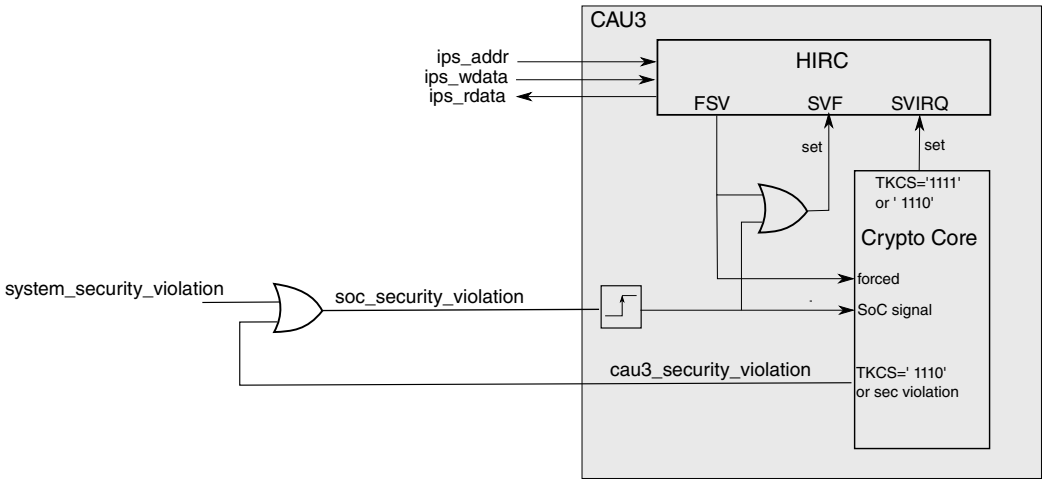
- Attempt to access CAU3 private memories, IMEM.IRAM or DMEM.DRAM, through the registers DBGMCMD, DBGMADR and DBGMDR if {DDBG,DDBGMC} != "00"
- Software setting security violation by executing STOP or STOPI instruction with status = "110"

Once the *cau3_security_violation* output signal is asserted it is up for the SoC logic to respond to this event by asserting the *soc_security_violation* signal.

If CAU3 is executing a task when the *soc_security_violation* signal asserts, it immediately branches to the *security violation routine*. If in STOP state, the CCore initiates the execution of the security routine. An especial case is when CAU3 is in low power mode, controlled by the assertion of CR[MDIS] bit. In this case, the module exits low power and executes the *security violation routine*. After that, it stays in security violation stop state , *SECVS*, out of low power mode, remaining at this state until a hardware reset is issued.

The figure below describes the CAU3 interface with the *soc_security_violation* signal and flag control logic.

**Figure 45-10. Security Violation Hardware Support**




The software can test the security violation logic by setting the *force security violation* control bit, CR[FSV] . In this case the behavior of the CCore is similar to the *SoC security violation* with the exception that a module reset, CR[MRST] can also be used, in addition to the hardware reset, to bring the module to normal mode of operation. If *soc_security_violation* is set, it is not possible to use CR[MRST] to reset the CCore to initialization run state (INIR).

The detection of the *soc_security_violation* signal transitioning to logic 1, or the set of CR[FSV] bit, sets the security violation flag, SR[SVF], immediately. Those events also trigger the execution of the *security violation routine*.

After SR[SVF] is set only accesses to PCT, MCFG, CR and SR registers are allowed. Accesses to other registers return an error.

The execution of any task may assert the *cau3_security_violation* signal and/or the SR[SVIRQ] flag, and consequently an interrupt request. Using the task completion status SR[TKCS]="1111", sets the SR[SVIRQ] flag. The task completion status SR[TKCS]="1110", asserts the *cau3_security_violation* signal and also sets the SR[SVIRQ] flag.

## 45.6.10 Module Reset

The module reset CR[MRST] provides for the application software the capability to reinitialize the CCore, in some circumstances, by forcing the execution of a STOP instruction. A typical scenario that would require the use of a module reset is when a task enters in an infinite loop due to a programming error. This task can then be terminated by CR[MRST], forcing the execution of a STOP instruction.

The MRST does not break any instruction being executed, meaning that, if there is an AHB transaction waiting for an *hready* acknowledgement, this instruction needs to complete, i.e. *hready* is asserted, before the STOP instruction is forced.

MRST is completely asynchronous to the task execution, therefore there is no way to synchronize the forced STOP with any task instruction.

The use of MRST should be possible in any scenario with the exception of the following, in which case setting this bit is ignored by the hardware:

- CCore at security violation run (SECVR) state
- CCore at security violation stop (SECVS) state if *soc_security_violation* was asserted
- CCore is running the initialization routine (INIR)
- CCore is already in STOP state

In all other cases when CR[MRST] is set, a STOP instruction execution is forced and the CCore transitions to STOP state. The task completion status (TKCS) corresponding to the forced STOP is "1001", indicating an error free task completion.

## 45.6.11 Handling Exceptions

This section describes the CAU3 exceptions, such as errors occurred during the execution of a task. It is also described the behavior of the interruptions that signal events to the host CPU.

CAU3 implements two interrupt signals to the host CPU to signal an *error free end of task* and an *end of task with error*. These interruptions are mutually exclusive, meaning that only one of these interrupt signals is asserted by a task and it is done at the end of the task. Also the interruptions do not assert when a task is executing. If enabled, they assert at the end of a task when the CCore is at the STOP state.

The *end of task with error* asserts at the occurrence of the following events:

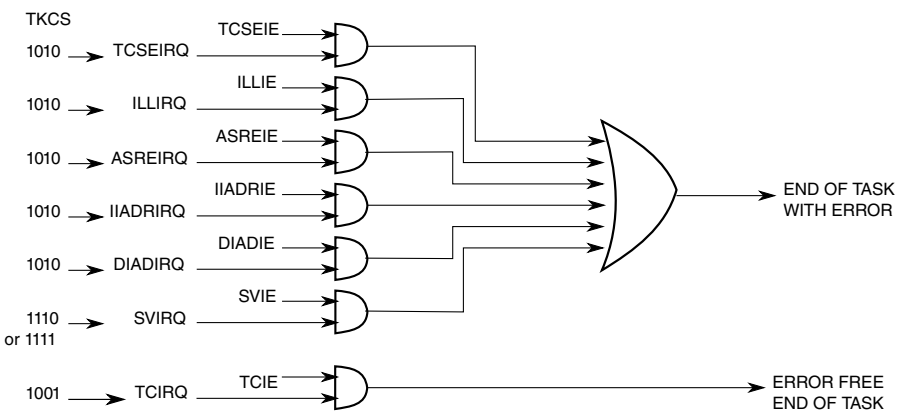
- DMEM illegal address
- IMEM illegal address
- AHB access error
- Execution of an illegal instruction

- Task termination error indicated by SR[TKCS] = "1010"
- Security violation task termination error indicated by SR[TKCS]= "1110" or "1111"

Several flags in the Status Register give the required resolution for the events signaled by the interruptions. The SR[TKCS] field is the main status indication of the most recent executed task and also the current state of the CryptoCore. In addition, there are individual flags to signal each completion status. Each flag controls the interrupt signals assertion. Error indication flags control the *end of task with error*. Task completion with no error flag set controls the *error free end of task* interrupt.

The figure below describes the logic behind the interrupt request signals.

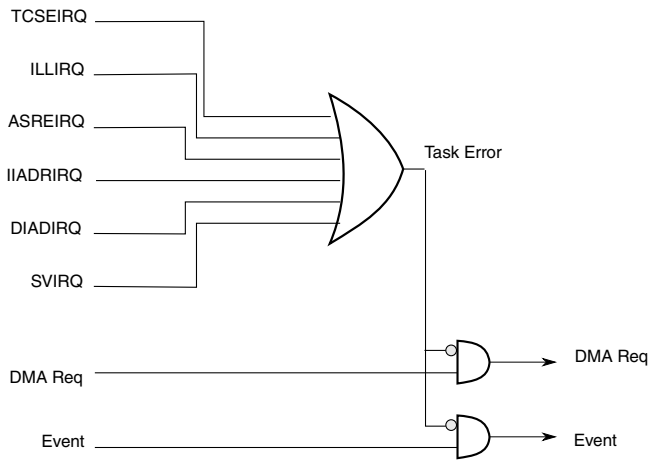
**Figure 45-11. Interrupt request logic**




Note that depending on the completion status, TKCS, the corresponding flag is set and if enabled , the interrupt signal is asserted. Clearing the flag, or flags, causes the interrupt to de-assert. TKCS and flags are independent registers, thus clearing the flags does not clear TKCS.

In case the task completion configuration, TCCFG, selects to signal a task completion by a DMA request (TCCFG='100') or assert the task completion event signal (TCCFG='010') and any error indication flag is set, then neither DMA request nor task completion event signal are asserted. The figure below describes this logic.

**Figure 45-12. DMA request and Task Completion Event signal logic**




### Instructions accessing illegal DMEM addresses

If a Load/Store instruction accesses an illegal DMEM address, the instruction is not executed and the task is terminated with an error. Also SR[DIADIRQ] flag is set.

## 45.6.12 Access to Memory Mapped Registers

This section describes the restrictions when accessing CAU3 memory mapped registers considering CryptoCore mode (debug enabled or not), access attributes and semaphore state. Also the state of the core (initialization run, running, stop, halted and security violation) is considered for the accesses as well as the low power mode.

The DBGCSR[DDBG] and DBGCSR[DDBGMC] debug bits define if the semaphore is enabled or disabled:

- {DDBG,DDBGMC} = '00': semaphore is disabled
- {DDBG,DDBGMC} != '00': semaphore is enabled

In the table below the access restriction is defined for all registers. The following convention is used:

- PS: requires PrivSec attributes- fail to comply with this requirement generates an error for reads and writes . Read accesses return zeroed data
- Y: access is allowed

- Yw: access is allowed only for the semaphore owner- fail to comply with this requirement generates an error for reads and writes . Read accesses return zeroed data
- Y(ro): access is allowed for read only - writes generate an error

**NOTE**

Y(ro) is used to identify read/write registers that become read only in specific scenarios. Please check the register definition section for information about registers access types for all registers.

- N: access is blocked - reads and writes generate an error

**Table 45-1. Semaphore and register access restrictions**

| Register | Attrib | Semaphore state | | | | CryptoCore State | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| name | Access mode | disabled unlocked | disabled locked | enabled unlocked | enabled locked | RUN[1] | STOP | HALT | LP | SEC VIOL |
| PCT | PS | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| MCFG | PS | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| CR | PS | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| SR | - | Y | Y | Y | Y | Y | Y | Y | Y(ro)[2] | Y |
| SEMA4 | - | Y | Y | Y | Yw[3] | N | Y | N | N | N |
| SMOWNR | - | Y | Y | Y | Y | Y | Y | Y | N | N |
| ARR | - | Y | Yw | N | Yw | N | Y | Y | N | N |
| CC_Rn | - | Y | Yw | N | Yw | N | Y | Y | N | N |
| CC_PC | - | Y | Yw | N | Yw | N | Y | Y | N | N |
| CC_CMD | - | Y | Yw | N | Yw | N | Y | N | N | N |
| CC_CF | - | Y | Yw | N | Yw | N | Y | Y | N | N |
| DBGCSR[4] | PS | Y | Y | Y | Y | Y | Y | Y | N | N |
| DBGPBR[5] | PS | Y | Y | Y | Y | N | Y | Y | N | N |
| DBGMCMD | PS | Y | Y | N | N | N | Y | Y | N | N |
| DBGMADR | PS | Y | Y | N | N | N | Y | Y | N | N |
| DBGMDR | PS | Y | Y | N | N | N | Y | Y | N | N |

1. **Initialization RUN is also included.**
2. This register is read/write and becomes read only in low power mode.
3. Writes by non owners do not generate errors, but have no effect. Reads return zeros if the master has no PS attribute, otherwise return correct register data.
4. DBGCSR is not accessible if DBGCSR[DDBG]=1, in Initialization and security violation RUN states.
5. DBGPBR is not accessible if DBGCSR[DDBG]=1.

In order to evaluate if a register is accessible in a particular semaphore configuration and CryptoCore state, it is required to extract the previous table row elements for the desired register inferring an AND operation between the semaphore and CryptoCore state elements. Consider the following equations to determine the register access allowances:

```
Y AND Y = Y (register is accessible)
Y AND Yw = Yw (register is accessible only by the semaphore owner)
Y AND N = N (register is not accessible)
Yw AND N = N (register is not accessible)
```

The Privilege Secure attributes are extracted from column 2. Below there is an example for the ARR register with the semaphore enabled and locked and the CryptoCore at STOP state:

```
ARR_access_permission = Yw AND Y (no PrivSec attributes required) = Yw
```

In this case ARR can be accessed by the semaphore owner only (more restrictive) and no privilege secure attributes are required.

Below there is an example for the DBGPBR with semaphore disabled and unlocked and CryptoCore at RUN state:

```
DBGPBR_access_permission = Y AND N (PrivSec attributes required) = N
```

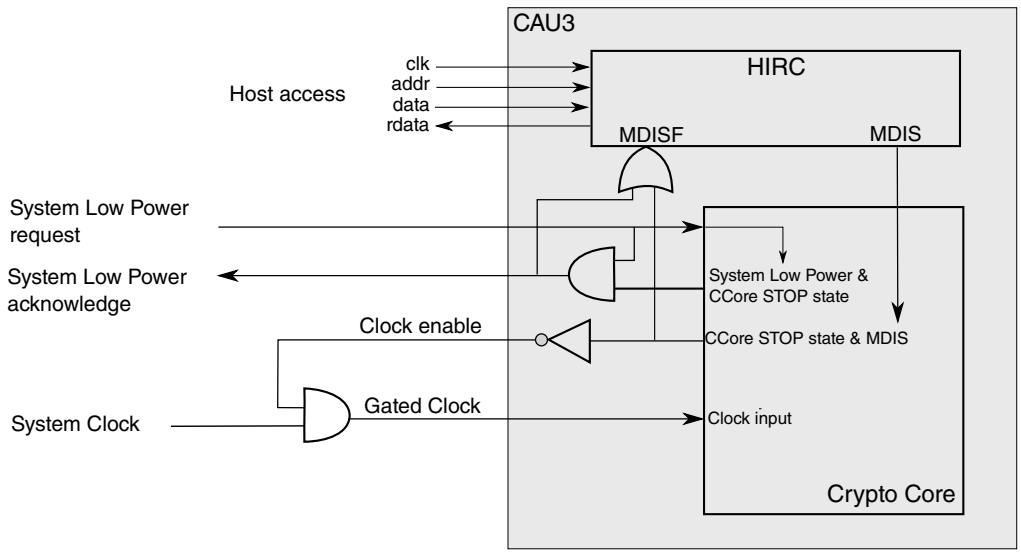
In this case DBGPBR is not accessible since it is blocked by the CryptoCore RUN state.

## 45.6.13 Low Power Operation

The CAU3's low power strategy is based on clock gating. In general, the internal clock to several registers is gated off if the register is not being updated at a specific clock cycle. The clock gating logic evaluates opportunities for gating off the clock on each clock cycle for each individual register. This strategy reduces the dynamic power but is not efficient if the entire module is not being used. In this case the clock for the whole

module should be gated off. This functionality is controlled by the CR[MDIS] bit. Once set, this bit turns the CryptoCore clock off and consequently blocks any activity in the module. The figure below describes the CAU3 clock gating control logic.

**Figure 45-13. Clock gating control logic**




**NOTE**

It is recommended to set the CR[MDIS] bit when the CryptoCore is already at STOP state, even though it would be possible to set it at any state. This procedure would avoid race conditions in the signals controlling the clock gating logic

Once set, the CR[MDIS] bit allows the CryptoCore clock to be gated off if the CryptoCore is at STOP state. At any other state the clock remains active until the STOP state is reached.

There is a second mechanism to turn the CryptoCore clock off which is the *system low power request*. This signal indicates an SoC request for the CryptoCore to enter the low power mode. If the CryptoCore is at STOP state the *system low power acknowledge* signal is asserted and the system clock is allowed to be gated off. Note that the CAU3 clock enable signal is not used in this process. When *system low power acknowledge* signal is asserted the clock enable signal is don't care.

**NOTE**

The system clock gating logic details are not described in the figure above. This logic is implemented at the SoC level.

The SR[MDISF], module disable flag, indicates that the CAU3 is at low power mode. This flag is set if the low power mode was entered by setting the CR[MDIS] bit or by the *system low power request* signal. In order to distinguish between the *system low power*

*mode* and MDIS low power mode, the CR[MDIS] should be considered. If CR[MDIS] is cleared and SR[MDISF] is set then CAU3 is in *system low power mode*. If CR[MDIS] and SR[MDISF] are both set, then CAU3 is in MDIS low power mode, but could also be in *system low power mode* at the same time.

In order to exit MDIS low power mode this bit should be cleared. The clock to CAU3 should be re-stored immediately after CR[MDIS]=0, assuming *system low power request* and *acknowledge* signals are not asserted. In case these signals are asserted the low power mode is under control by the SoC logic.

If a task is already running and the Host CPU software wants to set CAU3 CryptoCore in low power mode, then firstly the task needs to be terminated by a software reset, CR[MRST] = 1, and then the low power mode is enabled by setting CR[MDIS]=1. Since these two bits are in the CR register, they can be set at the same time.

**NOTE**

CAU3 does not enter low power mode if it is stopped with security violation flag set, CS[SVF]=1.

In order to avoid inconsistencies with DMA and Interrupt request signals, the interrupt and DMA request are de-asserted when CAU3 enters low power mode, either by CR[MDIS] =1 or by a system low power request signal assertion.

## 45.7 Public-key hardware accelerator (PKHA) functionality

The PKHA module is capable of performing a number of different operations used in public-key cryptography, including modular arithmetic functions such as addition, subtraction, multiplication, exponentiation, reduction, and inversion. All of these functions are provided in both integer and polynomial-binary field versions, except modular subtraction, which is the same as addition for binary polynomials. There are also elliptic-curve functions for point addition, point doubling, and point multiplication the standard prime and binary curves. Most of these functions can be performed timing-equalized to thwart timing-related side-channel attacks. PKHA also includes a Miller-Rabin primality test function for detecting prime numbers.

The PKHA internally performs modular multiply operations using "Montgomery multiplication". For efficiency, many of these functions have a variant which allows either inputs or outputs in Montgomery form. Some have variants to supply the Montgomery conversion factor. These save time over the variations without. Internally, the PKHA operates on digits of these values. Different versions of the PKHA may have a different digit size. This PKHA has a digit size of 16 bits. This has implications for the inputs and outputs of certain functions. See the discussion on Montgomery arithmetic.

Because the numbers used in public-key cryptography are typically quite large and often referenced many times during a function, the inputs to PKHA are loaded into registers. PKHA has four of these labeled A, B, E, and N. A and B are for operands and results. E is for "keys", and N holds the modulus. For ECC functions, A and B are divided up into equal-size quadrants to accommodate the greater number of inputs required.

PKHA also has two other types of functions for manipulating the data in the registers. These are the Clear Memory and Copy Memory functions. The Clear Memory function allows all or any combination of the registers to be overwritten with zeros. The Copy Memory functions can be used to copy data from any of the A, B or N registers or register quadrant to any register A, B, E or N.

PKHA requires that all data for a given function be loaded before the Mode Register is written to invoke a function. This convention indicates to PKHA that all needed data has been loaded, and the function can now be launched. The required data includes:

- PKHA A, B, E and N registers, as required per the function to be executed.
- PKHA A Size, PKHA B Size, PKHA E Size and PKHA N Size, as required.

Note that the PKHA A, B, E and N registers can be accessed only when the Mode Register is not programmed to any PKHA operation. Once the Mode Register contains any PKHA operation value, writes to the PKHA registers will be ignored, and reads will return zero data. After the PKHA operation is complete, the results can be read from the PKHA registers only after the Mode Register is cleared.

### 45.7.1 Modular math

Almost all math operations require with a modulus value in the N Memory. Math operations involving multiplication (multiplication, exponentiation, prime test, and ECC functions) are performed internally using Montgomery values.

### 45.7.2 About Montgomery values

The PKHA contains a Modular Arithmetic Unit. Multiplication is always modular multiplication:

$A * B \bmod N$.

The PKHA performs this computation with a Montgomery multiplier. A Montgomery multiplier can be more efficient than a multiply-then-reduce calculation because the modular reduction is done as part of the multiplication and the working product never gets larger than the modulus. In a normal multiplication, the product, before reduction, would be the size of the sum of the factors, so usually twice the size of the modulus. The

factors in a Montgomery multiplication each have an R factor, and, as part of the multiplication and modular reduction, one R is removed. Thus, the computation performed is:

*(AR * BR) / R* mod *N*.

The equivalent of "division by *R*" occurs even if one of the inputs does not have an *R* factor.

A number of PKHA functions accept inputs in Montgomery form instead of normal values. Some instead take $R^2$ mod *N* as an input. These functions can be faster than their normal-value alternatives if several operations are performed in a row or if these values are known in advance. This is because, before being used, ($R^2$ mod *N* needs to be computed and) normal values need to be converted internally to Montgomery form.

The Montgomery form of a value is *value*R* mod *N*, referred to here as $\underline{value}$. The term $R = 2^{SD}$ is the Montgomery factor, where D is the digit size (of a digit in the PKHA arithmetic unit), in bits, and S is the minimum number of digits needed to hold the value in N. R is therefore dependent on N and D.

To use the PKHA to convert a normal value to a Montgomery value, one must first compute (or know) $R^2$ mod *N*, the Montgomery Conversion Factor. The following steps can be used to convert a value from a normal value into its Montgomery form (A and B inputs may be reversed):

$R^2$ = MOD_R2(*N*)

$\underline{A}$ = MOD_MUL_IM_OM(*A*, B=$R^2$, *N*)

The equivalent F2M function can be used for binary polynomial values.

Eventually, the value needs to be converted out of Montgomery form. This can be done by performing another multiply ($R^2$ is not needed for this).

```
A = MOD_MUL_IM_OM(A=A, B=1, N)
```

Another method is to cause the PKHA to perform a multiplication and conversion to normal form. Internally, there are two multiplications: first the two inputs, then the product by one.

```
AB mod N = MOD_MUL_IM(A=A, B=B, N)
```

A third method is to have just one factor (either one) in Montgomery form:

```
AB mod N = MOD_MUL_IM_OM(A=A, B=B, N)
```

The following operations can be used to convert a value from a normal value into its Montgomery form (A and B inputs may be reversed):

The equivalent F2M functions can be used for binary polynomial values.

It is possible to add and subtract Montgomery values, if *R* mod *N* is the same. Do not mix and match Montgomery and normal values for addition or subtraction. 5 + 3R => *5/R + 3R* or *5/R + 3*; neither is likely the desired result.

## 45.7.3 Non-modular Math

Although addition, subtraction, and multiplication functions require a modulus, it is possible to perform these calculations without any reduction: the modulus must be larger than the expected result.

For addition and subtraction, this is easily done. For multiplication, the MOD_MUL function may be used, but it is not the most efficient, as internally first $R^2$ mod N will be computed, then two multiplications will be performed (first to convert one factor into Montgomery, then to compute the product, not in Montgomery).

For non-modular multiplication, MUL_IM_OM is much more efficient, as only one multiplication will be performed. This can be used if the factors are not in Montgomery form, i.e., if the product to be calculated is *A*B* instead of *A*B mod N*. Since the multiplier always "divides by *R*', a special modulus value in Nram is required which will make *R* have the value 1. This is done by creating a modulus *N = R-1* so that *R mod N* will have the value one. This way, normal values are the same as Montgomery values; no conversion is necessary and the multiplier will quietly "divide by one" to no effect.

As an example, on a PKHA with a digit size of 32 bits and a product which will be no more than six bytes long, $R = 2^{SD} = 2^{2*32} = 2^{64}$. Therefore the modulus must be 0xFFFFFFFFFFFFFFFF.

For computation with binary polynomials, the equivalent F2M functions may be used.

## 45.7.4 Elliptic-Curve Math

The PKHA provides point math operations on different types of elliptic curves. These include the ability to add two points (+ operator), double a point, and multiply a point by an integer (scalar) value (x operator).

The input points are assumed to be valid points on the curve. If non-point coordinates or invalid curve parameters are used an input, then a non-point set of coordinates are likely to be returned as output.

The minimum modulus is 1 byte. The maximum modulus is 1024 bits, 128 in length, or one quadrant.

If xx is the Point at Infinity, *P*and *Q* are points on the curve, and *j* and *k* are integers, then the following identities, as well as others easily derived by taking advantage of associative and commutive properites, hold:

- $P + Q = Q + P$
- $P = xx + P$
- $xx = 0 \times P$
- $(j + k)P = (j \times P) + (k \times P)$

There may be times when the negative of a point is necessary:

- When subtracting points $P_A - P_B$
- When multiplying by a negative integer: -abs(k) x $P_A$

To subtract, one can negate the second term and perform an addition, i.e.

$P_C = P_A - P_B = P_A + (-P_B)$

When multiplying by a negative value, one can either negate the starting point or the ending point. The multiplication value is the absolute value of the scalar, i.e., when *k* is negative

$P_C = k \times P_A = -abs(k) \times P_A = abs(k) \times (-1P_A) = -(abs(k) \times P_A)$

### 45.7.4.1 Point math over a prime field ($F_p$)

The ECC_MOD family of functions perform Add, Double, and scalar Multiply operations on points on a curve defined by the equation:

$E: y^2 = x^3 + ax + b \bmod p$

where p is the a prime integer > 3. These operations are available in Affine Coordinates (*x,y*).

The modulus (value in N memory) for these operations is *p*, also referred to as *q*.

The equality for the negative of a point *P*, in affine coordinates, is $-P = -(x,y) = (x, -y)$

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if they are not solutions to the curve equation E.

The representation of the point at infinity, in affine coordinates, depends upon the type of curve and the value of the *b* term of the curve's equation:

- Where *b* is equal to 0: (0, 1)
- Where *b* is not equal to 0: (0, 0)
- the equality for the negative of a point is $(X, -Y, Z) = (X, Y, Z)$

### 45.7.4.2 Point math over a binary field ($F_{2^m}$)

The ECC_F2M family of functions perform add, double, and scalar multiply operations on points on a curve defined by the equation:

E: $y^2 + xy = x^3 + ax^2 + b$

These operations are available in Affine Coordinates $(x,y)$. All inputs and output values of polynomial values are in polynomial basis. For example, $x^5+x+1$ is represented as 23h

The modulus (value in N memory) for these functions is $q$, the field-defining irreducible polynomial for the curve. Other documents use other symbols, including $p(t)$, $f(t)$, and $f$.

The equality for the negative of a point, in affine coordinates, is $-P = -(x,y) = (x, x+y)$.

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if they are not solutions to the curve equation E.

Because of the way the point operations are performed over a binary field, these functions require as an input the value $c$ rather than $b$. The relationship between these two values is:

$b = c^4 \bmod q$

and

$c = b^{2^{m-2}} \bmod q$, where $m$ is the degree (the power of its highest-power term) of $q$.

This $c$ value is referred to as $b'$ in the ECC Public-Key protocols for ECDSA sign, and so on. The calculation of $c$ is expensive, so it is obviously an advantage to calculate it only once or have it precomputed. See Special values for common ECC domains for these values for common ECC domains.

The representation of the point at infinity, in affine coordinates, is:

- (0, 0)

For $F_{2^m}$ curves,

- the curve equation is $E: Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$
- the equality for the negative of a point is $(X, X + Y, Z) = (X, Y, Z)$

All computations must be performed with the proper MOD or F2M operation, with the curve's modulus as the modulus of the function.

#### 45.7.4.3 Point math over a Montgomery-form curve ($F_p$)

The ECM_MOD family of functions perform scalar Multiply operations on the affine x value of points on a curve defined by the equation:

$E: By^2 = x^3 + Ax^2 + x \bmod p$

where p is the a prime integer.

The modulus (value in N memory) for these operations is $p$. The curve coefficient $B$ is not used in x-only multiply operations.

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if $x$ is not part of a solution to the curve equation E.

The equality for the negative of a point $P$, in affine coordinates, is $-P = -(x,y) = (x, -y)$

With the values left in registers after an x-only multiply, it is possible to compute the y which would correspond to the result of a (x,y) multiply.

#### 45.7.4.4 Point math over an Edwards-form curve prime field ($F_p$)

The ECT_MOD family of functions perform Add and scalar Multiply operations on points on a curve defined by the equation:

$E: ax^2 + y^2 = 1 + d^2x^2y^2 \bmod p$

where p is the a prime integer. These operations are available in Affine Coordinates.

These functions can also be used on the following curves, when $c^2$ is one:

- $E: x^2 + y^2 = c^2 + d^2x^2y^2 \bmod p$
- $E: x^2 + y^2 = c^2(1 + d^2x^2y^2) \bmod p$
- $E: ax^2 + y^2 = c^2(1 + d^2x^2y^2) \bmod p$

The modulus (value in N memory) for these operations is $p$.

The equality for the negative of a point $P$, in affine coordinates, is $-P = -(x,y) = (-x, y)$

The operations will not provide useful outputs if the inputs are not valid points on the curve, i.e., if they are not solutions to the curve equation E.

### 45.7.5 PKHA Mode Register

The following tables list the valid PKHA_MODE values for all PKHA functions:

PKHA Modular Arithmetic functions: Table 45-6

PKHA Elliptic Curve functions: PKHA MODE: Elliptic Curve Functions

PKHA Copy Memory functions: Table 45-14

**NOTE**

Use of any PKHA_MODE value not listed in these tables results in an invalid mode error.

### 45.7.5.1 PKHA MODE: clear memory function

**Table 45-2. PKHA Mode register format for clear memory function**

| 19 | 18 | 17 | 16 | | 11-10 | 9 | 8 | 7 | 6 | 5-0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Aram | Bram | Eram | Nram | . | Reserved | Q3 | Q2 | Q1 | Q0 | Function |
| *PKHA_MODE_MS* | | | | . | *PKHA_MODE_LS* | | | | | |

If the Function field in PKHA MODE specifies the clear memory function, PKHA expects to be in the format shown in Table 45-2. The PKHA RAMs to be cleared may be selected in any combination. Selecting one or more Quadrants for clearing will cause only the specified quadrants (of the specified RAMs) to be cleared. If no Quadrants are selected, then the whole RAM will be cleared.

**Table 45-3. PKHA mode register field descriptions for clear memory function**

| Bits | Description |
|---|---|
| 19 | Aram<br>This bit selects the A RAM for zeroization.<br>0: A not selected<br>1: A selected. |
| 18 | Bram<br>This bit selects the B RAM for zeroization.<br>0: B not selected<br>1: B selected |
| 17 | Eram<br>This bit selects the E RAM for zeroization.<br>0: E not selected<br>1: E selected |
| 16 | Nram<br>This bit selects the N RAM for zeroization.<br>0: N not selected<br>1: N selected |
| 11-10 | Reserved |

*Table continues on the next page...*

**Table 45-3. PKHA mode register field descriptions for clear memory function (continued)**

| Bits | Description |
|---|---|
| 9 | Quadrant 3<br>This bit selects the Quadrant 3 RAM for zeroization.<br>0: not selected<br>1: selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected. |
| 8 | Quadrant 2<br>This bit selects the Quadrant 2 RAM for zeroization.<br>0: not selected<br>1: selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected. |
| 7 | Quadrant 1<br>This bit selects the Quadrant 1 RAM for zeroization.<br>0: not selected<br>1: selected. Clearing will be only specified quadrant(s). Not valid is E RAM is selected. |
| 6 | Quadrant 0<br>This bit selects the Quadrant 0 RAM for zeroization.<br>0: not selected<br>1: selected. Clearing will be only specified quadrant(s). Not valid if E RAM is selected. |
| 5-0 | Function<br>The Function value for clearmemory is 000001. |

## 45.7.5.2 PKHA MODE: Arithmetic Functions

**Table 45-4. PKHA Mode Register Format for Arithmetic Functions**

| 19 | 18 | 17 | 16 | | 11 | 10 | 9-8 | 7-6 | 5-0 |
|---|---|---|---|---|---|---|---|---|---|
| inM | outM | F2m | R2 | . . | Reserved | Teq | OutSel | Reserved | Function |
| *PKHA_MODE_MS* | | | | . | *PKHA_MODE_LS* | | | | |

**Table 45-5. PKHA Mode register, format for arithmetic operation**

| Bits | Description |
|---|---|
| 19<br>inM | Inputs in Montgomery form. Indicates whether the inputs are in Montgomery form.<br>If inM=0 : Normal value representation<br>If inM=1 : Montgomery form. (Not valid for all functions.) |
| 18<br>outM | Outputs in Montgomery form. Indicates whether the outputs are to be left in Montgomery form or converted to normal values.<br>If outM=0 : Normal value representation<br>If outM=1 : Montgomery form. (Not valid for all functions.) |

*Table continues on the next page...*

**Table 45-5. PKHA Mode register, format for arithmetic operation (continued)**

| Bits | Description |
|---|---|
| 17<br>F2m | F2m. Indicates whether to use integer or binary polynomial arithmetic in executing the function.<br>If F2m=0 : Integer<br>If F2m=1 : Binary polynomial. (Not valid for all functions.) |
| 16<br>R2 | (R2 mod N). Indicates whether the term (R2 mod N) must be supplied as an input or will be calculated by the routine.<br>If R2=0 : ($R^2$ mod N) is calculated and applied, if needed<br>If R2=1 : ($R^2$ mod N) is an input. (Not valid for all functions.) |
| 11<br>Reserved | Reserved |
| 10<br>Teq | Timing Equalized. Indicates that a timing equalized version of the function should be executed.<br>If Teq=0 : No timing equalization<br>If Teq=1 : Timing equalization. (Not valid for all functions.) |
| 9-8<br>OutSel | Output destination select. Indicates which memory should contain the output of the selected function.<br>If OutSel=00b : B<br>If OutSel=01b : A<br>If OutSel=10b : Reserved<br>If OutSel=11b : Reserved |
| 7-6 | ECC Type<br>If EccType=00b : Standard/short Weierstrass, i.e. ordinary, traditional, "prime" (F2m=0) or "binary" (F2m=1), curve<br>If EccType=01b : Montgomery-form curve. F2m must be zero. (Not valid for all functions.)<br>If EccType=10b : (Twisted) Edwards form curve. F2m must be zero. (Not valid for all functions.)<br>If EccType=11b : Reserved |
| 5-0<br>Function | Function. Indicates which arithmetic function to execute.<br>If Function=000010b : Modular Addition (A + B) mod N<br>If Function=000011b : Modular Subtraction 1 (A - B) mod N<br>If Function=000100b : Modular Subtraction 2 (B - A) mod N<br>If Function=000101b : Modular Multiplication (A x B) mod N<br>If Function=000110b : Modular Exponentiation $A^E$ mod N<br>If Function=000111b : Modular Reduction A mod N<br>If Function=001000b : Modular Inversion $A^{-1}$ mod N<br>If Function=001001b : ECC Point Add (P1 + P2)<br>If Function=001010b : ECC Point Double (P2 + P2)<br>If Function=001011b : ECC Point Multiply (E x P1)<br>If Function=001100b : Montgomery Radix Constant $R^2$ mod N<br>If Function=001101 : CRT Radix Constant $R_E R_P$ mod P<br>If Function=001110 : Greatest Common Divisor GCD(A,N)-see note below<br>If Function=001111 : Miller-Rabin Primality Test -see note below |

**Table 45-5. PKHA Mode register, format for arithmetic operation**

| Bits | Description |
|---|---|
| | If Function=010111b : Modular Square Root<br><br>All other values for this field are currently reserved.<br>**NOTE:** When using the GCD function or any ECC function, a divide-by-zero error occurs if the value of the most significant digit of N is all zeros.<br>**NOTE:** When using the Miller-Rabin primality test function, if the most-significant digit of N is all zeros, the result is composite regardless of the value of N. |

**NOTE**

Note that the arithmetic functions with outputs going to the A RAM are identical to those with outputs going to the B RAM. The only difference is the output destination.

**Table 45-6. List of mode values for PKHA Integer Arithmetic Functions**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| MOD_ADD | Integer Modular Addition | B | 0 | 00002 | Integer Modular Addition (MOD_ADD) function |
| | | A | 0 | 00102 | |
| MOD_SUB_1 | Integer modular subtraction (A - B) | B | 0 | 00003 | Integer Modular Subtraction (MOD_SUB_1) function |
| | | A | 0 | 00103 | |
| MOD_SUB_2 | Integer modular subtraction (B - A) | B | 0 | 00004 | Integer Modular Subtraction (MOD_SUB_2) function |
| | | A | 0 | 00104 | |
| MOD_MUL<br>MOD_MUL_TEQ | Integer modular multiplication | B | 0 | 00005 | Integer Modular Multiplication (MOD_MUL) |
| | | A | 0 | 00105 | |
| | Timing equalized version | B | 1 | 00405 | |
| | | A | 1 | 00505 | |
| MOD_MUL_IM<br>MOD_MUL_IM_TEQ | Integer Modular Multiplication with Montgomery Inputs | B | 0 | 80005 | Integer Modular Multiplication with Montgomery Inputs (MOD_MUL_IM) |
| | | A | 0 | 80105 | |
| | Timing equalized version | B | 1 | 80405 | |
| | | A | 1 | 80505 | |
| MOD_MUL_IM_OM<br>MOD_MUL_IM_OM_TEQ | Integer Modular Multiplication with Montgomery Inputs and Outputs | B | 0 | C0005 | Integer Modular Multiplication with Montgomery Inputs and Outputs (MOD_MUL_IM_OM) Function |
| | | A | 0 | C0105 | |
| | Timinq equalized version | B | 1 | C0405 | |
| | | A | 1 | C0505 | |

*Table continues on the next page...*

**Table 45-6. List of mode values for PKHA Integer Arithmetic Functions (continued)**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| MOD_EXP<br>MOD_EXP_TEQ | Integer Modular Exponentiation | B | 0 | 00006 | Integer Modular Exponentiation (MOD_EXP and MOD_EXP_TEQ) |
| | | A | 0 | 00106 | |
| | Timing equalized version | B | 1 | 00406 | |
| | | A | 1 | 00506 | |
| MOD_EXP_IM<br>MOD_EXP_IM_TEQ | Integer Modular Exponentiation with Montgomery Inputs | B | 0 | 80006 | Integer Modular Exponentiation, Montgomery Input (MOD_EXP_IM and MOD_EXP_IM_TEQ) Function |
| | | A | 0 | 80106 | |
| | Timing equalized version | B | 1 | 80406 | |
| | | A | 1 | 80506 | |
| MOD_AMODN | Integer Modular Reduction | B | 0 | 00007 | Integer Modulo Reduction (MOD_AMODN) |
| | | A | 0 | 00107 | |
| MOD_INV | Integer Modular Inversion | B | 0 | 00008 | Integer Modular Inversion (MOD_INV) |
| | | A | 0 | 00108 | |
| MOD_R2 | Integer $R^2$ mod N | B | 0 | 0000C | Integer Montgomery Factor Computation (MOD_R2) |
| | | A | 0 | 0010C | |
| MOD_RR | Integer $R_E R_P$ mod p | B | 0 | 0000D | Integer $R_E R_P$ mod P (MOD_RR) |
| | | A | 0 | 0010D | |
| MOD_GCD | Integer Greatest Common Divisor | B | 0 | 0000E | Integer Greatest Common Divisor (MOD_GCD) |
| | | A | 0 | 0010E | |
| PRIME_TEST | Miller_Rabin primality test | B | 0 | 0000F | Miller_Rabin Primality Test (PRIME_TEST) |
| | | A | 0 | 0010F | |
| MOD_SQRT | Integer Modular Square Root | B | 0 | 00017 | Integer Modular Square Root (MOD_SQRT) |
| | | A | 0 | 20017 | |

1. PKHA_MODE_MS concatenated with 0000b concatenated with PKHA_MODE_LS

Arithmetic functions on a binary polynomials (characterestic two) (F2M). All operate in polynomial basis.

**Table 45-7. List of mode values for PKHA Binary Polynomial Arithmetic Functions**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| F2M_ADD | Binary Polynomial Modular Addition | B | 0 | 20002 | Binary Polynomial ($F_{2^m}$) Addition (F2M_ADD) function |
| | | A | 0 | 20102 | |

*Table continues on the next page...*

**Table 45-7. List of mode values for PKHA Binary Polynomial Arithmetic Functions (continued)**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| F2M_MUL<br>F2M_MUL_TEQ | Binary Polynomial Modular Multiplication | B | 0 | 20005 | Binary Polynomial ($F_{2^m}$) Modular Multiplication (F2M_MUL) |
| | | A | 0 | 20105 | |
| | Timing equalized version | B | 1 | 20405 | |
| | | A | 1 | 20505 | |
| F2M_MUL_IM<br>F2M_MUL_IM_TEQ | Binary Polynomial Modular Multiplication with Montgomery Inputs | B | 0 | A0005 | Binary Polynomial ($F_{2^m}$) Modular Multiplication with Montgomery Inputs (F2M_MUL_IM) Function |
| | | A | 0 | A0105 | |
| | Timing equalized version" | B | 1 | A0405 | |
| | | A | 1 | A0505 | |
| F2M_MUL_IM_OM<br>F2M_MUL_IM_OM_TEQ | Binary Polynomial Modular Multiplication with Montgomery Inputs and Output | B | 0 | E0005 | Binary Polynomial ($F_{2^m}$) Modular Multiplication with Montgomery Inputs and Outputs (F2M_MUL_IM_OM) Function |
| | | A | 0 | E0105 | |
| | Timing equalized version | B | 1 | E0405 | |
| | | A | 1 | E0505 | |
| F2M_EXP<br>F2M_EXP_TEQ | Binary Polynomial Modular Exponentiation | B | 0 | 20006 | Binary Polynomial ($F_{2^m}$) Modular Exponentiation (F2M_EXP and F2M_EXP_TEQ) |
| | | A | 0 | 20106 | |
| | Timing equalized version | B | 1 | 20406 | |
| | | A | 1 | 20506 | |
| F2M_AMODN | Binary Polynomial Modular Reduction | B | 0 | 20007 | Binary Polynomial ($F_{2^m}$) Modulo Reduction (F2M_AMODN) |
| | | A | 0 | 20107 | |
| F2M_INV | Binary Polynomial Modular Inversion | B | 0 | 20008 | Binary Polynomial ($F_{2^m}$) Modular Inversion (F2M_INV) |
| | | A | 0 | 20108 | |
| F2M_R2 | Binary Polynomial $R^2$ mod n | B | 0 | 2000C | Binary Polynomial ($F_{2^m}$) $R^2$ Mod N (F2M_R2) Function |
| | | A | 0 | 2010C | |
| F2M_GCD | Binary Polynomial Greatest Common Divisor | B | 0 | 2000E | Binary Polynomial ($F_{2^m}$) Greatest Common Divisor (F2M_GCD) Function |
| | | A | 0 | 2010E | |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

These functions are grouped here because they do not fall into one of the previous categories of PKHA functions.

### 45.7.5.3 PKHA MODE: Elliptic Curve Functions

**NOTE**

Note that the elliptic curve functions with outputs going to the A RAM are identical to those with outputs going to the B RAM. The only difference is the output destination.

Elliptic Curve Functions over a prime field (ECC_MOD), where prime p > 3.

**Table 45-8. List of mode values for Prime Field ($F_p$) Elliptic Curve Arithmetic Functions**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| ECC_MOD_ADD | ECC prime field point add - affine coordinates | B | 0 | 00009 | ECC $F_p$ Point Add, Affine Coordinates (ECC_MOD_ADD) Function |
| | | A | 0 | 00109 | |
| ECC_MOD_ADD_R2 | ECC prime field point add - affine coordinates, R2 input | B | 0 | 10009 | ECC $F_p$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECC_MOD_ADD_R2) Function |
| | | A | 0 | 10109 | |
| ECC_MOD_DBL | ECC prime field point double - affine coordinates | B | 0 | 0000A | ECC $F_p$ Point Double, Affine Coordinates (ECC_MOD_DBL) Function |
| | | A | 0 | 0010A | |
| ECC_MOD_MUL<br>ECC_MOD_MUL_TEQ | ECC prime field point multiply - affine coordinates | B | 0 | 0000B | ECC $F_p$ Point Multiply, Affine Coordinates (ECC_MOD_MUL and ECC_MOD_MUL_TEQ) Function |
| | | A | 0 | 0010B | |
| | Timing equalized version | B | 1 | 0040B | |
| | | A | 1 | 0050B | |
| ECC_MOD_MUL_R2<br>ECC_MOD_MUL_R2_TEQ | ECC prime field point multiply - affine coordinates, r2 mod n input | B | 0 | 1000B | ECC $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECC_MOD_MUL_R2 and ECC_MOD_MUL_R2_TEQ) Function |
| | | A | 0 | 1010B | |
| | Timing equalized version | B | 1 | 1040B | |
| | | A | 1 | 1050B | |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Elliptic Curve Functions over a binary field (ECC_F2M). All operate in polynomial basis.

**Table 45-9. List of mode values for Binary Field ($F_{2^m}$) Elliptic Curve Arithmetic Functions**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| ECC_F2M_ADD | ECC binary field point add - affine coordinates | B | 0 | 20009 | ECC $F_{2^m}$ Point Add, Affine Coordinates (ECC_F2M_ADD) Function |
| | | A | 0 | 20109 | |
| ECC_F2M_ADD_R2 | ECC binary field point add - affine coordinates, R2 input | B | 0 | 30009 | ECC $F_{2^m}$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECC_F2M_ADD_R2) Function |
| | | A | 0 | 30109 | |
| ECC_F2M_DBL | ECC binary field point double - affine coordinates | B | 0 | 2000A | ECC $F_{2^m}$ Point Double - Affine Coordinates (ECC_F2M_DBL) Function |
| | | A | 0 | 2010A | |
| ECC_F2M_MUL<br>ECC_F2M_MUL_TEQ | ECC binary field point multiply - affine coordinates | B | 0 | 2000B | ECC $F_{2^m}$ Point Multiply, Affine Coordinates (ECC_F2M_MUL and ECC_F2M_MUL_TEQ) Function |
| | | A | 0 | 2010B | |
| | Timing equalized version | B | 1 | 2040B | |
| | | A | 1 | 2050B | |
| ECC_F2M_MUL_R2<br>ECC_F2M_MUL_R2_TEQ | ECC binary field point multiply - affine coordinates, r2 mod n input | B | 0 | 3000B | ECC $F_{2^m}$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECC_F2M_MUL_R2 and ECC_F2M_MUL_R2_TEQ) Function |
| | | A | 0 | 3010B | |
| | Timing equalized version | B | 1 | 3040B | |
| | | A | 1 | 3050B | |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Elliptic Curve Functions over Montgomery-form curve on a prime field (ECM_MOD).

### Table 45-10. List of mode values for Montgomery Elliptic Curve Arithmetic Functions

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| ECM_MOD_MUL | ECC Montgomery curve point multiply - affine x coordinate | B | 0 | 0004B | ECM Modular Multiplication (ECM_MOD_MUL and ECM_MOD_MUL_TEQ) Function |
| | | A | 0 | 0014B | |
| ECM_MOD_MUL_R2_TEQ | Timing equalized version of ECC Montgomery curve point multiply - affine x coordinate | B | 1 | 0044B | |
| | | A | 1 | 0054B | |
| ECM_MOD_MUL_R2 | ECC Montgomery curve - affine x coordinates, r2 mod n input | B | 0 | 1004B | ECM $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECM_MOD_MUL_R2 and ECM_MOD_MUL_R2_TEQ) Function |
| | | A | 0 | 1014B | |
| ECM_MOD_MUL_R2_TEQ | Timing equalized version of ECC Montgomery curve - affine x coordinates, r2 mod n input | B | 1 | 1044B | |
| | | A | 1 | 1054B | |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

Elliptic Curve Functions over a prime field edwards curve (ECT_MOD).

### Table 45-11. List of mode values for Edwards Curve ($F_p$) Arithmetic Functions

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| ECT_MOD_ADD | ECC prime field point add - affine coordinates | B | 0 | 00089 | ECT $F_p$ Point Add, Affine Coordinates (ECT_MOD_ADD) Function |
| | | A | 0 | 00189 | |
| ECT_MOD_ADD_R2 | ECC prime field point add - affine coordinates, R2 input | B | 0 | 10089 | ECT $F_p$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECT_MOD_ADD_R2) Function |
| | | A | 0 | 10109 | |
| ECT_MOD_MUL | ECC prime field point multiply - affine coordinates | B | 0 | 0008B | ECT Modular Multiplication (ECT_MOD_MUL and ECT_MOD_MUL_TEQ) Function |
| | | A | 0 | 0010B | |
| ECT_MOD_MUL_TEQ | Timing equalized version of ECC prime field point multiply - affine coordinates | B | 1 | 0048B | |
| | | A | 1 | 0050B | |
| ECT_MOD_MUL_R2 | ECC prime field point multiply - affine coordinates, r2 mod n input | B | 0 | 1008B | ECT $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECT_MOD_MUL_R2 and ECT_MOD_MUL_R2_TEQ) Function |
| ECT_MOD_MUL_R2_TEQ | | A | 0 | 1018B | |

*Table continues on the next page...*

**Table 45-11. List of mode values for Edwards Curve ($F_p$) Arithmetic Functions (continued)**

| Function name | Brief description | Output reg | Teq | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) | Detailed description |
|---|---|---|---|---|---|
| | Timing equalized version of ECC prime field point multiply - affine coordinates, r2 mod n input | B | 1 | 1048B | |
| | | A | 1 | 1058B | |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

## 45.7.5.4 PKHA MODE: copy memory functions

**Table 45-12. PKHA Mode register, format for copy memory functions**

| 19-17 | 16 | 11-10 | 9-8 | 7-6 | 5-0 |
|---|---|---|---|---|---|
| Source Register | Destination Register | | Source Segment | Destination Segment | Function |
| *PKHA_MODE_MS* | *PKHA_MODE_LS* | | | | |

**Table 45-13. PKHA Mode register, field descriptions for copy memory functions**

| Bits | Description |
|---|---|
| 19-17<br>Source Register | Source Register. Specifies the register to be copied from.<br>If Source Register=000 : A<br>If Source Register=001 : B<br>If Source Register=011 : N<br>All other values are currently reserved. |
| 16<br>Destination Register<br>11-10 | Destination Register. Specifies the register to be copied to.<br>If Destination Register=000 : A<br>If Destination Register=001 : B<br>If Destination Register=010 : E<br>If Destination Register=011 : N<br>All other values are currently reserved.<br>**NOTE:** The source register and destination register fields must not be the same. |
| 9-8<br>Source Segment | Source Segment. Used when copying a register segment to specify which segment in the source register to copy from.<br>If Source Segment=00 : Segment 0<br>If Source Segment=01 : Segment 1<br>If Source Segment=10 : Segment 2<br>If Source Segment=11 : Segment 3 |

*Table continues on the next page...*

## Table 45-13. PKHA Mode register, field descriptions for copy memory functions (continued)

| Bits | Description |
|---|---|
| 7-6<br>Destination Segment | Destination Segment. Used when copying a register segment to specify which segment in the Destination Register to copy to.<br>If Destination Segment=00 : Segment 0<br>If Destination Segment=01 : Segment 1<br>If Destination Segment=10 : Segment 2<br>If Destination Segment=11 : Segment 3<br>**NOTE:** These bits must be zero when E is the destination register. |
| 5-0<br>Function | Function. Indicates which copy function to execute.<br>If Function=010000 : Copy Memory N-Size (copies the same number of words as are in the modulus.)<br>If Function=010001 : Copy Memory SRC-Size (copies the number of words specified in the source's size register) |

This table gives the encodings for the PKHA memory-to-memory copy functions. The top encoding in each cell is for Copy Memory, N-Size, and the bottom encoding is for Copy Memory, Source-Size ( Copy memory, N-Size and Source-Size (COPY_NSZ and COPY_SSZ)).

The encoding is in bits 19-0, including PKHA_MODE (i.e. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS) and reserved bits. (Hex)

## Table 45-14. Mode values for PKHA copy memory functions

| | Destination Memory | | | |
|---|---|---|---|---|
| **Source Memory** | **A** | **B** | **N** | **E** |
| A | | 00410<br>00411 | 00C10<br>00C11 | 00810<br>00811 |
| B | 20010<br>20011 | | 20C10<br>20C11 | 20810<br>20811 |
| N | 60010<br>60011 | 60410<br>60411 | | 60810<br>60811 |

This table gives the encodings for the PKHA memory-to-memory copy functions, when segments are involved. The top encoding in each cell is for Copy Memory, N-Size, and the bottom encoding is for Copy Memory, Source-Size ( Copy memory, N-Size and Source-Size (COPY_NSZ and COPY_SSZ)).

The encoding is in bits 19-0, including PKHA_MODE (i.e. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS) and reserved bits. (Hex)

## Table 45-15. Mode values for PKHA copy memory by segment functions

| | Destination Quadrant | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Source Quadrant** | **A0** | **A1** | **A2** | **A3** | **B0** | **B1** | **B2** | **B3** | **N0** | **N1** | **N2** | **N3** |
| A0 | | | | | 00410 | 00450 | 00490 | 004D0 | 00C10 | 00C50 | 00C90 | 00CD0 |

*Table continues on the next page...*

**Table 45-15. Mode values for PKHA copy memory by segment functions (continued)**

| | Destination Quadrant | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Source Quadrant** | **A0** | **A1** | **A2** | **A3** | **B0** | **B1** | **B2** | **B3** | **N0** | **N1** | **N2** | **N3** |
| | | | | | 00411 | 00451 | 00491 | 004D1 | 00C11 | 00C51 | 00C91 | 00CD1 |
| A1 | | | | | 00510 | 00550 | 00590 | 005D0 | 00D10 | 00D50 | 00D90 | 00DD0 |
| | | | | | 00511 | 00551 | 00591 | 005D1 | 00D11 | 00D51 | 00D91 | 00DD1 |
| A2 | | | | | 00610 | 00650 | 00690 | 006D0 | 00E10 | 00E50 | 00E90 | 00ED0 |
| | | | | | 00611 | 00651 | 00691 | 006D1 | 00E11 | 00E51 | 00E91 | 00ED1 |
| A3 | | | | | 00710 | 00750 | 00790 | 007D0 | 00F10 | 00F50 | 00F90 | 00FD0 |
| | | | | | 00711 | 00751 | 00791 | 007D1 | 00F11 | 00F51 | 00F91 | 00FD1 |
| B0 | 20010 | 20050 | 20090 | 200D0 | | | | | 20C10 | 20C50 | 20C90 | 20CD0 |
| | 20011 | 20051 | 20091 | 200D1 | | | | | 20C11 | 20C51 | 20C91 | 20CD1 |
| B1 | 20110 | 20150 | 20190 | 201D0 | | | | | 20D10 | 20D50 | 20D90 | 20DD0 |
| | 20111 | 20151 | 20191 | 201D1 | | | | | 20D11 | 20D51 | 20D91 | 20DD1 |
| B2 | 20210 | 20250 | 20290 | 202D0 | | | | | 20E10 | 20E50 | 20E90 | 20ED0 |
| | 20211 | 20251 | 20291 | 202D1 | | | | | 20E11 | 20E51 | 20E91 | 20ED1 |
| B3 | 20310 | 20350 | 20390 | 203D0 | | | | | 20F10 | 20F50 | 20F90 | 20FD0 |
| | 20311 | 20351 | 20391 | 203D1 | | | | | 20F11 | 20F51 | 20F91 | 20FD1 |
| N0 | 60010 | 60050 | 60090 | 600D0 | 60410 | 60450 | 60490 | 604D0 | | | | |
| | 60011 | 60051 | 60091 | 600D1 | 60411 | 60451 | 60491 | 604D1 | | | | |
| N1 | 60110 | 60150 | 60190 | 601D0 | 60510 | 60550 | 60590 | 605D0 | | | | |
| | 60111 | 60151 | 60191 | 601D1 | 60511 | 60551 | 60591 | 605D1 | | | | |
| N2 | 60210 | 60250 | 60290 | 602D0 | 60610 | 60650 | 60690 | 606D0 | | | | |
| | 60211 | 60251 | 60291 | 602D1 | 60611 | 60651 | 60691 | 606D1 | | | | |
| N3 | 60310 | 60350 | 60390 | 603D0 | 60710 | 60750 | 60790 | 607D0 | | | | |
| | 60311 | 60351 | 60391 | 603D1 | 60711 | 60751 | 60791 | 607D1 | | | | |

## 45.7.6 PKHA functions

The various PKHA functions are described in the following subsections. The following information applies to all PKHA functions.

- Mode Register bits that may be either 1's or 0's for the given function are represented with *x*.
- For convenience, in all the descriptions below the output is shown as the default B, although the actual output destination can be specified via the Class 1 Mode Register[OutSel] field to be either the B RAM or the A RAM.

- The descriptions specify the output register(s) and any other registers that might be modified. Note that the default output register is still modified but the output is placed into the specified destination register(s).
- Note that any parameter underlined is in Montgomery form (for example, $\underline{A}$ = AR mod N).
- It is important to note that the PKHA mathematical functions work in terms of "digits"; that is, the arithmetic unit is pipelined to work on a digit of data at a time. For PKHA-16 a digit = 16 bits (2 bytes) of data, PKHA-32 a digit = 32 bits (4 bytes) of data, for PKHA-64 a digit = 64 bits (8 bytes), and for PKHA-128 a digit = 128 bits (16 bytes). Therefore, the term 'digit' refers to 16, 32, 64, or 128 bits of data in the input and/or output values used by the PKHA arithmetic unit.

### 45.7.6.1 Clear Memory (CLEAR_MEMORY) function

This function clears the specified registers or quadrants of registers in the PKHA. This includes the A, B, N and E. All registers or quadrants of registers are written with zeros.

A detailed description may be found in PKHA MODE: clear memory function.

**Table 45-16. CLEAR_MEMORY function properties**

| Property | Notes |
|---|---|
| Mode value | *ABEN*_0000_00 *QQ_QQ* 00_0001, with the following restrictions on the combinations of ABEN and QQQQ: At least one of ABEN must be on. If E is on, all Q must be zero. Some example encodings are in the table below. |
| Input | None |
| Output | A = 0, B = 0, E = 0, N = 0, or some quadrant(s) thereof, as specified by ABEN and QQQQ. Each *Q* specifies a quadrant, in order from 3 through 0. |
| Requirements | The Mode Register specifies which registers or quadrants of registers to clear.<br>If no quadrants are selected, then all quadrants of the specified register(s) are cleared. |
| Side effects | None |
| Errors reported | Invalid Mode, if no registers are selected, or E with one or more quadrants is selected |
| Flags set | None |

**Table 45-17. Example mode values for PKHA clear memory functions**

| Function name | Register selects | | | | Quadrant selects | | | | Brief description | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | E | N | 3 | 2 | 1 | 0 | | |
| CLEAR_MEMORY | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Clear registers A, B, E, N | F0001 |
| | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Clear registers A, B, E | E0001 |

*Table continues on the next page...*

**Table 45-17. Example mode values for PKHA clear memory functions (continued)**

| Function name | Register selects | | | | Quadrant selects | | | | Brief description | Bits 19-0, including PKHA_MODE and reserved bits[1] (Hex) |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | E | N | 3 | 2 | 1 | 0 | | |
| | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | Clear registers A, B, N | D0001 |
| | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Clear registers A, E | A0001 |
| | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Clear registers A, N | 90001 |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | Clear registers B, N | 50001 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Clear register B | 40001 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Clear register E | 20001 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Clear register N | 10001 |
| | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Clear quadrants 2 and 3 of register A | 80301 |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Clear quadrant 0 of registers B, N | 50041 |
| | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | Clear quadrant 3 of registers A, B | C0201 |

1. PKHA_MODE_MS concatenated with 0h concatenated with PKHA_MODE_LS

## 45.7.6.2 Arithmetic Functions

### 45.7.6.2.1 Integer Modular Addition (MOD_ADD) function

**Table 45-18. MOD_ADD function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0010 ( output placed in B )<br>0000_0000_000***1***_0000_0010 ( output placed in A ) |
| Input | • N = modulus and data size, any integer<br>• A = first addend, any integer less than N<br>• B = second addend, any integer less than N |
| Output | B (or A, if selected) = (A + B) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are each < N. |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N.<br><br>If (A + B) >= N, N will be subtracted just once from the sum. That is, if (A + B) >= 2N, then the result will not be mod N. |
| Flags set | |

### 45.7.6.2.2 Integer Modular Subtraction (MOD_SUB_1) function

Modular subtraction can be described as follows. If A ≥ B or A = B = 0, then B = A - B. Otherwise, if A < B, then B = A + N - B. The result is always positive and less than N.

**Table 45-19. MOD_SUB_1 function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0011 ( output placed in B )<br>0000_0000_000***1***_0000_0011 ( output placed in A ) |
| Input | • N = modulus, any integer<br>• A = minuend, any integer less than N<br>• B = subtrahend, any integer less than N |
| Output | B (or A, if selected) = (A - B) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are less than N. |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.3 Integer Modular Subtraction (MOD_SUB_2) function

**Table 45-20. MOD_SUB_2 function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0100 ( output placed in B )<br>0000_0000_000***1***_0000_0100 ( output placed in A ) |
| Input | • N = modulus, any integer<br>• B = minuend, any integer less than or equal to N<br>• A = subtrahend, any integer less than or equal to N |
| Output | B (or A, if selected) = (B - A) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are < N |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.4 Integer Modular Multiplication (MOD_MUL)

The (AB) mod N computation is provided to assist in algorithms and protocols where a single modular multiplication is required and not as a chaining of multiplications. In the latter case, Montgomery form multiplication routines (that is, MOD_MUL_IM or MOD_MUL_IM_OM) are more efficient. This function first computes $R^2$ mod N, then multiples one factor to produce AR, then multiplies AR*B to produce AB.

**Table 45-21. MOD_MUL function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0101 ( output placed in B )<br>0000_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A = multiplicand, any integer less than N<br>• B = multiplier, any integer less than N |
| Output | B (or A, if selected) = (AxB) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are < N. |
| Side effects | A and E are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N.<br>• Divide-By-Zero Error is set if the most significant digit of the modulus is all zeros. |
| Flags set | |

### 45.7.6.2.5 Integer Modular Multiplication with Montgomery Inputs (MOD_MUL_IM)

This function takes its inputs, integers, in Montgomery form, multiplies them modulo the value in the N register and returns the result as a field value. To do this, it performs two multiplications: AR*BR => ABR and ABR*1 => AB.

**Table 45-22. MOD_MUL_IM function properties**

| Property | Notes |
|---|---|
| Mode value | 1000_0000_000***0***_0000_0101 ( output placed in B )<br>1000_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A = multiplicand, a value in Montgomery form<br>• B = multiplier, a value in Montgomery form |
| Output | B (or A if selected) = A x B mod N, the non-Montgomery product of the inputs |
| Requirements | • Minimum modulus size = 1 byte |

*Table continues on the next page...*

**Table 45-22. MOD_MUL_IM function properties (continued)**

| Property | Notes |
|---|---|
| | • Maximum modulus size = 512 bytes<br>• A and B are less than modulus N |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.6 Integer Modular Multiplication with Montgomery Inputs and Outputs (MOD_MUL_IM_OM) Function

This function performs the calculation A*B/R mod N, where R is the Montgomery factor for N. This can be used in several ways:

- If one value is a normal value, and the other is R2 mod N, then the result is the normal value converted to Montgomery.
- If A and B are both Montgomery values, then the result is the product of A and B as a Montgomery value.
- If only one of (A,B) is a Montgomery value, then the result is the product as a normal value.
- If one of (A,B) is a Montgomery value and the other is the value one, then the result is Montgomery value converted to a normal value.

**Table 45-23. MOD_MUL_IM_OM function properties**

| Property | Notes |
|---|---|
| Mode value | 1100_0000_000***0***_0000_0101 ( output placed in B )<br>1100_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A = multiplicand, a value in Mongomery format 0 ≤ A < N<br>• B = multiplier, a value in Mongomery format 0 ≤ B < N |
| Output | B (or A, if selected) = (AxB) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

#### 45.7.6.2.7 Integer Modular Exponentiation (MOD_EXP and MOD_EXP_TEQ)

This function is commonly used to perform a single-step RSA operation. It computes $R^2$ mod N and converts A to Montgomery form before beginning the exponentiation. MOD_EXP_TEQ performs the same operation as MOD_EXP but with an added timing equalization security feature. The exponentiation run-time of MOD_EXP_TEQ, for a given modulus and size of exponent, is constant. In general MOD_EXP will run faster than MOD_EXP_TEQ, but will never run slower.

**Table 45-24. MOD_EXP and MOD_EXP_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for MOD_EXP | 0000_0000_0**0**0***0***_0000_0110 ( output placed in B )<br>0000_0000_0**0**0***1***_0000_0110 ( output placed in A ) |
| Mode value for MOD_EXP_TEQ | 0000_0000_0**1**0***0***_0000_0110 ( output placed in B )<br>0000_0000_0**1**0***1***_0000_0110 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A = an integer 0 ≤ A < N<br>• E = exponent, any integer |
| Output | B (or A, if selected) = ($A^E$) mod N, a an integer 0 ≤ A < N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• Maximum key (exponent) size = 512> bytes<br>• A < N |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• A Size Error is set if the size of A is greater than size of N. |
| Flags set | |

#### 45.7.6.2.8 Integer Modular Exponentiation, Montgomery Input (MOD_EXP_IM and MOD_EXP_IM_TEQ) Function

This function is commonly used to perform a single-step RSA operation. It computes $\underline{A}$ = AR (mod N), where R is the Montgomery constant). The input data (base) to be exponentiated must be provided in the Montgomery form. The result will be returned in normal integer (non-Montgomery) representation. MOD_EXP_IM_TEQ performs the same operation as MOD_EXP_IM but with an added timing equalization security feature. The exponentiation run-time of MOD_EXP_IM_TEQ is constant for a given modulus and size of exponent. In general MOD_EXP_IM will run faster than MOD_EXP_IM_TEQ, but will never run slower.

**Table 45-25. MOD_EXP_IM and MOD_EXP_IM_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for MOD_EXP_IM | 1000_0000_0**0**0***0***_0000_0110 ( output placed in B )<br>1000_0000_0**0**0***1***_0000_0110 ( output placed in A ) |
| Mode value for MOD_EXP_IM_TEQ | 1000_0000_0**1**0***0***_0000_0110 ( output placed in B )<br>1000_0000_0**1**0***1***_0000_0110 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A = a value 0 ≤ A < N, in Montgomery form<br>• E = exponent, any integer (normal integer representation) |
| Output | B (or A, if selected) = ($A^E$) mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• Maximum key (exponent) size = 512 bytes<br>• A < N |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• A Size Error is set if the size of A is greater than size of N. |
| Flags set | |

### 45.7.6.2.9 Integer Modular Square Root (MOD_SQRT)

The modular square root function computes output result B, such that ( B x B ) mod N = input A. If no such B result exists, the result will be set to 0 and the PKHA "prime" flag will be set. Input values A and B are limited to a maximum size of 128 bytes. Note that two such square root values may exist. This algorithm will find either one of them, if any exist. The second possible square root (B') can be found by calculating B' = N - B.

**Table 45-26. MOD_SQRT function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0001_0111 ( output placed in B )<br>0000_0000_000***1***_0001_0111 ( output placed in A ) |
| Input | • N0 (N RAM, 1st quadrant) = modulus, any odd integer<br>• A0 (A RAM, 1st quadrant) = input value, for which a square root is to be calculated |
| Output | B0 (or A0, if selected) is calculated such that ( B0 x B0 ) mod N = A mod N. If no such B exists, then B is set to 0. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are < N. |
| Side effects | All of the following PKHA RAM quadrants are modified: E2, E3, N1, N2, N3, B1, B2, B3, A1, A2, A3. |
| Errors reported | N Size Error is set if the size of N is greater than 128. |
| Flags set | |

### 45.7.6.2.10 Integer Modulo Reduction (MOD_AMODN)

MOD_AMODN computes the remainder of A divided by N. A and N can be of any size and it is not required that A > N, but N must be non-zero.

**Table 45-27. MOD_AMODN function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0111 ( output placed in B )<br>0000_0000_000***1***_0000_0111 ( output placed in A ) |
| Input | • N = modulus, any non-zero integer<br>• A = any integer |
| Output | B (or A, if selected) = A mod N, A reduced modulo N |
| Requirements | • N = non-zero value<br>• Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Divide By Zero Error is set if N = 0. |
| Flags set | |

### 45.7.6.2.11 Integer Modular Inversion (MOD_INV)

MOD_INV computes the inverse of A, if an inverse exists. If the modulus, N, is prime, then all values of A, 1 ≤ A < N, are guaranteed to have an inverse mod N. If N is not prime, A may or may not have an inverse. It will have one only if GCD(A, N) == 1.

**Table 45-28. MOD_INV function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1000 ( output placed in B )<br>0000_0000_000***1***_0000_1000 ( output placed in A ) |
| Input | • N = modulus, any non-zero integer<br>• A = any non-zero integer lass than N |
| Output | B (or A, if selected) = $A^{-1}$ mod N, an integer, the multiplicative inverse of A |
| Requirements | • Neither A or N can be zero.<br>• A must be less than N.<br>• Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | A and E are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0, or if there is no inverse. |
| Flags set | None |

### 45.7.6.2.12 Integer Montgomery Factor Computation (MOD_R2)

This function is used to compute a constant to assist in converting operands into the Montgomery residue system representation. The constant $R^2$(mod N) is dependent upon the digit size of the PKHA and the value in N.

MUL, EXP, and ECC functions that do not have "IM" (Montgomery inputs) or an R2 input will internally invoke this routine to determine the constant and do the conversions before other operations.

If the modulus N is a protocol- or system-wide parameter that does not change frequently, such as in ECC operations for a specific curve, save this computed constant, because this routine takes a not-insignificant amount of time to complete.

**Table 45-29. MOD_R2 function properties**

| Property | Notes |
|---|---|
| Mode value | 1000_0000_000***0***_0000_1100 ( output placed in B )<br>1000_0000_000***1***_0000_1100 ( output placed in A ) |
| Input | N = modulus, any odd integer |
| Output | B (or A, if selected) = R2 mod N, where R = $2^{SD}$ where S is size of a digit in bits and D is the number of digits of N; in other words, D = ceiling [sizeof(N) in bits / S] |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even. |
| Flags set | None |

### 45.7.6.2.13 Integer $R_E R_P$ mod P (MOD_RR)

This function is used to compute a constant to assist in converting operands into the Montgomery residue system representation specifically for Chinese Remainder Theorem while performing RSA with a CRT implementation where a modulus E=P x Q, and P and Q are prime numbers. Although labeled $R_E R_P$ mod P, this routine (function) can also compute $R_E R_Q$ mod Q.

To use this routine, the size of the prime number (either P or Q) in bytes must be written into the Data Size Register, and the size of the modulus E = P x Q in bytes must be written into the Key Size Register. Next, the prime modulus (either P or Q) is written into N-memory.

**Table 45-30. MOD_RR function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1101 ( output placed in B )<br>0000_0000_000***1***_0000_1101 ( output placed in A ) |
| Input | • N = modulus P or Q of CRT, an odd integer<br>• Key size = number of bytes of E = P x Q (this size must be given, though content of E itself is not used) |
| Output | B (or A, if selected) = $R_E R_P$ mod P, where $R_E = 2^{SD(E)}$, and $R_P = 2^{SD(P)}$, where S is the size of a digit (in bits). D(E), D(P) are the number of digits of E, P respectively; in other words, D(E or P) = ceiling [size of (E or P) in bits / S] |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• Maximum key size (P-size + Q-size) = 512 bytes<br>• Key size > modulus size |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• Key Size Error is set if size of E = 0 or size of E > 512. |
| Flags set | None |

## 45.7.6.2.14 Integer Greatest Common Divisor (MOD_GCD)

MOD_GCD finds the greatest common divisor of two integers.

**Table 45-31. MOD_GCD function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1110 ( output placed in B )<br>0000_0000_000***1***_0000_1110 ( output placed in A ) |
| Input | • N = any integer. The most-significant digit of N must be non-zero.<br>• A = any integer less than or equal to N |
| Output | B (or A, if selected) = GCD(A,N), an integer less than or equal to A that divides both A and N<br>If the output is placed in B, the MOD_INV result is available in A. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and N may not both be zero |
| Side effects | A is modified |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0. |
| Flags set | |

### 45.7.6.2.15 Miller_Rabin Primality Test (PRIME_TEST)

**Table 45-32. PRIME_TEST function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1111 ( output placed in B )<br>0000_0000_000***1***_0000_1111 ( output placed in A ) |
| Input | • N[1] = Candidate prime integer<br>• A = An initial random seed for the base value of exponentiation; can be any integer 2 < A < N - 2<br>• B = "t" parameter, which is the number of trial runs. By default, it is set at 1 or B[7:0], whichever is greater. Only the lowest byte of the supplied value is used. |
| Output | B (or A, if selected) = 1 if candidate is believed to be prime, otherwise 0 |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and N may not both be zero |
| Side effects | N and A are modified |
| Errors reported | • N Size Error is set if size of N = 0 or size of N > 512.<br>• B Size Error is set if B[7:5] is nonzero and N size > 256.<br>• Divide-By-Zero Error is set if no seed can be found that is in the legal range of 2 < A < N-2. This occurs if N = 1 or N = 3. |
| Flags set | |

1. If the most significant digit of N is zero, the result is always composite, the output is the value zero, regardless of the primality of the value of N.

### 45.7.6.2.16 Binary Polynomial ($F_{2^m}$) Addition (F2M_ADD) function

This function performs binary polynomial modular addition without any modulo reduction, as the value in the N register is ignored. Only its size is used, to determine the size of the result.

This type of addition is the equivalent of a bitwise XOR and this function may be used for that purpose.

This function could as easily be labeled F2M_SUB, as it is mathematically equivalent.

**Table 45-33. F2M_ADD function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_0010 ( output placed in B )<br>0000_0000_000***1***_0000_0010 ( output placed in A ) |
| Input | • Size of N (modulus size)<br>• A = first addend, a binary polynomial<br>• B = second addend, a binary polynomial |
| Output | B (or A, if selected) = A xor B, a binary polynomial |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• The N need not be written, because its contents are ignored, but the size of N must be written. This size is needed because inputs A and B are considered binary polynomials modulo some irreducible polynomial N. |

*Table continues on the next page...*

**Table 45-33. F2M_ADD function properties (continued)**

| Property | Notes |
|---|---|
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.17 Binary Polynomial ($F_{2^m}$) Modular Multiplication (F2M_MUL)

The (AB) mod N computation is provided to assist in algorithms and protocols where a single modular multiplication is required and not as a chaining of multiplications. In the latter case, Montgomery form multiplication routines (that is, F2M_MUL_IM or F2M_MUL_IM_OM) are more efficient. This function first computes $R^2$ mod N, then multiples $A*R^2$ to produce AR, then multiplies AR*B to produce AB.

**Table 45-34. F2M_MUL function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_0101 ( output placed in B )<br>0010_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial<br>• A = multiplicand, a field element<br>• B = multiplier, a field element |
| Output | B (or A, if selected) = (AB) mod N, a field element |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are field elements modulo N. |
| Side effects | A and E are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.18 Binary Polynomial ($F_{2^m}$) Modular Multiplication with Montgomery Inputs (F2M_MUL_IM) Function

This function takes its inputs, binary polynomials, in Montgomery form, multiplies them modulo the value in the N register, used as a reduction polynomial, and returns the result as a field value. To do this, it performs two multiplications: AR*BR => ABR and ABR*1 => AB.

**Table 45-35. F2M_MUL_IM function properties**

| Property | Notes |
|---|---|
| Mode value | 1010_0000_000***0***_0000_0101 ( output placed in B )<br>1010_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, an "odd" binary polynomial of order m<br>• A = multiplicand, a binary polynomial < $2^m$, in Montgomery form<br>• B = multiplicand, a binary polynomial < $2^m$, in Montgomery form |
| Output | B (or A, if selected) = (AxB) mod N, a a binary polynomial < $2^m$, non-Montgomery form |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and B are field elements in Montgomery form and must be modulo reduced by irreducible polynomial N. |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.19 Binary Polynomial ($F_{2^m}$) Modular Multiplication with Montgomery Inputs and Outputs (F2M_MUL_IM_OM) Function

This function performs the calculation A*B/R mod N, where R is the Montgomery factor for N. This can be used in several ways:

- If one value is a normal value, and the other is R2 mod N, then the result is the normal value converted to Montgomery.
- If A and B are both Montgomery values, then the result is the product of A and B as a Montgomery value.
- If only one of (A,B) is a Montgomery value, then the result is the product as a normal value.
- If one of (A,B) is a Montgomery value and the other is the value one, then the result is Montgomery value converted to a normal value.

**Table 45-36. F2M_MUL_IM_OM function properties**

| Property | Notes |
|---|---|
| Mode value | 1110_0000_000***0***_0000_0101 ( output placed in B )<br>1110_0000_000***1***_0000_0101 ( output placed in A ) |
| Input | • N = modulus, an "odd" binary polynomial of order m<br>• A = a binary polynomial 0 < $2^m$, in Montgomery form.<br>• B = a binary polynomial 0 < $2^m$, in Montgomery form. |
| Output | B (or A, if selected) = (AxB) mod N, in Montgomery form |
| Requirements | • Minimum modulus size = 1 byte |

*Table continues on the next page...*

**Table 45-36. F2M_MUL_IM_OM function properties (continued)**

| Property | Notes |
|---|---|
| | • Maximum modulus size = 512 bytes<br>• A and B are field elements in Montgomery form. |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error is set if the size of B is greater than size of N. |
| Flags set | |

### 45.7.6.2.20 Binary Polynomial ($F_{2^m}$) Modular Exponentiation (F2M_EXP and F2M_EXP_TEQ)

This function is similar to MOD_EXP but works on binary polynomials. It is provided mainly to assist in the computation of elliptic curve parameter "c", where c = $b^{2^{m-2}}$ mod $n$) given an elliptic curve parameter "b" and the field-defining polynomial in N. It computes $R^2$ mod N and converts A to Montgomery form before beginning the exponentiation. F2M_EXP_TEQ performs the same operation as F2M_EXP but with an added timing equalization security feature. Its exponentiation run-time, for a given modulus and size of exponent, is constant. In general F2M_EXP will run faster than F2M_EXP_TEQ, but will never run slower.

**Table 45-37. F2M_EXP function properties**

| Property | Notes |
|---|---|
| Mode value for F2M_EXP | 0010_0000_0**00*0***_0000_0110 ( output placed in B )<br>0010_0000_0**00*1***_0000_0110 ( output placed in A ) |
| Mode value for F2M_EXP_TEQ | 0010_0000_0**10*0***_0000_0110 ( output placed in B )<br>0010_0000_0**10*1***_0000_0110 ( output placed in A ) |
| Input | • N = modulus, an "odd" binary polynomial of order m<br>• A = a binary polynomial < $2^m$<br>• E = exponent, any integer |
| Output | B (or A, if selected) = ($A^E$) mod N, a binary polynomial |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• Maximum key (exponent) size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• A Size Error is set if the size of A is greater than size of N. |
| Flags set | |

### 45.7.6.2.21 Binary Polynomial ($F_{2^m}$) Simultaneous Modular Exponentiation (F2M_SML_EXP)

F2M_SML_EXP performs two modular exponentiations on binary polynomials, and multiplies the results. This is faster than doing them separately. It is useful for DSA Verification.

**Table 45-38. F2M_SML_EXP function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0001_0110 ( output placed in B )<br>0000_0000_000***1***_0001_0110 ( output placed in A ) |
| Input | • N = modulus, any odd integer<br>• A0 = a field element, first base<br>• E = an integer, first exponent<br>• A2 = a field element, second base<br>• B = an integer, second exponent |
| Output | B (or A if selected) = $A0^E * A2^B$ mod N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 1/2 RAM<br>• Maximum key (exponent) size and B size = 1/4 RAM size<br>• A0 < N<br>• A2 < N<br>• The values in A0 and A2 must be the same number of bytes, matching the A SIZE register, and should be the same size as N |
| Side effects | A, B, and E are modified. |
| Errors reported | • N size error is set if N size > half RAM<br>• A size error is set if A size > half RAM<br>• B size error is set if B size > quarter-RAM<br>• E size error is set if E size > quarter-RAM |
| Flags set | PIZ is set if the result is zero.<br>PIO is set if the result is one. |

### 45.7.6.2.22 Binary Polynomial ($F_{2^m}$) Modulo Reduction (F2M_AMODN)

F2M_AMODN computes the remainder of A divided by N. This is the equivalent of the MOD_AMODN function applied to a binary polynomial. A and N can be of any size and it is not required that A > N, but N must be non-zero.

**Table 45-39. F2M_AMODN function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_0111 ( output placed in B )<br>0010_0000_000***1***_0000_0111 ( output placed in A ) |
| Input | • N = modulus, any non-zero polynomial<br>• A = any polynomial |
| Output | B (or A, if selected) = A mod N, a polynomial, binary element modulo N |

*Table continues on the next page...*

**Table 45-39. F2M_AMODN function properties (continued)**

| Property | Notes |
|---|---|
| Requirements | • N = non-zero value<br>• Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Divide By Zero Error is set if N = 0. |
| Flags set | |

### 45.7.6.2.23 Binary Polynomial ($F_{2^m}$) Modular Inversion (F2M_INV)

F2M_INV computes the multiplicative inverse of a binary polynomial.

**Table 45-40. F2M_INV function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_1000 ( output placed in B )<br>0010_0000_000***1***_0000_1000 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial<br>• A = a field element |
| Output | B (or A, if selected) = $A^{-1}$ mod N, a field element, the multiplicative inverse of A |
| Requirements | • A is an element of the binary polynomial field.<br>• Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | A and E are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0. |
| Flags set | None |

### 45.7.6.2.24 Binary Polynomial ($F_{2^m}$) $R^2$ Mod N (F2M_R2) Function

This function is used to compute the Montgomery Conversion Factor, which is used to convert operands into the Montgomery residue system representation. The constant $R^2$(mod N) is dependent upon the digit size of the PKHA and the value of N. If this value is not available, then this routine (function) is called to determine the constant before other operations. If N contains a protocol- or system-wide parameter that does not change frequently, such as in ECC operations for a specific curve, save this computed constant, because this routine takes a considerable amount of time to complete.

**Table 45-41. F2M_R2 function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_1100 ( output placed in B )<br>0010_0000_000***1***_0000_1100 ( output placed in A ) |
| Input | N = modulus, an irreducible polynomial |
| Output | B (or A, if selected) = R2 mod N, where R = $2^{SD}$ where S is size of a digit in bits and D is the number of digits of an irreducible polynomial, in other words D = ceiling [sizeof(N) in bits / S] |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes |
| Side effects | None |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• Modulus Even Error is set if N is even. |
| Flags set | None |

### 45.7.6.2.25 Binary Polynomial ($F_{2^m}$) Greatest Common Divisor (F2M_GCD) Function

MOD_GCD finds the greatest common divisor of two binary polynomials.

**Table 45-42. F2M_GCD function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_1110 ( output placed in B )<br>0010_0000_000***1***_0000_1110 ( output placed in A ) |
| Input | • N = any polynomial. The most-significant digit of N must be non-zero.<br>• A = any polynomial with degree less than or equal to N |
| Output | B (or A, if selected) = BINARY_GCD(A,N), a polynomial with degree less than or equal to polynomial A that divides both A and N |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 512 bytes<br>• A and N may not both be zero |
| Side effects | A is modified |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 512.<br>• A Size Error is set if the size of A is greater than size of N.<br>• Divide-By-Zero Error is set if N or A = 0, or if the most significant digit of N = 0. |
| Flags set | |

## 45.7.6.3 Elliptic Curve Functions

#### 45.7.6.3.1 ECC $F_p$ Point Add, Affine Coordinates (ECC_MOD_ADD) Function

ECC_MOD_ADD performs an addition of two points on an elliptic curve. The inputs and output are in affine coordinates.

**Table 45-43. ECC_MOD_ADD function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1001 ( output placed in B )<br>0000_0000_000***1***_0000_1001 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero<br>• [A0, A1] = first addend in affine coordinates<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• [B1, B2] = second addend in affine coordinates<br>• B3 = ignored |
| Output | [B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field and therefore are less than the modulus N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide-By-Zero Error is set if the most-significant digit of N = 0. |
| Flags set | None |

#### 45.7.6.3.2 ECC $F_p$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECC_MOD_ADD_R2) Function

ECC_MOD_ADD_R2 performs an addition of two points on an elliptic curve. The addends are input and the sum is output in affine coordinates. Since ECC_MOD_ADD_R2 has $R^2$ mod N as an additional input, this function is more efficient than ECC_MOD_ADD, which first must compute $R^2$ mod N before performing the addition.

**Table 45-44. ECC_MOD_ADD_R2 function properties**

| Property | Notes |
|---|---|
| Mode value | 0001_0000_000***0***_0000_1001 ( output placed in B )<br>0001_0000_000***1***_0000_1001 ( output placed in A ) |

*Table continues on the next page...*

**Table 45-44. ECC_MOD_ADD_R2 function properties (continued)**

| Property | Notes |
|---|---|
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero<br>• [A0, A1] = first addend point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• [B1, B2] = second addend point in affine coordinates (x,y)<br>• B3 = R2 ($R^2$ mod N) input |
| Output | [B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field formed by N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide-By-Zero Error is set if the most-significant digit of N = 0. |
| Flags set | None |

### 45.7.6.3.3 ECC $F_p$ Point Double, Affine Coordinates (ECC_MOD_DBL) Function

ECC_MOD_DBL computes the double (B + B) of a point B on an elliptic curve. The input and output are in affine coordinates.

**Table 45-45. ECC_MOD_DBL function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_0000_1010 ( output placed in B )<br>0000_0000_000***1***_0000_1010 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero.<br>• [A0, A1, A2] = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• [B1, B2] = input point in affine coordinates<br>• B3 = ignored |
| Output | [B1, B2] (or [A0, A1], if A output selected) = [B1, B2] + [B1, B2], where "+" represents an elliptic-curve point addition. Output is in affine coordinates (x, y). |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field formed by N. |
| Side effects | A0, A2, A3 and B3 are modified. |

*Table continues on the next page...*

**Table 45-45. ECC_MOD_DBL function properties (continued)**

| Property | Notes |
|---|---|
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0. |
| Flags set | |

### 45.7.6.3.4 ECC $F_p$ Point Multiply, Affine Coordinates (ECC_MOD_MUL and ECC_MOD_MUL_TEQ) Function

ECC_MOD_MUL computes the scalar multiplication of a point on an elliptic curve. The input and output are in affine coordinates. ECC_MOD_MUL_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. ECC_MOD_MUL in general will run faster than ECC_MOD_MUL_TEQ, but will never run slower.

**Table 45-46. ECC_MOD_MUL and ECC_MOD_MUL_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECC_MOD_MUL | 0000_0000_0**0**0***0***_0000_1011 ( output placed in B )<br>0000_0000_0**0**0***1***_0000_1011 ( output placed in A ) |
| Mode value for ECC_MOD_MUL_TEQ | 0000_0000_0**1**0***0***_0000_1011 ( output placed in B )<br>0000_0000_0**1**0***1***_0000_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero<br>• E = scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• B1 = ignored<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• The point (A0, A1) must be on the elliptic curve formed by (N, A3, B0). |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N. |

*Table continues on the next page...*

**Table 45-46. ECC_MOD_MUL and ECC_MOD_MUL_TEQ function properties (continued)**

| Property | Notes |
|---|---|
| | • B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0. |
| Flags set | |

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(-k). After the computation is complete, the formula -P= (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinates are the same.

### 45.7.6.3.5 ECC $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECC_MOD_MUL_R2 and ECC_MOD_MUL_R2_TEQ) Function

ECC_MOD_MUL_R2 computes a scalar multiplication of a point on an elliptic curve. The input point and the output point are in affine coordinates. Since ECC_MOD_MUL_R2 has $R^2$ mod N as an additional input, this function is more efficient than ECC_MOD_MUL, which first must compute $R^2$ mod N before performing the multiplication. ECC_MOD_MUL_R2_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. ECC_MOD_MUL_R2 in general will run faster than ECC_MOD_MUL_R2_TEQ, but will never run slower.

**Table 45-47. ECC_MOD_MUL_R2 and ECC_MOD_MUL_R2_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECC_MOD_MUL_R2 | 0001_0000_0**0**0***0***_0000_1011 ( output placed in B )<br>0001_0000_0**0**0***1***_0000_1011 ( output placed in A ) |
| Mode value for ECC_MOD_MUL_R2_TEQ | 0001_0000_0**1**0***0***_0000_1011 ( output placed in B )<br>0001_0000_0**1**0***1***_0000_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero.<br>• E = key, scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• B1 = R2 mod N, pre-computed as described in MOD_R2<br>• B2 = ignored |

*Table continues on the next page...*

**Table 45-47. ECC_MOD_MUL_R2 and ECC_MOD_MUL_R2_TEQ function properties (continued)**

| Property | Notes |
|---|---|
| | • B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 are elements of the prime field formed by the modulus N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0. |
| Flags set | |

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is, (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(- k). After the computation is complete, the formula -P = (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinate is the same.

### 45.7.6.3.6 ECC $F_{2^m}$ Point Add, Affine Coordinates (ECC_F2M_ADD) Function

ECC_F2M_ADD performs an addition of two points on an elliptic curve. The inputs and output are in affine coordinates.

**Table 45-48. ECC_F2M_ADD function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_1001 ( output placed in B )<br>0010_0000_000***1***_0000_1001 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero.<br>• [A0, A1] = first addend in affine coordinates<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "c" parameter, where c = $b^{2^{m-2}}$ mod *n* |

*Table continues on the next page...*

**Table 45-48. ECC_F2M_ADD function properties (continued)**

| Property | Notes |
|---|---|
| | • B1, B2] = second addend in affine coordinates<br>• B3 = ignored |
| Output | P[B1, B2] (or P[A0, A1], if A output selected) = P[A0, A1] + P[B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1, and B2 and elliptic curve parameters A3 and B0 are elements of the binary polynomial field N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide By Zero Error is set if the most significant digit of N = 0.<br>• C is Zero Error if B3 is zero. |
| Flags set | None |

### 45.7.6.3.7 ECC $F_{2^m}$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECC_F2M_ADD_R2) Function

ECC_F2M_ADD_R2 performs an addition of two points on an elliptic curve. The inputs and output are in affine coordinates. Since this function takes $R^2$ mod N as an additional input, ECC_F2M_ADD_R2 is more efficient than ECC_F2M_ADD, which first must compute $R^2$ mod N before performing the addition.

**Table 45-49. ECC_F2M_ADD_R2 function properties**

| Property | Notes |
|---|---|
| Mode value | 0011_0000_000***0***_0000_1001 ( output placed in B )<br>0011_0000_000***1***_0000_1001 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero.<br>• [A0, A1] = first addend in affine coordinates<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "c" parameter, where c = $b^{2^{m-2}}$ mod $n$. Must not be zero.<br>• [B1, B2] = second addend in affine coordinates<br>• B3 = R2 input |
| Output | P[B1, B2] (or P[A0, A1], if A output selected) = P[A0, A1] + P[B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the binary polynomial field. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N > 128. |

*Table continues on the next page...*

**Table 45-49. ECC_F2M_ADD_R2 function properties (continued)**

| Property | Notes |
|---|---|
| | • Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide By Zero Error is set if the most significant digit of N = 0.<br>• C is Zero Error if B3 is zero. |
| Flags set | None |

### 45.7.6.3.8 ECC $F_{2^m}$ Point Double - Affine Coordinates (ECC_F2M_DBL) Function

ECC_F2M_DBL computes the double (B + B) of a point B on an elliptic curve. The input and output are in affine coordinates.

**Table 45-50. ECC_F2M_DBL function properties**

| Property | Notes |
|---|---|
| Mode value | 0010_0000_000***0***_0000_1010 ( output placed in B )<br>0010_0000_000***1***_0000_1010 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero.<br>• A0, A1, A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod $n$<br>• [B1, B2] = input point in affine coordinates<br>• B3 = ignored |
| Output | P[B1, B2] (or P[A0, A1], if A output selected) = P[B1, B2] + P[B1, B2], where "+" represents an elliptic-curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates B1 and B2, and elliptic curve parameters A3 and B0 are elements of the binary polynomial field formed by N. |
| Side effects | A0, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0.<br>• C is Zero Error if B3 is zero. |
| Flags set | |

### 45.7.6.3.9 ECC $F_{2^m}$ Point Multiply, Affine Coordinates (ECC_F2M_MUL and ECC_F2M_MUL_TEQ) Function

ECC_F2M_MUL computes the scalar multiplication of a point on an elliptic curve. The input and output are in affine coordinates. ECC_F2M_MUL_TEQ performs the same operation as ECC_F2M_MUL but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. In general ECC_F2M_MUL will run faster than ECC_F2M_MUL_TEQ, but will never run slower.

**Table 45-51. ECC_F2M_MUL and ECC_F2M_MUL_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECC_F2M_MUL | 0010_0000_0**0**0***0***_0000_1011 ( output placed in B )<br>0010_0000_0**0**0***1***_0000_1011 ( output placed in A ) |
| Mode value for ECC_F2M_MUL_TEQ | 0010_0000_0**1**1***0***_0000_1011 ( output placed in B )<br>0010_0000_0**1**1***1***_0000_1011 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero.<br>• E = key, scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod $n$<br>• B1 = ignored<br>• B2 = ignored<br>• B3 = ignored |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes (irreducible polynomial of maximum degree 1023<br>• Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 must be elements of the binary polynomial field. |
| Side effects | A0, A1 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0.<br>• C is Zero Error if B3 is zero. |
| Flags set | |

The following special cases should be noted:

- For E = 0, this function returns a point at infinity (0,0).
- For E < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = -E). After the multiplication is

complete, the formula -P= (x, x+y) can be used to compute the *y* coordinate of the effective final result; the *x* coordinate stays the same.

### 45.7.6.3.10 ECC $F_{2^m}$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECC_F2M_MUL_R2 and ECC_F2M_MUL_R2_TEQ) Function

ECC_F2M_MUL_R2 computes the scalar multiplication of a point on an elliptic curve. The input and output are in affine coordinates. Since this function takes $R^2$ mod N as an additional input, ECC_F2M_MUL_R2 is more efficient than ECC_F2M_MUL, which first must compute $R^2$ mod N before performing the multiplication.
ECC_F2M_MUL_R2_TEQ performs the same operation as ECC_F2M_MUL_R2 but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. In general ECC_F2M_MUL_R2 will run faster than ECC_F2M_MUL_R2_TEQ, but will never run slower.

**Table 45-52. ECC_F2M_MUL_R2 and ECC_F2M_MUL_R2_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECC_F2M_MUL_R2 | 0011_0000_0**0**0***0***_0000_1011 ( output placed in B )<br>0011_0000_0**0**0***1***_0000_1011 ( output placed in A ) |
| Mode value for ECC_F2M_MUL_R2_TEQ | 0011_0000_0**1**0***0***_0000_1011 ( output placed in B )<br>0011_0000_0**1**0***1***_0000_1011 ( output placed in A ) |
| Input | • N = modulus, an irreducible polynomial. The most significant digit of N must be non-zero.<br>• E = key, scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "c" parameter where c = $b^{2^{m-2}}$ mod *n* and m = degree of polynomial M<br>• B1 = (f2m) R2 mod N, pre-computed as described in F2M_R2MODN (0Eh)<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four, equally size segments of A and B memory locations. |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes (irreducible polynomial of maximum degree 1023<br>• Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 must be elements of the binary polynomial field. |
| Side effects | A0, A1, A2, and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0.<br>• C is Zero Error if B3 is zero. |

*Table continues on the next page...*

**Table 45-52. ECC_F2M_MUL_R2 and ECC_F2M_MUL_R2_TEQ function properties (continued)**

| Property | Notes |
|---|---|
| Flags set | |

The following special cases should be noted:

- For k = 0, this function returns a point at infinity (0,0).
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(- k). After the computation is complete, the formula -P = (x, x+y) can be used to compute the "y" coordinate of the effective final result, and other coordinate is the same.

### 45.7.6.3.11 ECM Modular Multiplication (ECM_MOD_MUL and ECM_MOD_MUL_TEQ) Function

ECM_MOD_MUL computes the scalar multiplication of a point on an elliptic curve in Montgomery form. The input and output are just the x coordinates of the points. ECM_MOD_MUL_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3), constant for a given size of E. ECM_MOD_MUL in general will run faster than ECM_MOD_MUL_TEQ, but will never run slower.

This function computes a point multiplication on a Montgomery curve, using Montgomery values, by means of a Montgomery ladder. At the end of the ladder, P2 = P3 + P1, where P1 is the input and P3 is the result.

**Table 45-53. ECM_MOD_MUL and ECM_MOD_MUL_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECM_MOD_MUL | 0000_0000_0**0**0***0***_0100_1011 ( output placed in B )<br>0000_0000_0**0**0***0***_0100_1011 ( output placed in A ) |
| Mode value for ECM_MOD_MUL_TEQ | 0000_0000_0**1**0***0***_0100_1011 ( output placed in B )<br>0000_0000_0**1**0***1***_0100_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number.<br>• E = scalar multiplier (k), any integer<br>• [A0] = multiplicand, an input point's affine x coordinate<br>• A2 = ignored<br>• A3 = elliptic curve a24 parameter, that is, (A+2)/4<br>• B0 = ignored<br>• B1 = ignored<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |

*Table continues on the next page...*

**Table 45-53. ECM_MOD_MUL and ECM_MOD_MUL_TEQ function properties (continued)**

| Property | Notes |
|---|---|
| Output | • P[B1] (or P[A0], if A output selected) = E x P[A0], where "x" denotes elliptic curve scalar point multiplication. Output is the resulting point's affine x coordinate.<br>• N1 = R2<br>• A1 = X2R, the X the (X,Z) scalar multiplication, with the Montgomery factor. P2x = X2R/Z2R<br>• A2 = Z2R, the Z of the (X,Z) scalar multiplication, with the Montgomery factor<br>• A3 = a24R, the a24 input, with the Montgomery factor<br>• B0 = X3R, the X result of the (X,Z) scalar multiplication, with the Montgomery factor. P3x = X3R/Z3R<br>• B2 = Z3R, the Z result of the (X,Z) scalar multiplication, with the Montgomery factor<br>• B3 = X1R, the x input, with the Montgomery factor |
| Requirements | • Maximum modulus size = 128 bytes<br>• The x in (A0) should be on the elliptic curve formed by (N, A3 and "B"). |
| Side effects | All quadrants of A, B, and N are modified except N0. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N. |
| Flags set | |

### 45.7.6.3.12 ECM $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECM_MOD_MUL_R2 and ECM_MOD_MUL_R2_TEQ) Function

ECM_MOD_MUL_R2 computes a scalar multiplication of a point on an elliptic curve in Montgomery form. The input and output are just the x coordinates of the points. Since ECM_MOD_MUL_R2 has $R^2$ mod N as an additional input, this function is more efficient than ECM_MOD_MUL, which first must compute $R^2$ mod N before performing the multiplication. ECM_MOD_MUL_R2_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3), constant for a given size of E. ECM_MOD_MUL_R2 in general will run faster than ECM_MOD_MUL_R2_TEQ, but will never run slower.

This function computes a point multiplication on a Montgomery curve, using Montgomery values, by means of a Montgomery ladder. At the end of the ladder, P2 = P3 + P1, where P1 is the input and P3 is the result.

**Table 45-54. ECM_MOD_MUL_R2 and ECM_MOD_MUL_R2_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECM_MOD_MUL_R2 | 0001_0000_0**0**0***0***_0100_1011 ( output placed in B )<br>0001_0000_0**0**0***1***_0100_1011 ( output placed in A ) |

*Table continues on the next page...*

**Table 45-54. ECM_MOD_MUL_R2 and ECM_MOD_MUL_R2_TEQ function properties (continued)**

| Property | Notes |
|---|---|
| Mode value for ECM_MOD_MUL_R2_TEQ | 0001_0000_0**1**0***0***_0100_1011 ( output placed in B )<br>0001_0000_0**1**0***1***_0100_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number.<br>• E = scalar multiplier (k), any integer<br>• [A0] = multiplicand, an input point's affine x coordinate<br>• A2 = ignored<br>• A3 = elliptic curve a24 parameter, that is, (A+2)/4<br>• B0 = ignored<br>• B1 = R2 mod N, pre-computed as described in MOD_R2<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |
| Output | • P[B1] (or P[A0], if A output selected) = E x P[A0], where "x" denotes elliptic curve scalar point multiplication. Output is the resulting point's affine x coordinate.<br>• N1 = R2<br>• A1 = X2R, the X the (X,Z) scalar multiplication, with the Montgomery factor. P2x = X2R/Z2R<br>• A2 = Z2R, the Z of the (X,Z) scalar multiplication, with the Montgomery factor<br>• A3 = a24R, the a24 input, with the Montgomery factor<br>• B0 = X3R, the X result of the (X,Z) scalar multiplication, with the Montgomery factor. P3x = X3R/Z3R<br>• B2 = Z3R, the Z result of the (X,Z) scalar multiplication, with the Montgomery factor<br>• B3 = X1R, the x input, with the Montgomery factor |
| Requirements | • Maximum modulus size = 128 bytes<br>• The x in (A0) should be on the elliptic curve formed by (N, A3 and "B"). |
| Side effects | All quadrants of A, B, and N are modified except N0. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N. |
| Flags set | |

### 45.7.6.3.13 ECT Modular Multiplication (ECT_MOD_MUL and ECT_MOD_MUL_TEQ) Function

ECT_MOD_MUL computes the scalar multiplication of a point on an elliptic curve. The input and output are in affine coordinates. ECT_MOD_MUL_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. ECT_MOD_MUL in general will run faster than ECT_MOD_MUL_TEQ, but will never run slower.

**Table 45-55. ECT_MOD_MUL and ECT_MOD_MUL_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECT_MOD_MUL | 0000_0000_0**00*0***>_1000_1011 ( output placed in B )<br>0000_0000_0**00*0***_1000_1011 ( output placed in A ) |
| Mode value for ECT_MOD_MUL_TEQ | 0000_0000_0**10*0***_1000_1011 ( output placed in B )<br>0000_0000_0**10*1***_1000_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero<br>• E = scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" *(or should this be "d")* parameter<br>• B1 = ignored<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• The point (A0, A1) must be on the elliptic curve formed by (N, A3, B0). |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0. |
| Flags set | |

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(-k). After the computation is complete, the formula -P= (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinates are the same.

### 45.7.6.3.14 ECT $F_p$ Point Multiply, $R^2$ Mod N Input, Affine Coordinates (ECT_MOD_MUL_R2 and ECT_MOD_MUL_R2_TEQ) Function

ECT_MOD_MUL_R2 computes a scalar multiplication of a point on an elliptic curve. The input point and the output point are in affine coordinates. Since ECT_MOD_MUL_R2 has $R^2$ mod N as an additional input, this function is more

efficient than ECT_MOD_MUL, which first must compute $R^2$ mod N before performing the multiplication. ECT_MOD_MUL_R2_TEQ computes the same function, but with an added timing equalization security feature. Its computation run-time is, for a given curve (N, A3, B0), constant for a given size of E. ECT_MOD_MUL_R2 in general will run faster than ECT_MOD_MUL_R2_TEQ, but will never run slower.

**Table 45-56. ECT_MOD_MUL_R2 and ECT_MOD_MUL_R2_TEQ function properties**

| Property | Notes |
|---|---|
| Mode value for ECT_MOD_MUL_R2 | 0001_0000_0**00*0***_1000_1011 ( output placed in B )<br>0001_0000_0**00*1***_1000_1011 ( output placed in A ) |
| Mode value for ECT_MOD_MUL_R2_TEQ | 0001_0000_0**10*0***_1000_1011 ( output placed in B )<br>0001_0000_0**10*1***_1000_1011 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero.<br>• E = key, scalar multiplier (k), any integer<br>• [A0, A1] = multiplicand, an input point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• B1 = R2 mod N, pre-computed as described in MOD_R2<br>• B2 = ignored<br>• B3 = ignored<br>• A0-A3 and B0-B3 are four equal-size segments of A and B memory locations. |
| Output | • P[B1, B2] (or P[A0, A1], if A output selected) = E x P[A0, A1], where "x" denotes elliptic curve scalar point multiplication. Output is in affine coordinates (x,y).<br>• B0 = undefined<br>• B3 = undefined |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0 and A1 and elliptic curve parameters A3 and B0 are elements of the prime field formed by the modulus N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Key Size Error is set if size of E = 0 or size of E > 512.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if size of A is greater than size of N.<br>• B Size Error is set if size of B is greater than size of N.<br>• Divide-by-Zero Error is set if the most significant digit of N = 0. |
| Flags set | |

The following special cases should be noted:

- For k = 0, this function returns a point at infinity; that is, (0,0) if curve parameter "b" is nonzero and (0,1) otherwise.
- For k < 0, (that is, a negative scalar multiplication is required), its absolute value should be provided to the PKHA; that is, k = abs(- k). After the computation is complete, the formula -P = (x,-y) can be used to compute the "y" coordinate of the effective final result, and other coordinate is the same.

#### 45.7.6.3.15 ECT $F_p$ Point Add, Affine Coordinates (ECT_MOD_ADD) Function

ECT_MOD_ADD performs an addition of two points on an elliptic curve. The inputs and output are in affine coordinates.

**Table 45-57. ECT_MOD_ADD function properties**

| Property | Notes |
|---|---|
| Mode value | 0000_0000_000***0***_1000_1001 ( output placed in B )<br>0000_0000_000***1***_1000_1001 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero<br>• [A0, A1] = first addend in affine coordinates<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• [B1, B2] = second addend in affine coordinates<br>• B3 = ignored |
| Output | [B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field and therefore are less than the modulus N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide-By-Zero Error is set if the most-significant digit of N = 0. |
| Flags set | None |

#### 45.7.6.3.16 ECT $F_p$ Point Add, Affine Coordinates, $R^2$ Mod N Input (ECT_MOD_ADD_R2) Function

ECT_MOD_ADD_R2 performs an addition of two points on an elliptic curve. The addends are input and the sum is output in affine coordinates. Since ECT_MOD_ADD_R2 has $R^2$ mod N as an additional input, this function is more efficient than ECT_MOD_ADD, which first must compute $R^2$ mod N before performing the addition.

**Table 45-58. ECT_MOD_ADD_R2 function properties**

| Property | Notes |
|---|---|
| Mode value | 0001_0000_000***0***_1000_1001 ( output placed in B ) |

*Table continues on the next page...*

**Table 45-58. ECT_MOD_ADD_R2 function properties (continued)**

| Property | Notes |
|---|---|
| | 0001_0000_000***1***_1000_1001 ( output placed in A ) |
| Input | • N = modulus, a prime number. The most significant digit of N must be non-zero The most significant digit of N must be non-zero<br>• [A0, A1] = first addend point in affine coordinates (x,y)<br>• A2 = ignored<br>• A3 = elliptic curve "a" parameter<br>• B0 = elliptic curve "b" parameter<br>• [B1, B2] = second addend point in affine coordinates (x,y)<br>• B3 = R2 ($R^2$ mod N) input |
| Output | [B1, B2] (or [A0, A1], if A output selected) = [A0, A1] + [B1, B2], where "+" represents an elliptic curve point addition. Output is in affine coordinates. |
| Requirements | • Minimum modulus size = 1 byte<br>• Maximum modulus size = 128 bytes<br>• Point coordinates A0, A1, B1 and B2, and elliptic curve parameters A3 and B0 are elements of the prime field formed by N. |
| Side effects | A0, A1, A2, A3 and B3 are modified. |
| Errors reported | • Data Size Error is set if size of N = 0 or size of N > 128.<br>• Modulus Even Error is set if N is even.<br>• A Size Error is set if the size of A is greater than size of N.<br>• B Size Error will be set if size of B is greater than size of N.<br>• Divide-By-Zero Error is set if the most-significant digit of N = 0. |
| Flags set | None |

### 45.7.6.4 Copy memory, N-Size and Source-Size (COPY_NSZ and COPY_SSZ)

These functions copy data from a PKHA register (or register quadrant) specified as a source, to another PKHA register (or register quadrant) specified as a destination. COPY NSZ copies the amount of data specified by the N Size register. COPY_SSZ copies the amount of data specified in the source register's size register. The source and destination are specified in the Mode Value. The source can be A, B or N. The destination can be A, B, E or N, but not the same as the source.

In a quadrant copy, when NSZ/SSZ exceeds the length of a quadrant, the copy will carry on into the next (higher-numbered) quadrant(s).

When the copy operation has completed, the destination register's size register will be updated to contain the number of bytes copied.

**Table 45-59. COPY_NSZ and COPY_SSZ function properties**

<table>
<tr><th>Property</th><th>Notes</th></tr>
<tr><td>Mode value</td><td>
<table>
<tr><th>Bits 19:17</th><th>Bits 16,11,10</th><th>Bits 9:8</th><th>Bits 7:6</th><th>Bits 5:0</th></tr>
<tr><th>Source Register</th><th>Destination Register</th><th>Source Segment</th><th>Destination Segment</th><th>Function Code</th></tr>
<tr><td>000 = A Register</td><td>000 = A Register</td><td>00 = Segment 0</td><td>00 = Segment 0</td><td rowspan="4">01_0000 = Copy_NSZ<br><br>01_0001 = Copy_SSZ</td></tr>
<tr><td>001 = B Register</td><td>001 = B Register</td><td>01 = Segment 1</td><td>01 = Segment 1</td></tr>
<tr><td>011 = N Register</td><td>011 = N Register</td><td>10 = Segment 2</td><td>10 = Segment 2</td></tr>
<tr><td></td><td>010 = E Register</td><td>11 = Segment 3</td><td>11 = Segment 3</td></tr>
<tr><td>other values reserved</td><td>other values reserved</td><td></td><td></td><td></td></tr>
</table>
1. If the destination register is E, the source and destination seqments must be 00b.
</td></tr>
<tr><td>Input</td><td>None</td></tr>
<tr><td>Output</td><td>None</td></tr>
<tr><td>Requirements</td><td>For Copy_NSZ, the N-size Register must contain a valid value.<br>For Copy_SSZ, the source register's size register must contain a valid value.</td></tr>
<tr><td>Side effects</td><td>The destination register's size register is updated to the number of bytes copied.</td></tr>
<tr><td>Errors reported</td><td>None</td></tr>
<tr><td>Flags set</td><td>None</td></tr>
</table>

1. If the destination register is E, the source and destination seqments must be 00b.

## 45.7.7 Special values for common ECC domains

Software can sometimes use the PKHA more effectively if the Montgomery Conversion Factor ($R^2$ mod N) is either provided or previously used to convert other inputs into Montgomery form. For convenience, the conversion factors for common ECC domains have been computed and published here. Some of the other domain values are provided to aid in definite identification of the domain, in the case that the name is not found or is not an exact match.

The following tables give these values for the *q* and *r* modulus values found in ECC domains. These associated Montgomery values are dependent upon the PKHA digit size (16, 32, 64, 128). These tables are for a PKHA with a 16-bit digit.

ECC $F_{2^m}$ requires a *c* (also called *b'*) parameter for the elliptic curve in place of the *b* value. Table 45-61 provides these values in addition to the Montgomery values. The *b'* values are universal and do not change with PKHA digit size.

The following variable definitions apply to both tables. Variable names (*q, r, b, c*) follow the conventions of IEEE Std 1363.

**Name**

The names in these tables are associated with, or named in, various published standards. Neither the names nor the domains are guaranteed to be complete. Two values of the domain parameters are provided for purposes of identification.

- Those beginning with "P-", "K-", and "B-" are in FIPS 186 from NIST, found at www.csrc.nist.gov
- Those beginning with "ansix9" are names from ANS X9.62-2005; those beginning with "prime" or "c2pnb" are from an earlier ANSI document
- Those beginning with "sec" are from SEC 2 from the Standards for Efficient Cryptography group, found at www.secg.org
- Those beginning with "wtls" are taken from Wireless Transport Layer Security / Wireless Access Protocol, Version 06-Apr-2001, WAP-261-WTLS-20010406-a. Not all software libraries agree with the mapping of these names to values; care has been taken to identify the values based upon the source documentation.
- Those beginning with "ECDSA", "ECP", "EC2N", "ecp_group", and "Oakley" are from various RFCs found at www.ietf.org
- Those beginning with "GOST" are from the Russian standard GOST R 3410-2001
- Those beginning with "brainpool" are from ECC Brainpool, found at www.ecc-brainpool.org and republished in RFC 5639

**R**

*R* is the Montgomery factor. Its value is $2^{SD}$, where *D* is the PKHA digit size in bits, and *S* is the minimum number of digits needed to hold the modulus. As an example, for a modulus of nine bytes (72 bits), *R* would be

- $2^{80}$ for a PKHA with digit size of 16 bits
- $2^{96}$ for a PKHA with digit size of 32 bits
- $2^{128}$ for a PKHA with digit size of 64 or 128 bits

**q**

This is the *field-defining* value for the elliptic curve. For $F_p$ curves, it is the prime number used as the modulus for all point arithmetic; it is named *p* in some other publications. For $F_{2^m}$ curves, it is the irreducible binary polynomial used as the modulus for all point arithmetic. It is not, as usually defined, $q = 2^m$, i.e. the size of the field.

**L**

This is the number of bytes needed to hold *q* and each of its associated values: *R2modq*, *a*,*b*,*c*, the point coordinates *x* and *y*, the result of an ECDH key agreement, etc.

**R2modq**

This is $R^2 \bmod q$, the Montgomery Conversion Factor when *q* is the modulus.

**r**

This is the (usually prime) number which is the order of *G*, the generator point. It is also usually used as the modulus for the non-ECC-related arithmetic in an ECC primitive. This variable is named *n* in some other publications.

**N**

This is the number of bytes needed to hold *r* and each of its associated values: *R2modr*, private keys, each of the two components of an ECDSA signature, etc.

**R2modr**

This is $R^2 \ mod \ r$, the Montgomery Conversion Factor when *r* is the modulus.

**b / c (b')**

*b* is the coefficient for the $x^0$ (ones) term in an $F_{2^m}$ elliptic curve equation. Its relationship with *c* is $b = c^4$. *c* is sometimes referred to as *b'* in NXP documentation.

**A / a24**

*a24* is the special value derived from the A coefficient for the $y^2$term in a Montgomery-form elliptic curve equation. Its relationship with *A* is *a24 = (A+2)/4*.

The domains in the table are ordered by size.

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| secp112r1 | 14 | 14 | |
| wtls6 | q | | 0xDB7C2ABF62E35E668076BEAD208B<br>4451685225093714772084598273548427 |
| | R2modq | | 0x00030000000000000000000000 |
| | r | | 0xDB7C2ABF62E35E7628DFAC6561C5<br>44516852250937147764918915425489 33 |
| | R2modr | | 0x6A1DD1858198B5BADC6CFD628D0A |
| secp112r2 | 14 | 14 | |
| | q | | 0xDB7C2ABF62E35E668076BEAD208B<br>4451685225093714772084598273548427 |
| | R2modq | | 0x00030000000000000000000000 |
| | r | | 0x36DF0AAFD8B8D7597CA10520D04B<br>1112921306273428674967732714786891 |
| | R2modr | | 0x2C19866CFA502191682A07EBB1DF |
| wtls8 | 14 | 15 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFDE7<br>5192296858534827628530496329219559<br>$2^{112} - 2^9 - 2^4 - 2^3 - 1$ |
| | R2modq | | 0x00000000000000000000046671 |
| | r | | 0x0100000000000001ECEA551AD837E9<br>5192296858534827767273836114360297 |
| | R2modr | | 0x00E074FD104C86569DB6C204A52932 |

*Table continues on the next page...*

## Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| secp128r1 | 16 | 16 | |
| | q | | 0xFFFFFFFDFFFFFFFFFFFFFFFFFFFFFFFF<br>340282366762482138434845932244680310783<br>$2^{128} - 2^{97} - 1$ |
| | R2modq | | 0x00000024000000040000000800000011 |
| | r | | 0xFFFFFFFE0000000075A30D1B9038A115<br>340282366762482138443322565580356624661 |
| | R2modr | | 0x71875047CDD8151626BC6448FADE9BED |
| secp128r2 | 16 | 16 | |
| | q | | 0xFFFFFFFDFFFFFFFFFFFFFFFFFFFFFFFF<br>340282366762482138434845932244680310783<br>$2^{128} - 2^{97} - 1$ |
| | R2modq | | 0x00000024000000040000000800000011 |
| | r | | 0x3FFFFFFF7FFFFFFFBE0024720613B5A3<br>850705916906205346039557219268136605 79 |
| | R2modr | | 0x0EFCA409C09D126A99CD2E9404A3B434 |
| secp160k1<br>ansix9p160k1 | 20 | 21 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFAC73<br>1461501637330902918203684832716283019651637554291<br>$2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^9 - 2^8 - 2^7 - 2^3 - 2^2 - 1$ |
| | R2modq | | 0x0000000000000000000000010000A71A1B44BBA9 |
| | r | | 0x0100000000000000000001B8FA16DFAB9ACA16B6B3<br>1461501637330902918203686915170869725397159163571 |
| | R2modr | | 0x008CCA75EBCDCF2BABDFE44FB4D282030A56ECBB7D |
| secp160r1<br>ansix9p160r1<br>wtls7 | 20 | 21 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7FFFFFFF<br>1461501637330902918203684832716283019653785059327<br>$2^{160} - 2^{31} - 1$ |
| | R2modq | | 0x0000000000000000000000040000001000000001 |
| | r | | 0x0100000000000000000001F4C8F927AED3CA752257<br>1461501637330902918203687197606826779884643492439 |
| | R2modr | | 0x00436AB204A0E626837A98A22C8D8455E5563AADF1 |
| secp160r2<br>ansix9p160r2 | 20 | 21 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFAC73<br>1461501637330902918203684832716283019651637554291<br>$2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^9 - 2^8 - 2^7 - 2^3 - 2^2 - 1$ |
| | R2modq | | 0x0000000000000000000000010000A71A1B44BBA9 |

*Table continues on the next page...*

## Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | r | | 0x0100000000000000000000351EE786A818F3A1A16B |
| | | | 1461501637330902918203685085357192140653176136043 |
| | R2modr | | 0x00D68E04EB76E5A181476A1C23DB8912D0E66BA378 |
| wtls9 | 20 | 21 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC808F |
| | | | 1461501637330902918203684832716283019655932313743 |
| | | | $2^{160} - 2^{17} - 2^{16} - 2^{14} - 2^{13} - 2^{12} - 2^{11} - 2^{10} - 2^{9} - 2^{8} - 2^{6} - 2^{5} - 2^{4} - 1$ |
| | R2modq | | 0x0000000000000000000000000000C3C174FE1 |
| | r | | 0x0100000000000000000001CDC98AE0E2DE574ABF33 |
| | | | 1461501637330902918203687013445034429194588307251 |
| | R2modr | | 0x00519041ADCC3AB9A731EC45CB8499428E993A39C5 |
| brainpoolP160r1 | 20 | 20 | |
| | q | | 0xE95E4A5F737059DC60DFC7AD95B3D8139515620F |
| | | | 1332297598440044874827085558802491743757193798159 |
| | R2modq | | 0x6CF12F81C0CA7EF8FED717E0B333F8D625BC14FF |
| | r | | 0xE95E4A5F737059DC60DF5991D45029409E60FC09 |
| | | | 1332297598440044874827085038830181364212942568457 |
| | R2modr | | 0x2BC73851FC9BE6F69E31FE16FC61D4351FDF90EA |
| brainpoolP160t1 | 20 | 20 | |
| | q | | 0xE95E4A5F737059DC60DFC7AD95B3D8139515620F |
| | | | 1332297598440044874827085558802491743757193798159 |
| | R2modq | | 0x6CF12F81C0CA7EF8FED717E0B333F8D625BC14FF |
| | r | | 0xE95E4A5F737059DC60DF5991D45029409E60FC09 |
| | | | 1332297598440044874827085038830181364212942568457 |
| | R2modr | | 0x2BC73851FC9BE6F69E31FE16FC61D4351FDF90EA |
| P-192 | 24 | 24 | |
| secp192r1 | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF |
| ansix9p192r1 | | | 6277101735386680763835789423207666416083908700390324961279 |
| prime192v1 | | | $2^{192} - 2^{64} - 1$ |
| ECPRGF192Random | R2modq | | 0x0000000000000001000000000000000020000000000000001 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831 |
| | | | 6277101735386680763835789423176059013767194773182842284081 |
| | R2modr | | 0x28BE5677EA0581A24696EA5BBB3A6BEECE66BACCDEB35961 |
| secp192k1 | 24 | 24 | |
| ansix9p192k1 | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFEE37 |
| | | | 6277101735386680763835789423207666416102355444459739541047 |
| | | | $2^{192} - 2^{32} - 2^{12} - 2^{8} - 2^{7} - 2^{6} - 2^{3} - 1$ |

*Table continues on the next page...*

## Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | R2modq | | 0x0000000000000000000000000000000100002392013C4FD1 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFE26F2FC170F69466A74DEFD8D<br>6277101735386680763835789423061264271957123915200845512077 |
| | R2modr | | 0x6A21191C2EC4B2B1F0F4F172195E97E2461C1989250F0702 |
| prime192v2 | 24 | 24 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF<br>6277101735386680763835789423207666416083908700390324961279<br>$2^{192} - 2^{64} - 1$ |
| | R2modq | | 0x0000000000000001000000000000000020000000000000001 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFE5FB1A724DC80418648D8DD31<br>6277101735386680763835789423078825936192100537584385056049 |
| | R2modr | | 0xA4FEB8C277C030E139DA8CFB4E35E1F62814A261001BE8FF |
| prime192v3 | 24 | 24 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF<br>6277101735386680763835789423207666416083908700390324961279<br>$2^{192} - 2^{64} - 1$ |
| | R2modq | | 0x0000000000000001000000000000000020000000000000001 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFF7A62D031C83F4294F640EC13<br>6277101735386680763835789423166314882687165660350679936019 |
| | R2modr | | 0x45BCB42FF1CC05D0194D076B366D09BF0305982367330969 |
| brainpoolP192r1 | 24 | 24 | |
| | q | | 0xC302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297<br>4781668983906166242955001894344923773259119655253013193367 |
| | R2modq | | 0xB6225126EED34F1033BF484602C3FE69E2474C6972C7B21A |
| | r | | 0xC302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1<br>4781668983906166242955001894269038308119863659119834868929 |
| | R2modr | | 0x98769B9CE772102BBF4AFD5DBF53AFF0B4727C80E407E8F8 |
| brainpoolP192t1 | 24 | 24 | |
| | q | | 0xC302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297<br>4781668983906166242955001894344923773259119655253013193367 |
| | R2modq | | 0xB6225126EED34F1033BF484602C3FE69E2474C6972C7B21A |
| | r | | 0xC302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1<br>4781668983906166242955001894269038308119863659119834868929 |
| | R2modr | | 0x98769B9CE772102BBF4AFD5DBF53AFF0B4727C80E407E8F8 |
| P-224<br>secp224r1<br>ansix9p224r1 | 28 | 28 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000001<br>26959946667150639794667015087019630673557916260026308143510066298881 |

*Table continues on the next page...*

## Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| wtls12 | R2modq | | 0x00000000FFFFFFFFFFFFFFFFFFFFFFFE000000000000000000000001 |
| ECPRGF224Random | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFF16A2E0B8F03E13DD29455C5C2A3D |
| | | | 26959946667150639794667015087019625940457807714424391721682722368061 |
| | R2modr | | 0xD4BAA4CF1822BC47B1E979616AD09D9197A545526BDAAE6C3AD01289 |
| secp224k1 | 28 | 29 | |
| ansix9p224k1 | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFE56D |
| | | | 26959946667150639794667015087019630673637144422540572481099315275117 |
| | | | $2^{224} - 2^{32} - 2^{12} - 2^{11} - 2^{9} - 2^{7} - 2^{4} - 2^{1} - 1$ |
| | R2modq | | 0x0000000000000000000000000000000000000010000352602C23069 |
| | r | | 0x0100000000000000000000000001DCE8D2EC6184CAF0A971769FB1F7 |
| | | | 26959946667150639794667015087019640346510327083120074548994958668279 |
| | R2modr | | 0x0004DF3DA0993FF72BB882BD88BBFF3BF810C92B3D43C82C98C348E90E |
| brainpoolP224r1 | 28 | 28 | |
| | q | | 0xD7C134AA264366862A18302575D1D787B09F075797DA89F57EC8C0FF |
| | | | 22721622932454352787552537995910928073340732145944992304435472941311 |
| | R2modq | | 0x0578FD592E6A6CE43FE8A2AA96AF774C43C20E727867CA8064DCD04F |
| | r | | 0xD7C134AA264366862A18302575D0FB98D116BC4B6DDEBCA3A5A7939F |
| | | | 22721622932454352787552537995910923612567546342330757191396560966559 |
| | R2modr | | 0x4A73A6563211A5611E9CAE249F24919B9399652CADDAF8AA486CA401 |
| brainpoolP224t1 | 28 | 28 | |
| | q | | 0xD7C134AA264366862A18302575D1D787B09F075797DA89F57EC8C0FF |
| | | | 22721622932454352787552537995910928073340732145944992304435472941311 |
| | R2modq | | 0x0578FD592E6A6CE43FE8A2AA96AF774C43C20E727867CA8064DCD04F |
| | r | | 0xD7C134AA264366862A18302575D0FB98D116BC4B6DDEBCA3A5A7939F |
| | | | 22721622932454352787552537995910923612567546342330757191396560966559 |
| | R2modr | | 0x4A73A6563211A5611E9CAE249F24919B9399652CADDAF8AA486CA401 |
| prime239v1 | 30 | 30 | |
| | q | | 0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFFF |
| | | | 883423532389192164791648750360308885314476597252960362792450860609699839 |
| | R2modq | | 0x00000000000000000000005000000000005FFFFFFFFFFC000000000008 |
| | r | | 0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFFF9E5E9A9F5D9071FBD1522688909D0B |
| | | | 883423532389192164791648750360308884807550341691627752275345424702807307 |
| | R2modr | | 0x17789087ABE4B1BE15BDEB5DDB90D9E27D9E8C4D1C2BB940279628B08300 |
| prime239v2 | 30 | 30 | |
| | q | | 0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFFF |

*Table continues on the next page...*

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | | | 883423532389192164791648750360308885314476597252960362792450860609699839 |
| | R2modq | | 0x0000000000000000000000050000000000005FFFFFFFFFFFC000000000008 |
| | r | | 0x7FFFFFFFFFFFFFFFFFFFFFFF800000CFA7E8594377D414C03821BC582063<br>883423532389192164791648750360308886392687657546993855147765732451295331 |
| | R2modr | | 0x3CB60034F6CF025EBF73DDE8691B5F3A7A21FC865937164FCB06304ACE1E |
| prime239v3 | 30 | 30 | |
| | q | | 0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFFFFFFFFF8000000000007FFFFFFFFFF<br>883423532389192164791648750360308885314476597252960362792450860609699839 |
| | R2modq | | 0x0000000000000000000000050000000000005FFFFFFFFFFC000000000008 |
| | r | | 0x7FFFFFFFFFFFFFFFFFFFFFFF7FFFF975DEB41B3A6057C3C432146526551<br>883423532389192164791648750360308884771190369765922550517967171058034001 |
| | R2modr | | 0x2441C5CD91500EB94E46F167137CF1B742FF58158145184FF0BE2809E6AD |
| P-256 | 32 | 32 | |
| secp256r1<br>ansix9p256r1<br>prime256v1<br>ECDSA-256<br>ecp_group_19<br>ECPRGF256Random | q | | 0xFFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF<br>115792089210356248762697446949407573530086143415290314195533631308867097853951 |
| | R2modq | | 0x00000004FFFFFFFDFFFFFFFFFFFFFFFEFFFFFFFBFFFFFFFF000000000000000003 |
| | r | | 0xFFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551<br>115792089210356248762697446949407573529996955224135760342422259061068512044369 |
| | R2modr | | 0x66E12D94F3D956202845B2392B6BEC594699799C49BD6FA683244C95BE79EEA2 |
| secp256k1 | 32 | 32 | |
| ansix9p256k1 | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F<br>115792089237316195423570985008687907853269984665640564039457584007908834671663<br>$2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ |
| | R2modq | | 0x000000000000000000000000000000000000000000000001000007A2000E90A1 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141<br>115792089237316195423570985008687907852837564279074904382605163141518161494337 |
| | R2modr | | 0x9D671CD581C69BC5E697F5E45BCD07C6741496C20E7CF878896CF21467D7D140 |

*Table continues on the next page...*

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| brainpoolP256r1 | 32 | 32 | |
| | q | | 0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377<br>76884956397045344220809746629001649093037950200943055203735601445031516197751 |
| | R2modq | | 0x4717AA21E5957FA8A1ECDACD6B1AC8075CCE4C26614D4F4D8CFEDF7BA6465B6C |
| | r | | 0xA9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7<br>76884956397045344220809746629001649092737531784414529538755519063063536359079 |
| | R2modr | | 0x0B25F1B9C32367629B7F25E76C815CB0F35D176A1134E4A0E1D8D8DE3312FCA6 |
| brainpoolP256t1 | 32 | 32 | |
| | q | | 0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377<br>76884956397045344220809746629001649093037950200943055203735601445031516197751 |
| | R2modq | | 0x4717AA21E5957FA8A1ECDACD6B1AC8075CCE4C26614D4F4D8CFEDF7BA6465B6C |
| | r | | 0xA9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7<br>76884956397045344220809746629001649092737531784414529538755519063063536359079 |
| | R2modr | | 0x0B25F1B9C32367629B7F25E76C815CB0F35D176A1134E4A0E1D8D8DE3312FCA6 |
| GOSTR3410-CryptoPro-A | 32 | 32 | |
| | q | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD97<br>115792089237316195423570985008687907853269984665640564039457584007913129639319<br>$2^{256} - 2^9 - 2^6 - 2^5 - 2^3 - 1$ |
| | R2modq | | 0x00000000000000000000000000000000000000000000000000000000000005CF11 |
| | r | | 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C611070995AD10045841B09B761B893<br>115792089237316195423570985008687907853073762908499243225378155805079068850323 |
| | R2modr | | 0x551FE9CB451179DBF74885D08A3714C6FB07F8222E76DD529AC2D7858E79A469 |
| GOSTR3410-CryptoPro-B | 32 | 32 | |
| | q | | 0x8000000000000000000000000000000000000000000000000000000000000C99 |

*Table continues on the next page...*

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | | | 57896044618658097711785492504343953926634992332820282019728792003956564823193 |
| | R2modq | | 0x000000000000000000000000000000000000000000000000000000027ACDC4 |
| | r | | 0x800000000000000000000000000015F700CFFF1A624E5E497161BCC8A198F<br>57896044618658097711785492504343953927102133160255826820068844496087732066703 |
| | R2modr | | 0x09D1D2C4E50824664A2E7E2F6882CF102A3104A7EA43E85529B721F4E6CD7823 |
| GOSTR3410-CryptoPro-C | 32 | 32 | |
| | q | | 0x9B9F605F5A858107AB1EC85E6B41C8AACF846E86789051D37998F7B9022D759B<br>70390085352083305199547718019018437841079516630045180471284346843705633502619 |
| | R2modq | | 0x807A394EDE097652186304212849C07B1017BB39C2D346C5409973B4C427FCEA |
| | r | | 0x9B9F605F5A858107AB1EC85E6B41C8AA582CA3511EDDFB74F02F3A6598980BB9<br>70390085352083305199547718019018437840920882647164081035322601458352298396601 |
| | R2modr | | 0x7AA61B49A49D4759C67E5D0EE96E8ED304FDA8694AFDA24BE94FAAB66ABA180E |
| brainpoolP320r1 | 40 | 40 | |
| | q | | 0xD35E472036BC4FB7E13C785ED201E065F98FCFA6F6F40DEF4F92B9EC7893EC28FCD412B1F1B32E27<br>1763593322239166354161909842446019520889512772719515192772960415288640868802149818095501499903527 |
| | R2modq | | 0xA259BA4A6C2D92525455A964E614D6D21F4C881F30C5B676C2478A8D906978EF994EE88A743B52F9 |
| | r | | 0xD35E472036BC4FB7E13C785ED201E065F98FCFA5B68F12A32D482EC7EE8658E98691555B44C59311<br>1763593322239166354161909842446019520889512772717686063760686124016784784845843468355685258203921 |
| | R2modr | | 0x31EC87C73200B14FE30D35244E6390FE86B330BCAF86C40991C3001BE0E16805679D29DF2513E4CD |
| brainpoolP320t1 | 40 | 40 | |
| | q | | 0xD35E472036BC4FB7E13C785ED201E065F98FCFA6F6F40DEF4F92B9EC7893EC28FCD412B1F1B32E27<br>1763593322239166354161909842446019520889512772719515192772960415288640868802149818095501499903527 |
| | R2modq | | 0xA259BA4A6C2D92525455A964E614D6D21F4C881F30C5B676C2478A8D906978EF994EE88A743B52F9 |
| | r | | 0xD35E472036BC4FB7E13C785ED201E065F98FCFA5B68F12A32D482EC7EE8658E98691555B44C59311 |

*Table continues on the next page...*

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | | | 1763593322239166354161909842446019520889512772717686063760686124016784784845843468355685258203921 |
| | R2modr | | 0x31EC87C73200B14FE30D35244E6390FE86B330BCAF86C40991C3001BE0E16805679D29DF2513E4CD |
| P-384 | 48 | 48 | |
| secp384r1<br>ansix9p384r1<br>ECDSA-384<br>ecp_group_20<br>ECPRGF384Random | q | | 0xFFF

## Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| brainpoolP512r1 | 64 | 64 | |
| | q | | 0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308717D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3<br>89489622076502325516566028151591534221626096440983545113445971872000570104135524399179343041919569427654465303864273459379638943099239285360705345 34607816947 |
| | R2modq | | 0x3C4C9D05A9FF6450202E19402056EECCA16DAA5FD42BFF8319486FD8D5898057E0C19A7783514A2553B7F9BC905AFFD3793FB1302715790549AD144A6158F205 |
| | r | | 0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308705 53E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90069<br>89489622076502325516566028151591534221626096440983545113445971872000570104134185283789817306435249598574513983700292805830942156138820439733543921155441 69 |
| | R2modr | | 0xA794586A718407B095DF1B4C194B2E56723C37A22F16BBDFD7F9CC263B790DE3A6F230C72F0207E83EC64BD033B7627F0886B75895283DDDD2A3681ECDA81671 |
| brainpoolP512t1 | 64 | 64 | |
| | q | | 0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308717D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3<br>89489622076502325516566028151591534221626096440983545113445971872000570104135524399179343041919569427654465303864273459379638943099239285360705345 34607816947 |
| | R2modq | | 0x3C4C9D05A9FF6450202E19402056EECCA16DAA5FD42BFF8319486FD8D5898057E0C19A7783514A2553B7F9BC905AFFD3793FB1302715790549AD144A6158F205 |
| | r | | 0xAADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308705 53E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90069<br>89489622076502325516566028151591534221626096440983545113445971872000570104134185283789817306435249598574513983700292805830942156138820439733543921155441 69 |
| | R2modr | | 0xA794586A718407B095DF1B4C194B2E56723C37A22F16BBDFD7F9CC263B790DE3A6F230C72F0207E83EC64BD033B7627F0886B75895283DDDD2A3681ECDA81671 |
| P-521<br>secp521r1<br>ansix9p521r1<br>ECDSA-521<br>ecp_group_21<br>ECPRGF521Random | 66 | 66 | |
| | q | | 0x01FFFF

**Table 45-60. Special Values for common ECC $F_p$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000000000<br>0000000000000000000000000000000000000000000000000000000000004000 |
| | r | | 0x01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF<br>FFFFFFA51868783BF2F966B7FCC0148F709A5D03BB5C9B8899C47AEBB6FB71E9<br>1386409<br>6864797660130609714981900799081393217269435300143305409394463459185543<br>1833976553942450577463332171975329639963713633211138647686124403803403<br>7280889270700544 9 |
| | R2modr | | 0x01EA0391A39A5B5A3AFE8C44383D2D8E03D1492D0D455BCC6D61A8E567BCC<br>FF3D13D47710A7F95E335838E57ADA6E991171E2EF21D4642A23FB7351C540856<br>C0FBB8 |

**Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| sect113r1 | 15 | 15 | |
| wtls4 | q | | 0x020000000000000000000000201<br>$x^{113} + x^9 + 1$ |
| | R2modq | | 0x00000000000000000001000040000000 |
| | b | | 0xE8BEE4D3E2260744188BE0E9C723 |
| | c | | 0x0173E834AF28EC76CB83BD8DFEB2D5 |
| | r | | 0x0100000000000000D9CCEC8A39E56F<br>5192296858534827689835882578830703 |
| | R2modr | | 0x002D02609ABE76F866BDCE5B3F9BCC |
| sect113r2 | 15 | 15 | |
| | q | | 0x020000000000000000000000201<br>$x^{113} + x^9 + 1$ |
| | R2modq | | 0x00000000000000000001000040000000 |
| | b | | 0x95E9A9EC9B297BD4BF36E059184F |
| | c | | 0x0054D9F03957174A32329167D7FE71 |
| | r | | 0x0100000000000000108789B2496AF93<br>5192296858534827702972497909952403 |
| | R2modr | | 0x00471CB662E29CB41ABC888E16FF49 |
| wtls1 | 15 | 14 | |
| | q | | 0x020000000000000000000000201<br>$x^{113} + x^9 + 1$ |
| | R2modq | | 0x00000000000000000001000040000000 |
| | b | | 0x01 |

*Table continues on the next page...*

**Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | c | | 0x00000000000000000000000000000001 |
| | r | | 0xFFFFFFFFFFFFFFFFDBF91AF6DEA73 |
| | | | 5192296858534827627896703833467507 |
| | R2modr | | 0x000511F09A1AB7E0A600B6A46FA9 |
| sect131r1 | 17 | 17 | |
| | q | | 0x080000000000000000000000000000010D |
| | | | $x^{131} + x^8 + x^3 + x^2 + 1$ |
| | R2modq | | 0x00000000000000000000000040144000000 |
| | b | | 0x0217C05610884B63B9C6C7291678F9D341 |
| | c | | 0x03DB89B405E491160E3B2F07B0CE20B37E |
| | r | | 0x040000000000000000023123953A9464B54D |
| | | | 136112946768375385389393275568536556065 3 |
| | R2modr | | 0x03CE363344739BBCD370FB96ADB81025B9 |
| sect131r2 | 17 | 17 | |
| | q | | 0x080000000000000000000000000000010D |
| | | | $x^{131} + x^8 + x^3 + x^2 + 1$ |
| | R2modq | | 0x00000000000000000000000040144000000 |
| | b | | 0x04B8266A46C55657AC734CE38F018F2192 |
| | c | | 0x07CBB9920D71A48E099C38D71DA6490EB1 |
| | r | | 0x040000000000000000016954A233049BA98F |
| | | | 136112946768375385387953504341281286798 3 |
| | R2modr | | 0x02EBB3EA65BB896320D34767E38BE77465 |
| Oakley 3 | 20 | - | |
| | q | | 0x080000000000000000000000040000000000000001 |
| | | | $x^{155} + x^{62} + 1$ |
| | R2modq | | 0x00000040000000000000000000000000000000400 |
| | b | | 0x07338F |
| | c | | 0x00311000000223A000C4474000088E8000111D1D |
| B-163 | 21 | 21 | |
| ansix9t163r2 | q | | 0x0800000000000000000000000000000000000000C9 |
| sect163r2 | | | $x^{163} + x^7 + x^6 + x^3 + 1$ |
| EC2NGF163Random | R2modq | | 0x00000000000000000000000000000014104000000 |
| | b | | 0x020A601907B8C953CA1481EB10512F78744A3205FD |
| | c | | 0x072C4E1EF7CB2F3A035D33104294159609138BB404 |
| | r | | 0x0400000000000000000000292FE77E70C12A4234C33 |
| | | | 5846006549323611672814742442876390689256843201587 |
| | R2modr | | 0x011E0B46683488BE6C9C55E513CB679874C4E0E0B1 |

*Table continues on the next page...*

**Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| K-163 | 21 | 21 | |
| ansix9t163k1 | q | | 0x0800000000000000000000000000000000000000C9 |
| sect163k1 | | | $x^{163} + x^7 + x^6 + x^3 + 1$ |
| EC2NGF163Koblitz | R2modq | | 0x00000000000000000000000000000014104000000 |
| wtls3 | b | | 0x01 |
| | c | | 0x00000000000000000000000000000000000000000001 |
| | r | | 0x040000000000000000000020108A2E0CC0D99F8A5EF |
| | | | 584600654932361167281474175359844834832911857406 3 |
| | R2modr | | 0x013877C8B9719E20D16A359191E004B3D7105A3DB4 |
| sect163r1 | 21 | 21 | |
| ansix9t163r1 | q | | 0x0800000000000000000000000000000000000000C9 |
| | | | $x^{163} + x^7 + x^6 + x^3 + 1$ |
| | R2modq | | 0x00000000000000000000000000000014104000000 |
| | b | | 0x0713612DCDDCB40AAB946BDA29CA91F73AF958AFD9 |
| | c | | 0x05ED403ED58EB45B1CCECA0F4F61655549861BE052 |
| | r | | 0x03FFFFFFFFFFFFFFFFFFF48AAB689C29CA710279B |
| | | | 584600654932361167281473846509879898130442041129 1 |
| | R2modr | | 0x030E95D72BF45AD7608E54BE73B44B4B1C327B65C1 |
| wtls5 | 21 | 21 | |
| | q | | 0x0800000000000000000000000000000000000000000107 |
| | | | $x^{163} + x^8 + x^2 + x^1 + 1$ |
| | R2modq | | 0x00000000000000000000000000000040054000000 |
| | b | | 0xC9517D06D5240D3CFF38C74B20B6CD4D6F9DD4D9 |
| | c | | 0x0453E1E4B7291F5C2D53CE18483F007081E7EA26EC |
| | r | | 0x0400000000000000000001E60FC8821CC74DAEAFC1 |
| | | | 584600654932361167281474162622639205657383263840 1 |
| | R2modr | | 0x007218FA96704CFBABEA28DE5A57F5A5C73FE73E87 |
| Oakley 4 | 24 | - | |
| | q | | 0x020000000000000000000000000000200000000000000000 1 |
| | | | $x^{185} + x^{69} + 1$ |
| | R2modq | | 0x00000000010000000000000000000000000000000004000 |
| | b | | 0x1EE9 |
| | c | | 0x000000000000300000018000C00C000000638030001C0009 |
| sect193r1 | 25 | 25 | |
| ansix9t193r1 | q | | 0x020000000000000000000000000000000000000000008001 |
| | | | $x^{193} + x^{15} + 1$ |
| | R2modq | | 0x00000000000000000000000000000001000000040000000 |

*Table continues on the next page...*

## Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | b | | 0xFDFB49BFE6C3A89FACADAA7A1E5BBC7CC1C2E5D831478814 |
| | c | | 0x0167B35EB4313F263D0F7A3D5036F0A0A3C980D40E5A053ED2 |
| | r | | 0x0100000000000000000000000C7F34A778F443ACC920EBA49 |
| | | | 627710173538668076383578942326954805369157518605104019719 3 |
| | R2modr | | 0x00555F659B9F0A6812CD5A09619A2E7446C7F16EDAA6C25C24 |
| sect193r2 | 25 | 25 | |
| ansix9t193r2 | q | | 0x020000000000000000000000000000000000000000008001 |
| | | | $x^{193} + x^{15} + 1$ |
| | R2modq | | 0x0000000000000000000000000000000001000000040000000 |
| | b | | 0xC9BB9E8927D4D64C377E2AB2856A5B16E3EFB7F61D4316AE |
| | c | | 0x006989FE6BFE30EDDC3244269F3AAD18D66CF3DB3E3302FAA8 |
| | r | | 0x0100000000000000000000000015AAB561B005413CCD4EE99D5 |
| | | | 627710173538668076383578942331495536243729822227984014382 9 |
| | R2modr | | 0x0021EBDCE9B24356750478EA3C9B8611E749E5446745FCDD4F |
| B-233 | 30 | 30 | |
| sect233r1 | q | | 0x020000000000000000000000000000000000000000400000000000000001 |
| ansix9t233r1 | | | $x^{233} + x^{74} + 1$ |
| EC2NGF233Random | R2modq | | 0x0000000000000000004000000000000000000000000000000000004000 |
| wtls11 | b | | 0x66647EDE6C332C7F8C0923BB58213B333B20E9CE4281FE115F7D8F90AD |
| | c | | 0x0007D5EF4389DFF11ECDBA39C30970D3CE35CEBBA58473F64B4DC0F2686C |
| | r | | 0x010000000000000000000000000013E974E72F8A6922031D2603CFE0D7 |
| | | | 690174634679056378743475586227702555583981273734501355537938363448546 3 |
| | R2modr | | 0x00405345696AB044AA57CDD6D0CC913DB0DBA856F036B34D7EDB8C1383DC |
| K-233 | 30 | 29 | |
| sect233k1 | q | | 0x020000000000000000000000000000000000000000400000000000000001 |
| ansix9t233k1 | | | $x^{233} + x^{74} + 1$ |
| EC2NGF233Koblitz | R2modq | | 0x0000000000000000004000000000000000000000000000000000004000 |
| wtls10 | b | | 0x01 |
| | c | | 0x000000000000000000000000000000000000000000000000000000000001 |
| | r | | 0x80000000000000000000000000069D5BB915BCD46EFB1AD5F173ABDF |
| | | | 345087317339528189371737793113851276057094098886225212632808702474134 3 |
| | R2modr | | 0x193B9A7B59BEBED80293C813EEB5B6D7DB65FF0B4A6545982B006BBF08 |
| sect239k1 | 30 | 30 | |
| ansix9t239k1 | q | | 0x800000000000000000040000000000000000000000000000000000000001 |
| | | | $x^{239} + x^{158} + 1$ |

*Table continues on the next page...*

**Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | R2modq | | 0x2000000000000000000000000000000000000000800000000000000000004 |
| | b | | 0x01 |
| | c | | 0x0000000000000000000000000000000000000000000000000000000000000001 |
| | r | | 0x20000000000000000000000005A79FEC67CB6E91F1C1DA800E478A5<br>2208558830972980411979121875928648149482165613217098488874802192153622 13 |
| | R2modr | | 0x1E805199383E8C975E5EA68E20339652293F129E7CF3AB36516F788CE5A5 |
| B-283<br>sect283r1<br>ansix9t283r1<br>EC2NGF283Random | 36 | 36 | |
| | q | | 0x0800000000000000000000000000000000000000000000000000000000000000000000000010A1<br>$x^{283} + x^{12} + x^7 + x^5 + 1$ |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000000000401100400 |
| | b | | 0x027B680AC8B8596DA5A4AF8A19A0303FCA97FD7645309FA2A581485AF6263E313B79A2F5 |
| | c | | 0x03D8C93D3B0EA81D9294034D7EE3135D0AC5FC8D9CB0276F7211F880F0D81CA4C6E87B38 |
| | r | | 0x03FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEF90399660FC938A90165B042A7CEFADB307<br>7770675568902916283677847627294075626569625924376904889109196526770044277787378692871 |
| | R2modr | | 0x0299ADD3FE013DB2E23755FAA9545A49222D8461643773D41D288BCA1EBB695767D7CA78 |
| K-283<br>sect283k1<br>ansix9t283k1<br>EC2NGF283Koblitz | 36 | 36 | |
| | q | | 0x0800000000000000000000000000000000000000000000000000000000000000000000000010A1<br>$x^{283} + x^{12} + x^7 + x^5 + 1$ |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000000000401100400 |
| | b | | 0x01 |
| | c | | 0x0000000000000000000000000000000000000000000000000000000000000000000000001 |
| | r | | 0x01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9AE2ED07577265DFF7F94451E061E163C61<br>3885337784451458141838923813647037813284811733793061324295874997529815829704422603873 |
| | R2modr | | 0x00868DB4F6424B4B297831F5EBA11AE2B1AC177F33C6C133859892757E41B7CAE8927BDB |
| B-409<br>sect409r1<br>ansix9t409r1 | 52 | 52 | |
| | q | | 0x0200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000001 |

*Table continues on the next page...*

## Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| EC2NGF409Random | | | $x^{409} + x^{87} + 1$ |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000004000 |
| | b<br>c | | 0x21A5C2C8EE9FEB5C4B9A753B7B476B7FD6422EF1F3DD674761FA99D6AC27C8A9A197B272822F6CD57A55AA4F50AE317B13545F<br>0x0149B8B7BEBD9B63653EF1CD8C6A5DD105A2AAAC36FE2EAE43CF28CE1CB7C830C1ECDBFA413AB07FE35A57811AE4F88D30AC63FB |
| | r | | 0x0100000000000000000000000000000000000000000000000001E2AAD6A612F33307BE5FA47C3C9E052F838164CD37D9A21173<br>661055968790248598951915308032771039828404682964281219284648798304157774827374805208143723762179110965979867288366567526771 |
| | R2modr | | 0x0007C24B27E70A941D4738F415F186A66A9FF8783C798E99D4A152BF0CE0C0CC273CDA3D70AA82C43A336EFBE1479034DB8EF936 |
| K-409<br>sect409k1<br>ansix9t409k1<br>EC2NGF409Koblitz | 52 | 51 | |
| | q | | 0x0200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000001<br>$x^{409} + x^{87} + 1$ |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000004000 |
| | b<br>c | | 0x01<br>0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001 |
| | r | | 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE5F83B2D4EA20400EC4557D5ED3E3E7CA5B4B5C83B8E01E5FCF<br>330527984395124299475957654016385519914202341482140609642324395022880711289249191050673258457777458014096366590617731358671 |
| | R2modr | | 0x50DC9B805D4BEBA0701EDA0D529DAD74A3ED9914801EFC3F5D0760180600F3725B714A1C6E7B3C68A06DF3709E9354226F8D6C |
| B-571<br>sect571r1<br>ansix9t571r1<br>EC2NGF571Random | 72 | 72 | |
| | q | | 0x0800000000000000000000000000000000000000000000000000000000000

**Table 45-61. Special Values for common ECC $F_{2^m}$ domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | r | | 0x03FFFF

**Table 45-62. Special Values for common ECM MOD (Montgomery curves) domains when PKHA digit size is 16 bits (continued)**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000000005A4 |
| | A | | 0x076D06 |
| | A24 | | 486662<br>0x01DB42<br>121666 |
| | r | | 0x1000000000000000000000000000000014DEF9DEA2F79CD65812631A5CF5D3ED<br>7237005577332262213973186563042994240857116359379907606001950938285454250989 |
| | R2modr | | 0x0399411B7C309A3DCEEC73D217F5BE65D00E1BA768859347A40611E3449C0F01 |

**Table 45-63. Special Values for common ECT MOD (Edwards curves) domains when PKHA digit size is 16 bits**

| Name | L | N | |
|---|---|---|---|
| | var | | Value (hex, decimal, sums of powers) |
| edwards25519 | 32 | 32 | |
| | q | | 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFED<br>$2^{255}$-19 |
| | R2modq | | 0x0000000000000000000000000000000000000000000000000000000000000005A4 |
| | r | | 0x1000000000000000000000000000000014DEF9DEA2F79CD65812631A5CF5D3ED<br>7237005577332262213973186563042994240857116359379907606001950938285454250989 |
| | R2modr | | 0x0399411B7C309A3DCEEC73D217F5BE65D00E1BA768859347A40611E3449C0F01 |

## 45.8 Application/initialization information

This section discusses how to initialize and use the CAU3 module.

### 45.8.1 Code example

A code fragment is shown below as an example of how the CAU3 is used. This example shows the AES decrypting a single 16-byte block

```
//*****************************************************************************
// AES: Decrypts a single 16-byte block
//   arguments
//          *in         pointer to 16-byte block of input chiphertext
//          *key_sch    pointer to key schedule (44, 52, 60 longwords)
//           nr         number of AES rounds (10, 12, 14 = f(key_schedule))
//          *out        pointer to 16-byte block of output plaintext
//
//   NOTE   Input and output blocks may overlap
//
//   calling convention
//   void   cau_aes_decrypt (const unsigned char *in,
//                           const unsigned char *key_sch,
//                           const int            nr,
//                           unsigned char       *out)

  .equ        p_in,       R4
  .equ        p_key_sch,  R5
  .equ        nr,         R6
  .equ        p_out,      R7
  .equ        key_sch0,   R8
  .equ        key_sch1,   R9
  .equ        key_sch2,   R10
  .equ        key_sch3,   R11
  .equ        nr_shf,     R12

  .global cau_aes_decrypt
cau_aes_decrypt:

// get parameters
  mov         p_in,        R0         // copy *ciphertext to p_in
  mov         p_key_sch,   R1         // copy *input key to p_key_sch
  mov         nr,          R2         // copy number of AES rounds to nr
  mov         p_out,       R3         // copy *output plaintext to p_out

//copy nr into nr_shf
  mov         nr_shf,      nr

// load the 128-bit ciphertext input block into CAU RA0/RA1/RA2/RA3 registers
  ldria       R0, [p_in], 4           // R0 = cipher[0]
  ldria       R1, [p_in], 4           // R1 = cipher[1]
  ldria       R2, [p_in], 4           // R2 = cipher[2]
  ldria       R3, [p_in], 4           // R3 = cipher[3]

// the key_sch pointer (p_key_sch) is adjusted to define the end of the elements
// the adjustment factor = f(nr) and defined by the expression:
// end of key_sch = (4*nr + 4) * 4 = 16*nr + 16 for nr = {10, 12, 14}

  lsli        nr_shf, nr, 4                 // nr_shf =  16*nr = nr<<4

  add         p_key_sch, p_key_sch, nr_shf  // p_key_sch = p_key_sch + nr
  addqi       p_key_sch, p_key_sch, 16      // p_key_sch + 16

// XOR keys
  ldeoru      R3, ([p_key_sch], -4)
  ldeoru      R2, ([p_key_sch], -4)
  ldeoru      R1, ([p_key_sch], -4)
  ldeoru      R0, ([p_key_sch], -4)
  subi        nr, nr, 1                   // form nr - 1 as loop count

aes_decrypt_key128_L0:

// InvShiftRows
  aesir

// InvSubBytes
  aesis R3, R3
  aesis R2, R2
  aesis R1, R1
```

```
  aesis R0, R0

// InvMixColumns
  ldru        key_sch3, ([p_key_sch], -4)
  ldru        key_sch2, ([p_key_sch], -4)
  ldru        key_sch1, ([p_key_sch], -4)
  ldru        key_sch0, ([p_key_sch], -4)

  aesisc      R3, R3, key_sch3
  aesisc      R2, R2, key_sch2
  aesisc      R1, R1, key_sch1
  aesisc      R0, R0, key_sch0

  dbnz        nr, aes_decrypt_key128_L0 // decrement the nr count
                                        // branch to aes_decrypt_key128_L0
                                        //if nr != 0

// InvShiftRows
  aesir

// InvSubBytes
  aesis R3, R3
  aesis R2, R2
  aesis R1, R1
  aesis R0, R0

// XOR keys
  ldeoru      R3, ([p_key_sch], -4)
  ldeoru      R2, ([p_key_sch], -4)
  ldeoru      R1, ([p_key_sch], -4)
  ldeoru      R0, ([p_key_sch], -4)


// store the 128-bit plaintext output block into memory pointed from p_out
  stria       R0, [p_out], 4           // store plaintext[0]
  stria       R1, [p_out], 4           // store plaintext[1]
  stria       R2, [p_out], 4           // store plaintext[2]
  str         R3, ([p_out], 0)         // store plaintext[3]

//stop function
  stopi       CAU3_TKCS_STP_OK         // stop, signaling normal termination
```

## 45.8.2 Assembler equate values

```
@ 3rd Generation Cryptographic Acceleration Unit (CAU3) ISA Assembler
@
@ VERSION: $Id: cau3_assembler_equate_values.dita.rca 1.2 Fri Apr 15 17:09:08 2016 rab001
Experimental $
@

@  BASIC REGISTER "NAMES"

    .equ    R0,     0x00
    .equ    R1,     0x01
    .equ    R2,     0x02
    .equ    R3,     0x03
    .equ    R4,     0x04
    .equ    R5,     0x05
    .equ    R6,     0x06
    .equ    R7,     0x07
    .equ    R8,     0x08
    .equ    R9,     0x09
    .equ    R10,    0x0a
```

```
.equ    R11,    0x0b
.equ    R12,    0x0c
.equ    R13,    0x0d
.equ    R14,    0x0e
.equ    R15,    0x0f
.equ    R16,    0x10
.equ    R17,    0x11
.equ    R18,    0x12
.equ    R19,    0x13
.equ    R20,    0x14
.equ    R21,    0x15
.equ    R22,    0x16
.equ    R23,    0x17
.equ    R24,    0x18
.equ    R25,    0x19
.equ    R26,    0x1a
.equ    R27,    0x1b
.equ    R28,    0x1c
.equ    R29,    0x1d
.equ    R30,    0x1e
.equ    R31,    0x1f

.equ    r0,     0x00
.equ    r1,     0x01
.equ    r2,     0x02
.equ    r3,     0x03
.equ    r4,     0x04
.equ    r5,     0x05
.equ    r6,     0x06
.equ    r7,     0x07
.equ    r8,     0x08
.equ    r9,     0x09
.equ    r10,    0x0a
.equ    r11,    0x0b
.equ    r12,    0x0c
.equ    r13,    0x0d
.equ    r14,    0x0e
.equ    r15,    0x0f
.equ    r16,    0x10
.equ    r17,    0x11
.equ    r18,    0x12
.equ    r19,    0x13
.equ    r20,    0x14
.equ    r21,    0x15
.equ    r22,    0x16
.equ    r23,    0x17
.equ    r24,    0x18
.equ    r25,    0x19
.equ    r26,    0x1a
.equ    r27,    0x1b
.equ    r28,    0x1c
.equ    r29,    0x1d
.equ    r30,    0x1e
.equ    r31,    0x1f

.equ    W0,     0x00
.equ    W1,     0x01
.equ    W2,     0x02
.equ    W3,     0x03
.equ    W4,     0x04
.equ    W5,     0x05
.equ    W6,     0x06
.equ    W7,     0x07
.equ    W8,     0x08
.equ    W9,     0x09
.equ    W10,    0x0a
.equ    W11,    0x0b
.equ    W12,    0x0c
.equ    W13,    0x0d
.equ    W14,    0x0e
```

```
    .equ    W15,    0x0f
    .equ    W16,    0x10
    .equ    W17,    0x11
    .equ    W18,    0x12
    .equ    W19,    0x13
    .equ    W20,    0x14
    .equ    W21,    0x15
    .equ    W22,    0x16
    .equ    W23,    0x17
    .equ    W24,    0x18
    .equ    W25,    0x19
    .equ    W26,    0x1a
    .equ    W27,    0x1b
    .equ    W28,    0x1c
    .equ    W29,    0x1d
    .equ    W30,    0x1e
    .equ    W31,    0x1f

    .equ    w0,     0x00
    .equ    w1,     0x01
    .equ    w2,     0x02
    .equ    w3,     0x03
    .equ    w4,     0x04
    .equ    w5,     0x05
    .equ    w6,     0x06
    .equ    w7,     0x07
    .equ    w8,     0x08
    .equ    w9,     0x09
    .equ    w10,    0x0a
    .equ    w11,    0x0b
    .equ    w12,    0x0c
    .equ    w13,    0x0d
    .equ    w14,    0x0e
    .equ    w15,    0x0f
    .equ    w16,    0x10
    .equ    w17,    0x11
    .equ    w18,    0x12
    .equ    w19,    0x13
    .equ    w20,    0x14
    .equ    w21,    0x15
    .equ    w22,    0x16
    .equ    w23,    0x17
    .equ    w24,    0x18
    .equ    w25,    0x19
    .equ    w26,    0x1a
    .equ    w27,    0x1b
    .equ    w28,    0x1c
    .equ    w29,    0x1d
    .equ    w30,    0x1e
    .equ    w31,    0x1f

    .equ    x0,     0x00
    .equ    x1,     0x01
    .equ    x2,     0x02
    .equ    x3,     0x03
    .equ    x4,     0x04
    .equ    x5,     0x05
    .equ    x6,     0x06
    .equ    x7,     0x07
    .equ    x8,     0x08
    .equ    x9,     0x09
    .equ    x10,    0x0a
    .equ    x11,    0x0b
    .equ    x12,    0x0c
    .equ    x13,    0x0d
    .equ    x14,    0x0e
    .equ    x15,    0x0f

    .equ    X0,     0x00
    .equ    X1,     0x01
```

```
    .equ    X2,     0x02
    .equ    X3,     0x03
    .equ    X4,     0x04
    .equ    X5,     0x05
    .equ    X6,     0x06
    .equ    X7,     0x07
    .equ    X8,     0x08
    .equ    X9,     0x09
    .equ    X10,    0x0a
    .equ    X11,    0x0b
    .equ    X12,    0x0c
    .equ    X13,    0x0d
    .equ    X14,    0x0e
    .equ    X15,    0x0f

    .equ    SP,     0x1e
    .equ    LR,     0x1f
    .equ    PC,     0x20
    .equ    SR,     0x21
    .equ    CMD,    0x22

    .equ    sp,     0x1e
    .equ    lr,     0x1f
    .equ    pc,     0x20
    .equ    sr,     0x21
    .equ    cmd,    0x22

@ external register definitions
    .equ    x_pct,      0x0
    .equ    x_mcfg,     0x1
    .equ    x_cr,       0x2
    .equ    x_sr,       0x3
    .equ    x_key0_w0,  0x8
    .equ    x_key0_w1,  0x9
    .equ    x_key0_w2,  0xa
    .equ    x_key0_w3,  0xb
    .equ    x_key1_w0,  0xc
    .equ    x_key1_w1,  0xd
    .equ    x_key1_w2,  0xe
    .equ    x_key1_w3,  0xf
```

# Chapter 46
# Analog-to-Digital Converter (ADC)

## 46.1 Chip-specific LPADC information

### 46.1.1 LPADC0 channel assignments

The following table shows the channel assignments for the LPADC0 module.

**Table 46-1. LPADC0 channel assignments**

| Channel name | LPADC channel (CMDLn[ADCH]) | Input signal | Direct connection |
|---|---|---|---|
| CH0A | 000000 | LPADC0_SE0 | via LPADC0MUX0 |
| CH1A | 000001 | LPADC0_SE1 | via LPADC0MUX0 |
| CH2A | 000010 | LPADC0_SE2 | via LPADC0MUX0 |
| CH3A | 000011 | LPADC0_SE3 | via LPADC0MUX0 |
| CH4A | 000100 | LPADC0_SE4 | via LPADC0MUX1 |
| CH5A | 000101 | LPADC0_SE5 | via LPADC0MUX1 |
| CH6A | 000110 | LPADC0_SE6 | via LPADC0MUX1 |
| CH7A | 000111 | LPADC0_SE7 | via LPADC0MUX1 |
| CH8A | 001000 | LPADC0_SE8 | via LPADC0MUX2 |
| CH9A | 001001 | LPADC0_SE9 | via LPADC0MUX2 |
| CH10A | 001010 | LPADC0_SE10 | via LPADC0MUX2 |
| CH11A | 001011 | LPADC0_SE11 | via LPADC0MUX2 |
| CH12A | 001100 | LPADC0_SE12 | via LPADC0MUX2 |
| CH13A | 001101 | LPADC0_SE13 | via LPADC0MUX2 |
| CH14A | 001110 | LPADC0_SE14 | via LPADC0MUX2 |
| CH15A | 001111 | LPADC0_SE15 / $V_{REF_OUT}$ | Direct Connection |
| CH16A | 010000 | LPADC0_SE16 / DAC0_OUT | Direct Connection |
| CH17A | 010001 | Reserved | — |
| CH18A | 010010 | LPADC0_SE18 | via LPADC0MUX6 |

*Table continues on the next page...*

## Table 46-1. LPADC0 channel assignments (continued)

| Channel name | LPADC channel (CMDLn[ADCH]) | Input signal | Direct connection |
|---|---|---|---|
| CH19A | 010011 | LPADC0_SE19 | via LPADC0MUX6 |
| CH20A | 010100 | LPADC0_SE20 | via LPADC0MUX6 |
| CH21A | 010101 | LPADC0_SE21 | via LPADC0MUX6 |
| CH22A | 010110 | LPADC0_SE22 | via LPADC0MUX6 |
| CH23A | 010111 | LPADC0_SE23 | via LPADC0MUX6 |
| CH24A | 011000 | Reserved | — |
| CH25A | 011001 | Reserved | — |
| CH26A | 011010 | Reserved | — |
| CH27A | 011011 | Reserved | — |
| CH28A | 011100 | Reserved | — |
| CH29A | 011101 | Reserved | — |
| CH30A | 011110 | Reserved | — |
| CH31A | 011111 | Reserved | — |
| CH0B | 100000 | Reserved | — |
| CH1B | 100001 | Reserved | — |
| CH2B | 100010 | Reserved | — |
| CH3B | 100011 | Reserved | — |
| CH4B | 100100 | Reserved | — |
| CH5B | 100101 | Reserved | — |
| CH6B | 100110 | Reserved | — |
| CH7B | 100111 | Reserved | — |
| CH8B | 101000 | Reserved | — |
| CH9B | 101001 | Reserved | — |
| CH10B | 101010 | Reserved | — |
| CH11B | 101011 | Reserved | — |
| CH12B | 101100 | Reserved | — |
| CH13B | 101101 | Reserved | — |
| CH14B | 101110 | Reserved | — |
| CH15B | 101111 | Reserved | — |
| CH16B | 110000 | $V_{DDIO1}$ | via LPADC0MUX2 |
| CH17B | 110001 | $V_{DDIO2}$ | via LPADC0MUX6 |
| CH18B | 110010 | VBAT_DIV4 | via LPADC0MUX6 |
| CH19B | 110011 | Reserved | — |
| CH20B | 110100 | Reserved | — |
| CH21B | 110101 | $V_{DD_CORE}$ | via LPADC0MUX7 |
| CH22B | 110110 | Reserved | — |
| CH23B | 110111 | Reserved | — |
| CH24B | 111000 | Reserved | — |
| CH25B | 111001 | Reserved | — |

*Table continues on the next page...*

**Table 46-1. LPADC0 channel assignments (continued)**

| Channel name | LPADC channel (CMDLn[ADCH]) | Input signal | Direct connection |
|---|---|---|---|
| CH26B | 111010 | Temperature Sensor | via LPADC0MUX7 |
| CH27B | 111011 | Bandgap[1] | via LPADC0MUX7 |
| CH28B | 111100 | Reserved | — |
| CH29B | 111101 | $V_{REFH}$/LPADC0_SE29 | via LPADC0MUX7 |
| CH30B | 111110 | $V_{REFL}$/LPADC0_SE30 | via LPADC0MUX7 |
| CH31B | 111111 | DCDC | via LPADC0MUX7 |

1. This is the System Power Management (SPM) bandgap 1V reference voltage. Prior to reading from this LPADC channel, ensure that the bandgap buffer is enabled by setting the SPM_CORELPCNFG[BGBEN] bit. Refer to the device data sheet for the bandgap voltage (VBG) specification.

## 46.1.2 Triggers

The LPADC supports both software and hardware triggers. The primary hardware mechanism for triggering the LPADC is the Trigger Multiplexor (TRGMUX). The TRGMUX takes all of the hardware triggers and multiplexes them onto the trigger inputs of the system modules. This allows a broader range of triggers to be used to initiate autonomous action. Because many of the modules which use triggers to initiate actions also have trigger outputs when those actions have completed, sequences that involve multiple peripherals can be created through the proper chaining of the triggers through the TRGMUX.

The following table shows the trigger inputs to the LPADC and where they are generated.

**Table 46-2. LPADC0 trigger inputs**

| Trigger | Source |
|---|---|
| 0 | TRGMUX_LPADC0_trigger0 |
| 1 | TRGMUX_LPADC0_trigger1 |
| 2 | TRGMUX_LPADC0_trigger2 |
| 3 | TRGMUX_LPADC0_trigger3 |

For operation of triggers in different modes, refer to Power Management chapter.

## 46.1.3 Voltage reference options

The LPADC0 supports the following high voltage references which can be configured by :

- Option 1: $V_{REFH}$ - Voltage Reference High
- Option 2: $V_{REF_OUT}$ - Internal Voltage Reference Output
- Option 3: $V_{DDA}$ - Analog Voltage Supply

The high voltage reference source is selected by setting the LPADC0_CFG [REFSEL] field. Option 1 is connected to the $V_{REFH}$ pin. Option 2 is connected to the $V_{REF_OUT}$ pin. Option 3 is connected to the $V_{DDA}$ pin.

The LPADC0 supports the following low voltage references:

- $V_{REFL}$ - Voltage Reference Low ($V_{REFL}$)

### 46.1.4 Functional clock options

The LPADC0 has multiple functional clock sources that can be selected for use during data conversion. These sources includes multiple external clocks and an internal clock source. The clocks are controlled by the System Clock Generator (SCG) and the Peripheral Clock Control (PCC) modules. For information on configuring and enabling these external clocks, please see the SCG chapter. For information on selecting and enabling the LPADC's internal and external functional clocks, please refere to the PCC chapter.

The internal clock source is a 2 MHz Internal Resistor Capacitor Oscillator (LPADCIRC) and is provided so that the LPADC can function in low-power modes when all other clock sources are disabled. The 2 MHz LPADCIRC is controlled through the internal and selected by PCC.

### 46.1.5 Direct Memory Access (DMA) support

Customers may require continuous sampling of the LPADC for certain applications. As the sample rate increases, a larger load is placed on the processor core to move the data from the LPADC peripheral to memory. To eliminate this loading, the LPADC supports data transfers from the LPADC FIFOs to the system memory through DMA. A DMA request can be enabled to trigger when the FIFO exceeds a programmable watermark.

## 46.2 Introduction

The 12-bit analog-to-digital converter (LPADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

**NOTE**

For the chip specific modes of operation, see the power management information of the device.

**Note**

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding 0b, and decimal values have no preceding character.

## 46.2.1 Features

Following are the features of the LPADC module.

- Linear successive approximation algorithm
  - single-ended operation with 12-bit resolution
- Channel support for up to 64 analog input channels for conversion of external pins and from internal sources
  - Measurement of on-chip analog sources such as DAC, temperature sensor or bandgap
- Configurable analog input sample time
- Configurable speed options to accommodate operation in low power modes of the device.
- Trigger detect with up to 4 trigger sources with priority level configuration. Software or hardware trigger option for each.
- 15 command buffers allow independent options selection and channel sequence scanning.
- Automatic compare for less-than, greater-than, within range, or out-of-range with "store on true" and "repeat until true" options
- 16-entry conversion result data FIFO with configurable watermark and overflow detection
- Interrupt, DMA or polled operation

## 46.2.2 Block diagram

The following figure is the LPADC module block diagram.

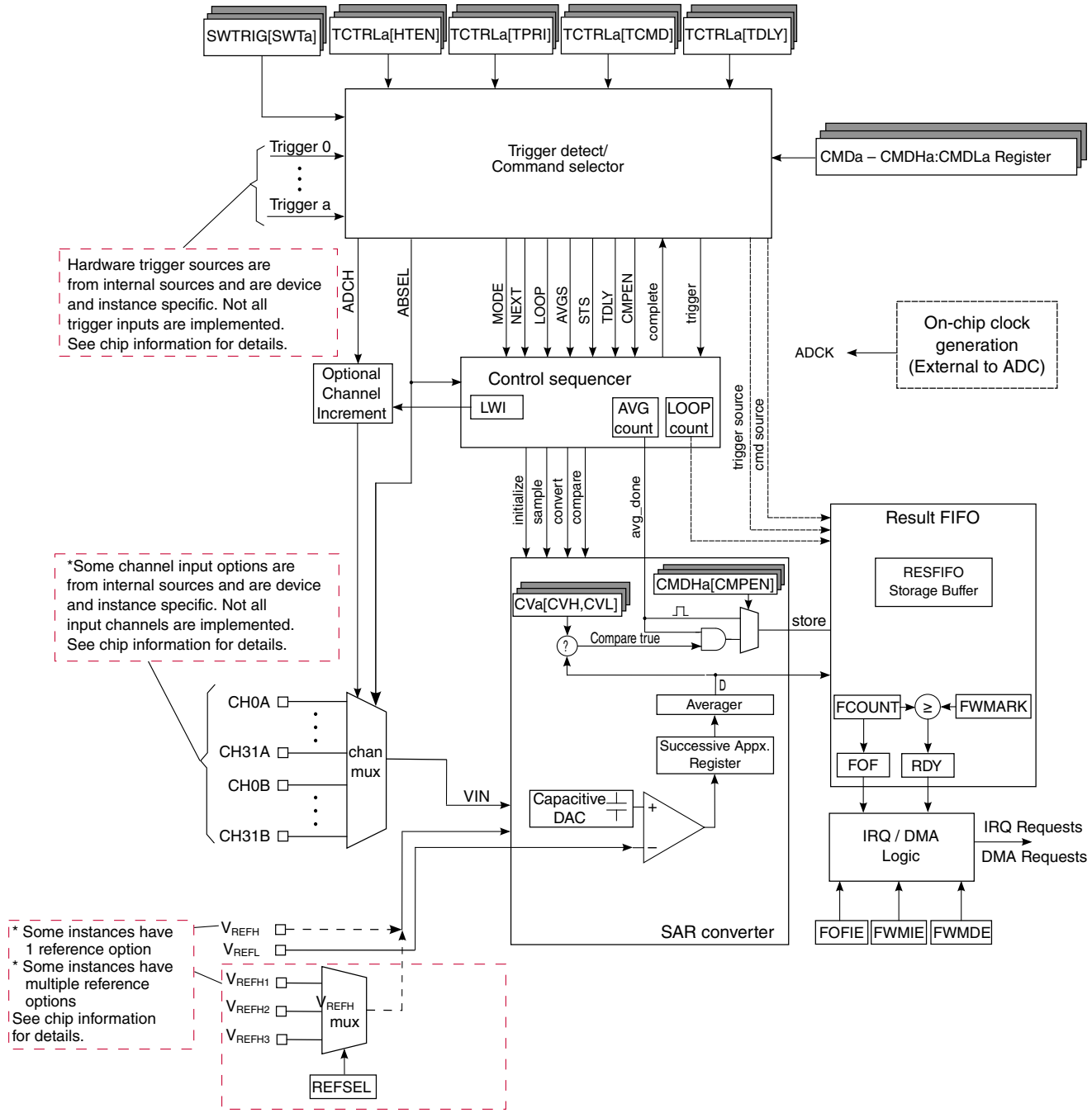



**Figure 46-1. LPADC block diagram**

## 46.2.3 Modes of operation

The table below shows the operation of LPADC module in various chip modes.

**Table 46-3. Chip modes supported by the ADC module**

| Chip mode | LPADC Operation |
|---|---|
| Run | Normal operation |
| Stop/Wait | Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is clear and the module is using an external or internal clock source which remains operating during stop/wait modes. When the CTRL[DOZEN] bit is set, the module will wait for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry. |
| Low Leakage Stop | The Doze Enable (CTRL[DOZEN]) bit is ignored and the ADC will wait for the current transfer to complete any pending operation before acknowledging low-leakage mode entry. |

## 46.3 Signal descriptions

The LPADC module supports up to 64 analog channel inputs. The module also requires supply and ground connections.

**Table 46-4. LPADC signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| $V_{DDA}$ | Analog Power Supply | I |
| $V_{SSA}$ | Analog Ground | I |
| CH31A–CH0A | A-side Analog Channel Inputs | I |
| CH31B–CH0B | B-side Analog Channel Inputs | I |

**NOTE**

The voltage reference high ($V_{REFH}$) used by the LPADC is supplied from either an on-chip voltage reference source or from an off-chip source supplied through external pins. $V_{REFL}$ is always from an external pin and must be at the same voltage potential as $V_{SSA}$. See the chip configuration information on the voltage reference options specific to this packaged device.

**NOTE**

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via the CFG[REFSEL] field). See the chip configuration information on the voltage reference options specific to this packaged device.

### 46.3.1 Analog Power ($V_{DDA}$)

The ADC analog portion uses $V_{DDA}$ as its power connection. In some packages, $V_{DDA}$ is connected internally to $V_{DD}$. If externally available, connect the $V_{DDA}$ pin to the same voltage potential as $V_{DD}$. External filtering may be necessary to ensure clean $V_{DDA}$ for good results.

### 46.3.2 Analog Ground ($V_{SSA}$)

The ADC analog portion uses $V_{SSA}$ as its ground connection. In some packages, $V_{SSA}$ is connected internally to $V_{SS}$. If externally available, connect the $V_{SSA}$ pin to the same voltage potential as $V_{SS}$.

### 46.3.3 Analog Channel Inputs (CH*n*A and CH*n*B)

The CMDL*a*[ADCH] and CMDL*a*[ABSEL] bitfields control selection of a input channel. Each ADC command independently makes a channel selection.

**NOTE**

Some of the input channels are from on-chip sources such as temperature sensors and reference voltage sources and may only be connected to individual instances of the ADC module. Some of the input channel options in the bitfield-setting descriptions may not be available for your device. Refer to ADC configuration section in the chip configuration chapter for the channels supported on this device.

## 46.4 LPADC register descriptions

This section describes the LPADC registers.

### 46.4.1 LPADC Memory map

LPADC0 base address: 4004_A000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0708h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0F04_1004h |
| 10h | LPADC Control Register (CTRL) | 32 | RW | 0000_0000h |
| 14h | LPADC Status Register (STAT) | 32 | W1C | 0000_0000h |
| 18h | Interrupt Enable Register (IE) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable Register (DE) | 32 | RW | 0000_0000h |
| 20h | LPADC Configuration Register (CFG) | 32 | RW | 0080_0000h |
| 24h | LPADC Pause Register (PAUSE) | 32 | RW | 0000_0000h |
| 30h | LPADC FIFO Control Register (FCTRL) | 32 | RW | 0000_0000h |
| 34h | Software Trigger Register (SWTRIG) | 32 | WORZ | 0000_0000h |
| 40h | LPADC Offset Trim Register (OFSTRIM) | 32 | RW | 0000_0000h |
| C0h - CCh | Trigger Control Register (TCTRL0 - TCTRL3) | 32 | RW | 0000_0000h |
| 100h | LPADC Command Low Buffer Register (CMDL1) | 32 | RW | 0000_0000h |
| 104h | LPADC Command High Buffer Register (CMDH1) | 32 | RW | 0000_0000h |
| 108h | LPADC Command Low Buffer Register (CMDL2) | 32 | RW | 0000_0000h |
| 10Ch | LPADC Command High Buffer Register (CMDH2) | 32 | RW | 0000_0000h |
| 110h | LPADC Command Low Buffer Register (CMDL3) | 32 | RW | 0000_0000h |
| 114h | LPADC Command High Buffer Register (CMDH3) | 32 | RW | 0000_0000h |
| 118h | LPADC Command Low Buffer Register (CMDL4) | 32 | RW | 0000_0000h |
| 11Ch | LPADC Command High Buffer Register (CMDH4) | 32 | RW | 0000_0000h |
| 120h | LPADC Command Low Buffer Register (CMDL5) | 32 | RW | 0000_0000h |
| 124h | LPADC Command High Buffer Register (CMDH5) | 32 | RW | 0000_0000h |
| 128h | LPADC Command Low Buffer Register (CMDL6) | 32 | RW | 0000_0000h |
| 12Ch | LPADC Command High Buffer Register (CMDH6) | 32 | RW | 0000_0000h |
| 130h | LPADC Command Low Buffer Register (CMDL7) | 32 | RW | 0000_0000h |
| 134h | LPADC Command High Buffer Register (CMDH7) | 32 | RW | 0000_0000h |
| 138h | LPADC Command Low Buffer Register (CMDL8) | 32 | RW | 0000_0000h |
| 13Ch | LPADC Command High Buffer Register (CMDH8) | 32 | RW | 0000_0000h |
| 140h | LPADC Command Low Buffer Register (CMDL9) | 32 | RW | 0000_0000h |
| 144h | LPADC Command High Buffer Register (CMDH9) | 32 | RW | 0000_0000h |
| 148h | LPADC Command Low Buffer Register (CMDL10) | 32 | RW | 0000_0000h |
| 14Ch | LPADC Command High Buffer Register (CMDH10) | 32 | RW | 0000_0000h |
| 150h | LPADC Command Low Buffer Register (CMDL11) | 32 | RW | 0000_0000h |
| 154h | LPADC Command High Buffer Register (CMDH11) | 32 | RW | 0000_0000h |
| 158h | LPADC Command Low Buffer Register (CMDL12) | 32 | RW | 0000_0000h |
| 15Ch | LPADC Command High Buffer Register (CMDH12) | 32 | RW | 0000_0000h |
| 160h | LPADC Command Low Buffer Register (CMDL13) | 32 | RW | 0000_0000h |
| 164h | LPADC Command High Buffer Register (CMDH13) | 32 | RW | 0000_0000h |
| 168h | LPADC Command Low Buffer Register (CMDL14) | 32 | RW | 0000_0000h |
| 16Ch | LPADC Command High Buffer Register (CMDH14) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 170h | LPADC Command Low Buffer Register (CMDL15) | 32 | RW | 0000_0000h |
| 174h | LPADC Command High Buffer Register (CMDH15) | 32 | RW | 0000_0000h |
| 200h - 20Ch | Compare Value Register (CV1 - CV4) | 32 | RW | 0000_0000h |
| 300h | LPADC Data Result FIFO Register (RESFIFO) | 32 | RO | 0000_0000h |

## 46.4.2 Version ID Register (VERID)

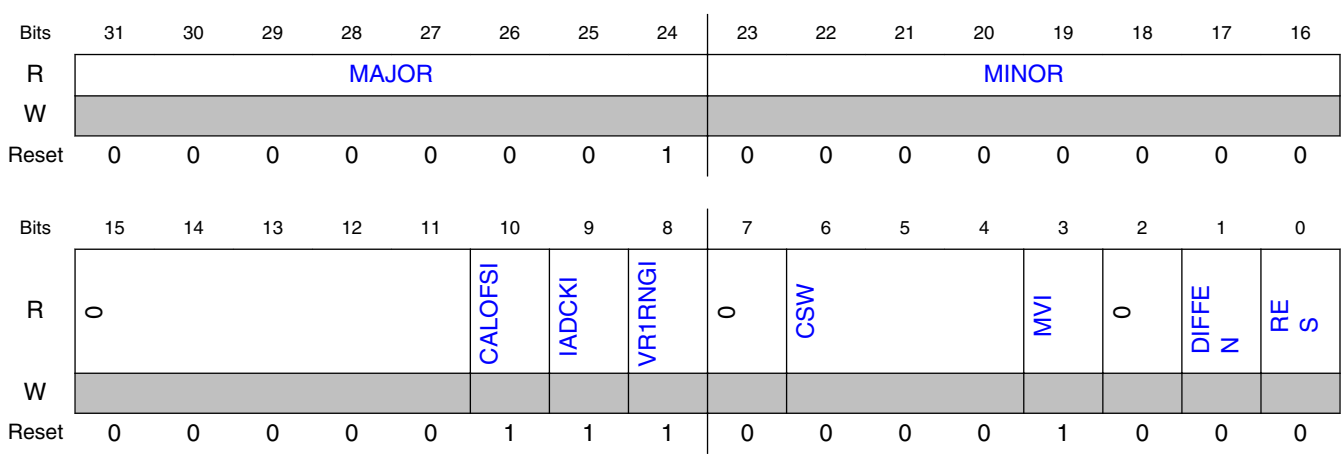
### 46.4.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 46.4.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

### 46.4.2.3 Diagram



### 46.4.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read-only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read-only field returns the minor version number for the module specification. |
| 15-11<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 10<br>CALOFSI | Calibration Offset Function Implemented<br>This read-only field indicates if a calibration function and trimming control register are implemented. When supported, the CFG[CALOFS] and OFSTRIM[OFSTRIM] fields are available in support of this function.<br>0b - Offset calibration and offset trimming not implemented.<br>1b - Offset calibration and offset trimming implemented. |
| 9<br>IADCKI | Internal LPADC Clock implemented<br>This read-only field indicates if this implementation of the LPADC includes an internal clock source. When supported, the CFG[ADCKEN] field is available and documented for enabling the clock source. When available, this clock source is used in clock selection logic external to the module for use within the system.<br>0b - Internal clock source not implemented.<br>1b - Internal clock source (and CFG[ADCKEN]) implemented. |
| 8<br>VR1RNGI | Voltage Reference 1 Range Control Bit Implemented<br>This read-only field indicates if a control bit is implemented for selecting the input voltage range on Voltage Reference Option 1.When range control is required, the CFG[VREF1RNG] field is available and documented for controlling the Voltage Reference Option 1.<br>0b - Range control not required. CFG[VREF1RNG] is not implemented.<br>1b - Range control required. CFG[VREF1RNG] is implemented. |
| 7<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 6-4<br>CSW | Channel Scale Width<br>This read-only field indicates if channel scaling is supported by this implementation. When supported, each command buffer has a control field (CMDLa[CSCALE]) for setting input scaling.<br>000b - Channel scaling not supported.<br>001b - Channel scaling supported. 1-bit CSCALE control field.<br>110b - Channel scaling supported. 6-bit CSCALE control field. |
| 3<br>MVI | Multi Vref Implemented<br>This read-only field indicates if multiple Voltage Reference High inputs are supported by this implementation. When multiple voltage references are supported, the CFG[REFSEL] field is available and documented for selecting the Voltage Reference High options.<br>0b - Single voltage reference high (VREFH) input supported.<br>1b - Multiple voltage reference high (VREFH) inputs supported. |
| 2<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 1<br>DIFFEN | Differential Supported |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This read-only field indicates if differential operation is supported by this implementation. When supported, each command buffer has control fields (CMDLa[DIFF] and CMDLa[ABSEL]) for configuring for differential operation and expanding the number of supported channels from 32 to 64.<br>0b - Differential operation not supported.<br>1b - Differential operation supported. CMDLa[DIFF] and CMDLa[ABSEL] control fields implemented. |
| 0<br>RES | Resolution<br>This read-only field indicates the maximum accuracy supported by this implementation. When RES=1, each command buffer has a control field (CMDLa[MODE]) for selecting resolution of conversions for the associated command.<br>0b - Up to 13-bit differential/12-bit single ended resolution supported.<br>1b - Up to 16-bit differential/15-bit single ended resolution supported. |

## 46.4.3 Parameter Register (PARAM)

### 46.4.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 46.4.3.2 Function

The Parameter register indicates the size of several variable integration options for this instance on the device.

### 46.4.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMD_NUM | | | | | | | | CV_NUM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FIFOSIZE | | | | | | | | TRIG_NUM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 46.4.3.4 Fields

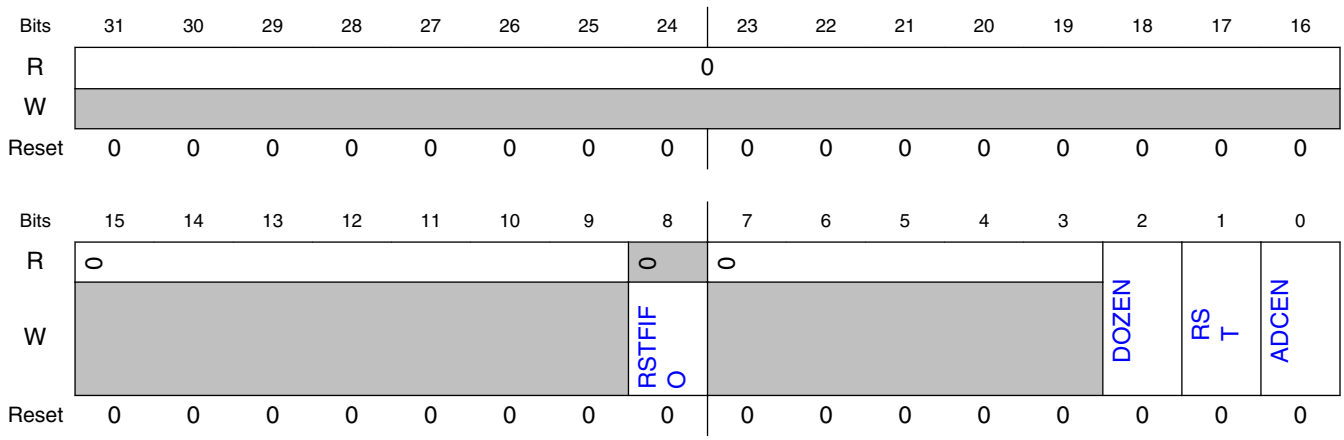
| Field | Function |
|---|---|
| 31-24<br>CMD_NUM | Command Buffer Number<br>Number of command buffers implemented. |
| 23-16<br>CV_NUM | Compare Value Number<br>Number of compare value registers implemented. |
| 15-8<br>FIFOSIZE | Result FIFO Depth<br>The maximum number of conversion datawords that can be stored in the result FIFO before an overflow occurs. This field is read-only.<br>00000001b - Result FIFO depth = 1 dataword.<br>00000100b - Result FIFO depth = 4 datawords.<br>00001000b - Result FIFO depth = 8 datawords.<br>00010000b - Result FIFO depth = 16 datawords.<br>00100000b - Result FIFO depth = 32 datawords.<br>01000000b - Result FIFO depth = 64 datawords. |
| 7-0<br>TRIG_NUM | Trigger Number<br>Number of Triggers implemented. |

### 46.4.4 LPADC Control Register (CTRL)

#### 46.4.4.1 Offset

| Register | Offset |
|---|---|
| CTRL | 10h |

## 46.4.4.2 Diagram




## 46.4.4.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 8<br>RSTFIFO | Reset FIFO<br>0b - No effect.<br>1b - FIFO is reset. |
| 7-3<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 2<br>DOZEN | Doze Enable<br>Controls system transition to Stop and Wait power modes while LPADC is converting. When DOZEN is clear, immediate entries to Wait or Stop are allowed. Note that the selected clock source provided from the on-chip clock source must be able to continue operating. When the DOZEN bit is set, the LPADC waits for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry. When the system is entering a Low Leakage Stop mode, the DOZEN bit is ignored and the module waits for the current transfer to complete any pending operation before acknowledging low leakage mode entry.<br>0b - LPADC is enabled in Doze mode.<br>1b - LPADC is disabled in Doze mode. |
| 1<br>RST | Software Reset<br>Resets all internal logic and registers, except the Control Register. Remains set until cleared by software.<br>0b - LPADC logic is not reset.<br>1b - LPADC logic is reset. |
| 0<br>ADCEN | LPADC Enable<br>0b - LPADC is disabled.<br>1b - LPADC is enabled. |

## 46.4.5 LPADC Status Register (STAT)

### 46.4.5.1 Offset

| Register | Offset |
|---|---|
| STAT | 14h |

### 46.4.5.2 Function

The Status Register provides the current status of the LPADC module.

### 46.4.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CMDACT | | | | 0 | | | | | | TRGACT | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | 0 | | | | | | FOF | RDY |
| W | | | | | | | | | | | | | | | W1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 46.4.5.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 27-24<br>CMDACT | Command Active<br>CMDACT is a read-only status field indicating the command that is actively being processed.<br>0000b - No command is currently in progress.<br>0001b - Command 1 currently being executed.<br>0010b - Command 2 currently being executed.<br>0011-1111b - Associated command number is currently being executed. |
| 23-18<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 17-16 | Trigger Active |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TRGACT | TRGACT is a read-only status field indicating the trigger associated with the command actively being processed. TRGACT only has meaning when CMDACT is non-zero.<br>00b - Command (sequence) associated with Trigger 0 currently being executed.<br>01b - Command (sequence) associated with Trigger 1 currently being executed.<br>10b - Command (sequence) associated with Trigger 2 currently being executed.<br>11b - Command (sequence) associated with Trigger 3 currently being executed. |
| 15-9 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 8 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 7-2 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 1 | Result FIFO Overflow Flag |
| FOF | Indicates that more data has been written to the Result FIFO than it can hold. The newer data is not stored and the FIFO remains holding the original contents. This field asserts regardless of the value of IE[FOFIE]. However, an interrupt request is issued only if IE[FOFIE] is set. This flag is cleared by writing a 1.<br>0b - No result FIFO overflow has occurred since the last time the flag was cleared.<br>1b - At least one result FIFO overflow has occurred since the last time the flag was cleared. |
| 0 | Result FIFO Ready Flag |
| RDY | Indicates when the number of valid datawords in the result FIFO is greater than the watermark level set in the FCTRL[FWMARK] bitfield. This field asserts regardless of the value of FWMIE. However, an interrupt request or DMA request occurs only when the associated control bit (IE[FWMIE] and DE[FWMDE]) is set. This flag is cleared when the FCOUNT (which decrements on each read of the RESFIFO register) is less than or equal to the watermark level set in the FCTRL[FWMARK] bitfield.<br>0b - Result FIFO data level not above watermark level.<br>1b - Result FIFO holding data above watermark level. |

## 46.4.6 Interrupt Enable Register (IE)

### 46.4.6.1 Offset

| Register | Offset |
|---|---|
| IE | 18h |

### 46.4.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | FOFIE | FWMIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 46.4.6.3 Fields

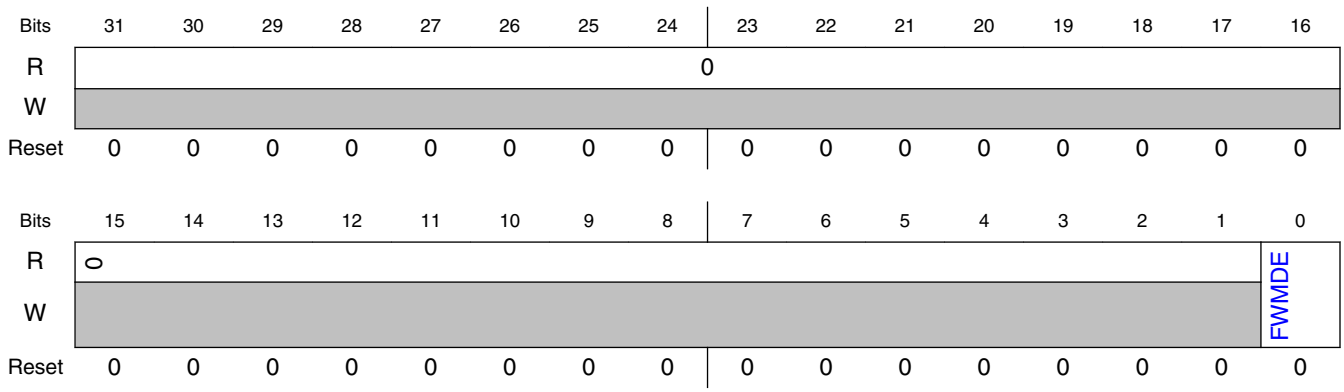
| Field | Function |
|---|---|
| 31-2<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 1<br>FOFIE | Result FIFO Overflow Interrupt Enable<br>Configures ADC to generate overflow interrupt requests when FOF flag is asserted.<br>0b - FIFO overflow interrupts are not enabled.<br>1b - FIFO overflow interrupts are enabled. |
| 0<br>FWMIE | FIFO Watermark Interrupt Enable<br>Configures the module to generate watermark interrupt requests when RDY flag is asserted.<br>0b - FIFO watermark interrupts are not enabled.<br>1b - FIFO watermark interrupts are enabled. |

## 46.4.7 DMA Enable Register (DE)

### 46.4.7.1 Offset

| Register | Offset |
|---|---|
| DE | 1Ch |

### 46.4.7.2 Diagram




### 46.4.7.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 0<br>FWMDE | FIFO Watermark DMA Enable<br>Configures ADC to generate DMA requests when RDY flag is asserted.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |

## 46.4.8 LPADC Configuration Register (CFG)

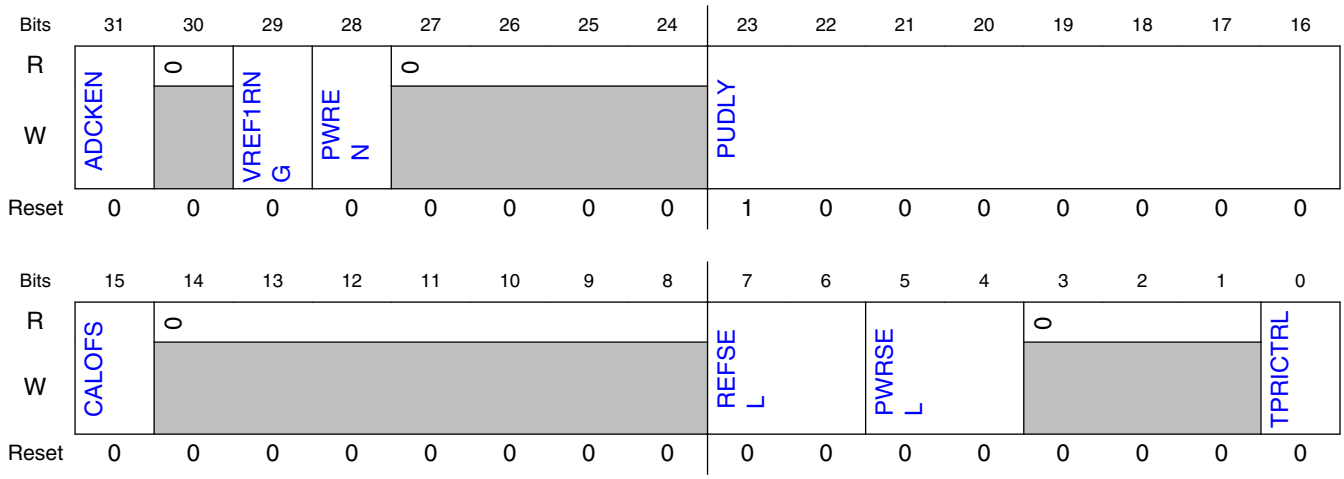
### 46.4.8.1 Offset

| Register | Offset |
|---|---|
| CFG | 20h |

### 46.4.8.2 Function

The Configuration Register controls ADC functions that are common to all commands. The CFG cannot be changed while the CTRL[ADCEN] bit is set. Writes to CFG while CTRL[ADCEN] is set are ignored.

### 46.4.8.3 Diagram




### 46.4.8.4 Fields

| Field | Function |
|---|---|
| 31<br>ADCKEN | LPADC asynchronous clock enable<br>Enables the internally generated clock source. The clock source is used in clock selection logic at the chip level and is optionally used for the LPADC clock source, depending on chip-level clock select definition.<br>0b - LPADC internal clock is disabled<br>1b - LPADC internal clock is enabled |
| 30<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 29<br>VREF1RNG | Enable support for low voltage reference on Option 1 Reference<br>The VREF1RNG control bit should be set when Reference Option 1 is used and the reference is below 1.8V. When the reference is above 1.8V, the VREF1RNG control bit should be clear.<br>0b - Configuration required when Voltage Reference Option 1 input is in high voltage range<br>1b - Configuration required when Voltage Reference Option 1 input is in low voltage range |
| 28<br>PWREN | LPADC Analog Pre-Enable<br>Enables the LPADC analog circuits. When setting CFG[PWREN], user code should delay for a period exceeding the analog startup time of $t_{ADCSTUP}$ before enabling the LPADC for operation. The module is still operational even when CFG[PWREN] is left clear but command execution start is delayed for a period defined by CFG[PUDLY]. Refer to the device datasheet for LPADC idle and LPADC active power consumption parameters.<br>0b - LPADC analog circuits are only enabled while conversions are active. Performance is affected due to analog startup delays.<br>1b - LPADC analog circuits are pre-enabled and ready to execute conversions without startup delays (at the cost of higher DC current consumption). When PWREN is set, the power up delay is enforced such that any detected trigger does not begin ADC operation until the power up delay time has passed. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 27-24<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 23-16<br>PUDLY | Power Up Delay<br>When CFG[PWREN]=0b0, the LPADC analog circuits are only powered while the module is active and there is a counted delay defined by CFG[PUDLY] after an initial trigger transitions the LPADC from its Idle state to allow time for the analog circuits to stabilize. The startup delay count of (PUDLY*4) ADCK cycles must result in a longer delay than the analog startup time of $t_{ADCSTUP}$. Accuracy of the initial conversion after activation is degraded if CFG[PUDLY] is set to too small a value.<br>When CFG[PWREN]=0b1 prior to LPADC activation, the analog circuits are pre-enabled and the activation delay defined by CFG[PUDLY] is bypassed. |
| 15<br>CALOFS | Configure for offset calibration function<br>When CALOFS is set, the LPADC is configured to perform a calibration function anytime the module executes a conversion. Any channel selected is ignored and the value returned in the RESFIFO is a signed value between -31 and 31. -32 is not a valid and is never a returned value. Software should copy the lower 6-bits of the conversion result stored in the RESFIFO after a completed calibration conversion to the OFSTRIM[OFSTRIM] field. The OFSTRIM field is used in normal operation for offset correction.<br>0b - Calibration function disabled<br>1b - Configure for offset calibration function |
| 14-8<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 7-6<br>REFSEL | Voltage Reference Selection<br>Selects the voltage reference high used for conversions.<br>**NOTE:** See the chip configuration information on the voltage reference options specific to this packaged device.<br>00b - (Default) Option 1 setting.<br>01b - Option 2 setting.<br>10b - Option 3 setting.<br>11b - Reserved |
| 5-4<br>PWRSEL | Power Configuration Select<br>Configures the module for power and performance. In the highest power setting, the highest conversion rates are possible. Refer to the device datasheet for power and performance capabilities for each setting. The highest power setting corresponds to 0b11 with decreasing power down to 0b00.<br>00b - Level 1 (Lowest power setting)<br>01b - Level 2<br>10b - Level 3<br>11b - Level 4 (Highest power setting) |
| 3-1<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 0<br>TPRICTRL | LPADC trigger priority control<br>This bitfield controls how higher priority triggers are handled. See Trigger detect and command execution for a detailed explanation trigger event handling.<br>0b - If a higher priority trigger is detected during command processing, the current conversion is aborted and the new command specified by the trigger is started.<br>1b - If a higher priority trigger is received during command processing, the current conversion is completed (including averaging iterations if enabled) and stored to the RESFIFO before the higher priority trigger/command is initiated. Note that compare until true commands can be interrupted prior to resulting in a true conversion. |

## 46.4.9 LPADC Pause Register (PAUSE)

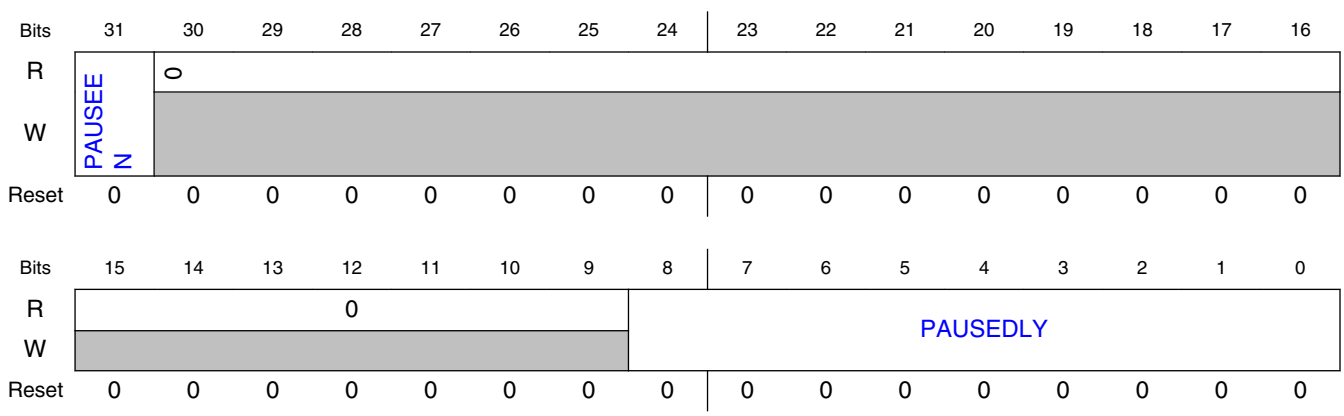
### 46.4.9.1 Offset

| Register | Offset |
|---|---|
| PAUSE | 24h |

### 46.4.9.2 Function

The Pause Register controls an optional inserted delay between conversions. PAUSE cannot be changed while the CTRL[ADCEN] bit is set. Writes to PAUSE while CTRL[ADCEN] is set are ignored.

### 46.4.9.3 Diagram




### 46.4.9.4 Fields

| Field | Function |
|---|---|
| 31<br>PAUSEEN | PAUSE Option Enable<br>Enables the LPADC pausing function. When enabled, a programmable delay is inserted during command execution sequencing between LOOP iterations, between commands in a sequence, and between conversions when command is executing in for "Compare Until True" configuration.<br>0b - Pause operation disabled<br>1b - Pause operation enabled |

*Table continues on the next page...*

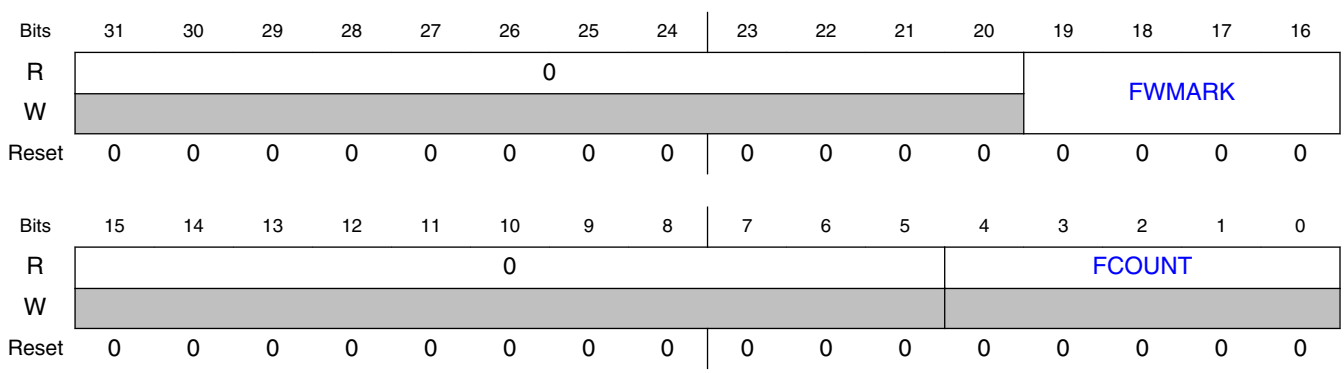
| Field | Function |
|---|---|
| 30-9<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 8-0<br>PAUSEDLY | Pause Delay<br>When PAUSEEN is set, the PAUSEDLY field controls the duration of pausing during command exection sequencing. The pause delay is a count of (PAUSEDLY*4) ADCK cycles. |

## 46.4.10 LPADC FIFO Control Register (FCTRL)

### 46.4.10.1 Offset

| Register | Offset |
|---|---|
| FCTRL | 30h |

### 46.4.10.2 Diagram




### 46.4.10.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 19-16<br>FWMARK | Watermark level selection<br>FWMARK is a programmable threshold setting. When the number of datawords stored in the ADC Result FIFO is greater than the value in this field, the STAT[RDY] flag is asserted to indicate stored data has |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | reached the programmable threshold. When IE[FWMIE] is set, an interrupt request is generated. When DE[FWMDE] is set, a DMA request is generated. |
| 15-5<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 4-0<br>FCOUNT | Result FIFO counter<br>This read-only field indicates the number of datawords that are stored in the result FIFO. This value may be used in conjunction with PARAM[FIFOSIZE] to calculate how much room is left in the result FIFO. FCOUNT is incremented with each store of new data to the result FIFO and decrements with each read of the result FIFO. The FIFO is reset by writing to the CTRL[RSTFIFO] bit, resulting in FCTRL[FCOUNT] initialized to 0x0. |

## 46.4.11 Software Trigger Register (SWTRIG)

### 46.4.11.1 Offset

| Register | Offset |
|---|---|
| SWTRIG | 34h |

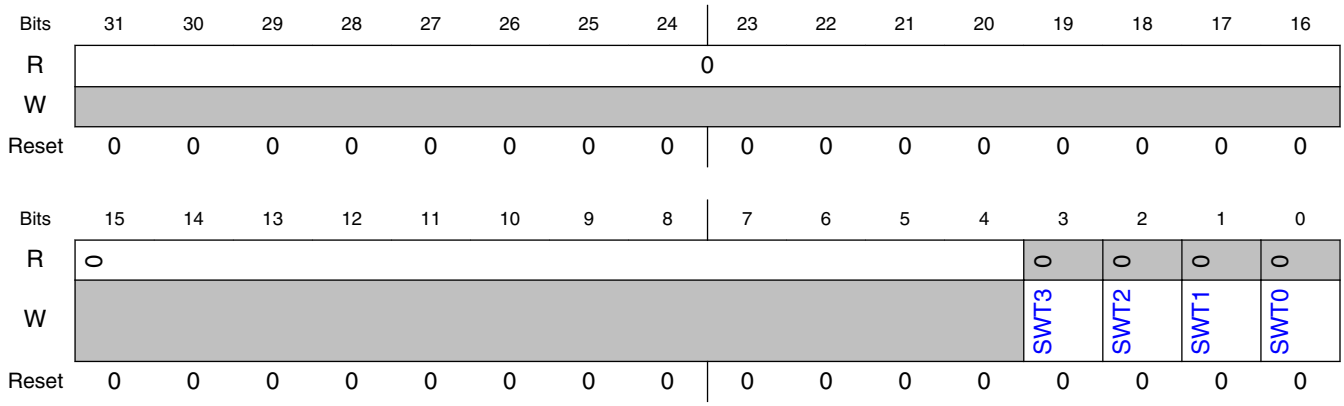
### 46.4.11.2 Function

The Software Trigger Register (SWTRIG) is written to initiate software triggered conversions. Writes to SWTRIG register are ignored while CTRL[ADCEN] is clear.

**NOTE**

There is a synchronization delay of 3 ADCK clock cycles when writing to set CTRL[ADCEN]. Writes to SWTRIG will continue to be ignored ultil CTRL[ADCEN] can propagate into the ADCK domain.

## 46.4.11.3 Diagram




## 46.4.11.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 3<br>SWT3 | Software trigger 3 event<br>Write 1 to SWT3 generates a trigger 3 event. Writing 1 to SWT3 is ignored while the trigger 3 event is being serviced or is pending.<br>0b - No trigger 3 event generated.<br>1b - Trigger 3 event generated. |
| 2<br>SWT2 | Software trigger 2 event<br>Write 1 to SWT2 generates a trigger 2 event. Writing 1 to SWT2 is ignored while the trigger 2 event is being serviced or is pending.<br>0b - No trigger 2 event generated.<br>1b - Trigger 2 event generated. |
| 1<br>SWT1 | Software trigger 1 event<br>Write 1 to SWT1 generates a trigger 1 event. Writing 1 to SWT1 is ignored while the trigger 1 event is being serviced or is pending.<br>0b - No trigger 1 event generated.<br>1b - Trigger 1 event generated. |
| 0<br>SWT0 | Software trigger 0 event<br>Write 1 to SWT0 generates a trigger 0 event. Writing 1 to SWT0 is ignored while the trigger 0 event is being serviced or is pending.<br>0b - No trigger 0 event generated.<br>1b - Trigger 0 event generated. |

## 46.4.12 LPADC Offset Trim Register (OFSTRIM)

### 46.4.12.1 Offset

| Register | Offset |
|---|---|
| OFSTRIM | 40h |

### 46.4.12.2 Function

The Offset Trim register is used to trim for offset. The LPADC supports a calibration step in which the module is configured to perform a calibration operation to determine the value needed in the OFSTRIM register.

To determine the value to put in the OFSTRIM register, set the CFG[CALOFS] and trigger a conversion using standard triggering methods. Any selected channel is ignored and the conversion result stored in the RESFIFO after a completed conversion is a signed value between -31 and 31. To minimize the offset during normal operation, software should read the conversion result from the RESFIFO calibration operation and write the lower 6 bits to the OFSTRIM register.

### 46.4.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | OFSTRIM | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 46.4.12.4 Fields

| Field | Function |
|---|---|
| 31-6 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | This read-only field is reserved and always has the value 0. |
| 5-0<br>OFSTRIM | Trim for offset<br>OFSTRIM is a 6-bit signed value between -31 and 31. -32 is not a valid value. |

## 46.4.13 Trigger Control Register (TCTRL0 - TCTRL3)

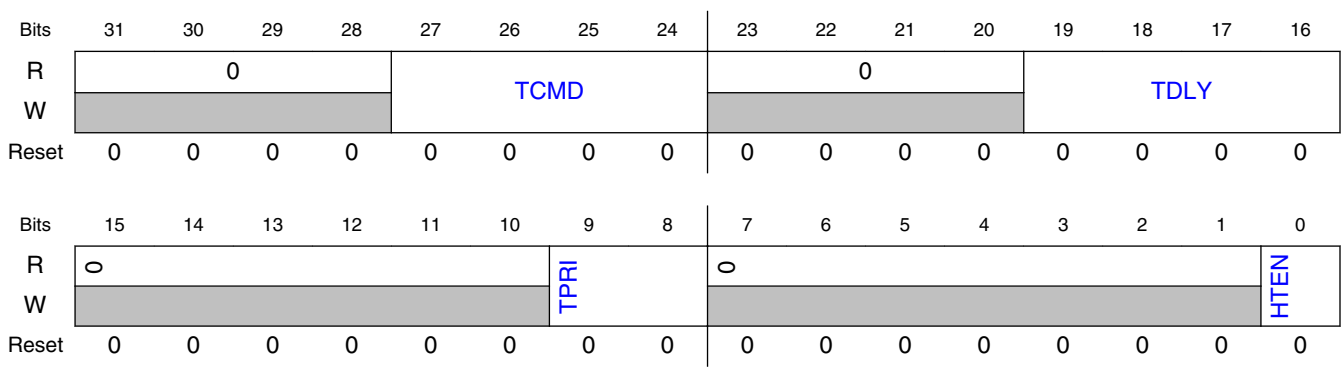
### 46.4.13.1 Offset

| Register | Offset |
|---|---|
| TCTRL0 | C0h |
| TCTRL1 | C4h |
| TCTRL2 | C8h |
| TCTRL3 | CCh |

### 46.4.13.2 Function

The Trigger Control (TCTRL*a*) register implements control fields associated with each implemented trigger source. When the LPADC is actively executing commands, only one of the TCTRL*a* registers is actively controlling ADC conversions. The actively controlling TCTRL*a* register cannot be updated while the LPADC is active. A write to a TCTRL*a* registers while that trigger control register is controlling the ADC operation is ignored.

### 46.4.13.3 Diagram



### 46.4.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 27-24<br>TCMD | Trigger command select<br>Selects the command from command buffer to execute upon detection of the associated trigger event. When a trigger is received while the LPADC is busy converting and the new trigger has higher priority than trigger associated with the current command being executed, the command in this register, associated with the new trigger, is executed under control of CFG[TPRICTRL]. When CFG[TPRICTRL]=0b0, the interruption due to higher priority trigger event is immediate. When CFG[TPRICTRL]=0b1, the current conversion (including any hardware averaging) is allowed to complete and the interruption is at the next loop boundary.<br>0000b - Not a valid selection from the command buffer. Trigger event is ignored.<br>0001b - CMD1 is executed<br>0010-1110b - Corresponding CMD is executed<br>1111b - CMD15 is executed |
| 23-20<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 19-16<br>TDLY | Trigger delay select<br>Selects the trigger delay duration to wait at the start of servicing a trigger event. Each trigger source has an associated programmable delay prior to beginning an initial conversion. When TDLY field is clear, then no delay is incurred. When TDLY is set to a non-zero value, the duration for the delay is $2^{TDLY}$ ADCK cycles. |
| 15-10<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 9-8<br>TPRI | Trigger priority setting<br>This bitfield sets the priority of the associated trigger source. If two or more triggers have the same priority level setting, the lower order trigger event has the higher priority (e. g., if Trigger 0 and Trigger 1 are both pending triggers and TCTRL0[TPRI] is configured the same as TCTRL1[TPRI], then the command associated with Trigger 0 is serviced first).<br>00b - Set to highest priority, Level 1<br>01-10b - Set to corresponding priority level<br>11b - Set to lowest priority, Level 4 |
| 7-1<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 0<br>HTEN | Trigger enable<br>Enables hardware trigger source to initiate conversion on the rising-edge of the input trigger source.<br>**NOTE:** Enabling hardware trigger does not disable software triggers.<br>0b - Hardware trigger source disabled<br>1b - Hardware trigger source enabled |

# 46.4.14 LPADC Command Low Buffer Register (CMDL1 - CMDL 15)

## 46.4.14.1 Offset

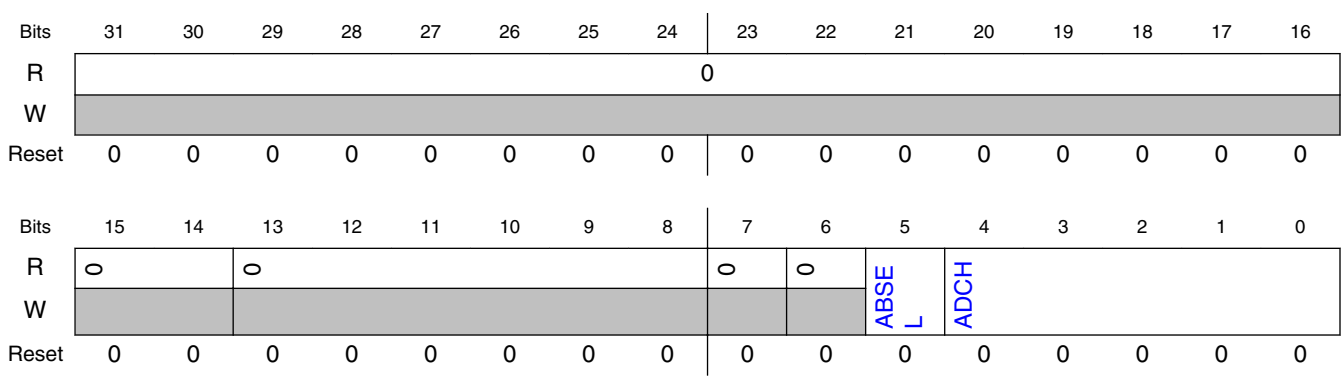
For a = 1 to 15:

| Register | Offset |
|---|---|
| CMDLa | F8h + (a × 8h) |

## 46.4.14.2 Function

There are 15 command buffers (CMD*a*), each constructed from two 32-bit registers (CMDL*a*:CMDH*a*) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the TCTRL*a*[TCMD] bitfield. When the LPADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer cannot be updated while the LPADC is active. A write to a CMD buffer while that CMD buffer is controlling the ADC operation is ignored.

## 46.4.14.3 Diagram



### 46.4.14.4 Fields

| Field | Function |
|---|---|
| 31-14 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 13-8 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 7 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 6 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 5 | A-side vs. B-side Select |
| ABSEL | ABSEL is used in conjunction with ADCH field to select one of 32 A-channel or one of 32 B-channel inputs.<br>0b - The associated A-side channel is converted.<br>1b - The associated B-side channel is converted. |
| 4-0 | Input channel select |
| ADCH | Each ADCH channel selection has an associated A side and an associated B side input. The ADCH setting in conjunction with the ABSEL bitfield setting selects the input from one of the channel inputs. See the ABSEL bitfield description.<br>**NOTE:** Some of the input channels are from internal resources such as temperature sensors and bandgap voltage sources and may only be connected to individual instances of the ADC module. Some of the input channel options in the bitfield-setting descriptions might not be available for your device. For the ADC channel assignments for your device, see the chip-specific information.<br>00000b - Select CH0A or CH0B<br>00001b - Select CH1A or CH1B<br>00010b - Select CH2A or CH2B<br>00011b - Select CH3A or CH3B<br>00100-11101b - Select corresponding channel CHnA or CHnB<br>11110b - Select CH30A or CH30B<br>11111b - Select CH31A or CH31B |

## 46.4.15 LPADC Command High Buffer Register (CMDH1 - CMDH 4)

### 46.4.15.1 Offset

| Register | Offset |
|---|---|
| CMDH1 | 104h |
| CMDH2 | 10Ch |

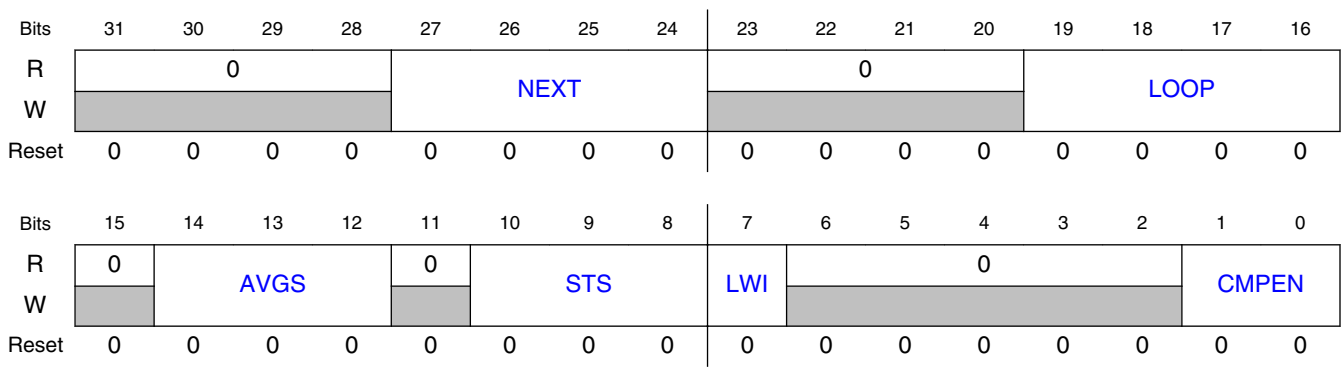
*Table continues on the next page...*

| Register | Offset |
|---|---|
| CMDH3 | 114h |
| CMDH4 | 11Ch |

## 46.4.15.2 Function

There are 15 command buffers (CMD*a*), each constructed from two 32-bit registers (CMDL*a*:CMDH*a*) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the TCTRL*a*[TCMD] bitfield. When the LPADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer cannot be updated while the LPADC is active. A write to a CMD buffer while that CMD buffer is controlling ADC operation is ignored.

## 46.4.15.3 Diagram




## 46.4.15.4 Fields

| Field | Function |
|---|---|
| 31-28 | Reserved |
| — | This read-only field is reserved and always has the value 0. |
| 27-24 | Next Command Select |
| NEXT | Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order and the order of execution is strictly controlled by the NEXT field (for example, a sequence of commands could be CMD2, CMD1, CMD3). Unending |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Lower priority trigger events cannot be serviced until a higher priority triggered command (or sequence of commands) completes.<br>0000b - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger.<br>0001b - Select CMD1 command buffer register as next command.<br>0010-1110b - Select corresponding CMD command buffer register as next command<br>1111b - Select CMD15 command buffer register as next command. |
| 23-20<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 19-16<br>LOOP | Loop Count Select<br>Selects how many times this command executes (and stores conversion result to RESFIFO) before finish and transition to the next command or Idle state. LWI field controls whether a single channel is converted on each iteration or auto channel increment results in channel scanning functionality.<br>0000b - Looping not enabled. Command executes 1 time.<br>0001b - Loop 1 time. Command executes 2 times.<br>0010b - Loop 2 times. Command executes 3 times.<br>0011-1110b - Loop corresponding number of times. Command executes LOOP+1 times.<br>1111b - Loop 15 times. Command executes 16 times. |
| 15<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 14-12<br>AVGS | Hardware Average Select<br>Selects how many ADC conversions are averaged to create the ADC result ($2^{AVGS}$). An internal storage buffer is used to capture temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for more detailed description on usage of AVGS, LOOP, and NEXT bitfields in command execution sequencing.<br>000b - Single conversion.<br>001b - 2 conversions averaged.<br>010b - 4 conversions averaged.<br>011b - 8 conversions averaged.<br>100b - 16 conversions averaged.<br>101b - 32 conversions averaged.<br>110b - 64 conversions averaged.<br>111b - 128 conversions averaged. |
| 11<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 10-8<br>STS | Sample Time Select<br>When programmed to 000 the minimum sample time of 3 ADCK cycles is selected. When STS is programmed to a non-zero value the sample time is ($3 + 2^{STS}$) ADCK cycles. The shortest sample time maximizes conversion speed for lower impedance inputs. Extending sample time allows higher impedance inputs to be accurately sampled. Longer sample times can also be used to lower overall power consumption when command looping and sequencing is configured and high conversion rates are not required.<br>000b - Minimum sample time of 3 ADCK cycles.<br>001b - $3 + 2^1$ ADCK cycles; 5 ADCK cycles total sample time.<br>010b - $3 + 2^2$ ADCK cycles; 7 ADCK cycles total sample time.<br>011b - $3 + 2^3$ ADCK cycles; 11 ADCK cycles total sample time.<br>100b - $3 + 2^4$ ADCK cycles; 19 ADCK cycles total sample time. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 101b - 3 + $2^5$ ADCK cycles; 35 ADCK cycles total sample time.<br>110b - 3 + $2^6$ ADCK cycles; 67 ADCK cycles total sample time.<br>111b - 3 + $2^7$ ADCK cycles; 131 ADCK cycles total sample time. |
| 7<br>LWI | Loop with Increment<br>When LWI is clear, the LOOP field selects the number of times the selected channel is converted consecutively. When LWI is set, auto channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution.<br>Example 1: LOOP = 0x8, LWI = 0b0, ABSEL = 0b0, ADCH = 0xD. Convert on channel 13A 8 times.<br>Example 2: LOOP = 0x8, LWI= 0b1, ABSEL = 0b0, ADCH = 0xD. Run channels 13A, 14A, 15A, ... 20A each one time.<br>Maximum channel scanning using a single command buffer then is defined by the maximum value of the LOOP field (16).<br>Note, when LWI is set, the minimum sample length must be extended to CMDHa[STS] = 0x3.<br>0b - Auto channel increment disabled<br>1b - Auto channel increment enabled |
| 6-2<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 1-0<br>CMPEN | Compare Function Enable<br>After an ADC channel input is sampled and converted and any averaging iterations are performed, the CMDH*a*[CMPEN] field guides operation of the automatic compare function to optionally only store when the compare operation is true. When compare is enabled, the conversion result is compared to the compare value registers (CV*a*[CVH] and CV*a*[CVL]). There are multiple options on command sequencing related to the compare function. See Compare function for a detailed explanation of the compare options.<br>**NOTE:** Not all Command Buffers have implemented the CMPEN field. The CMPEN field is only available in Command Buffers that have a corresponding Compare Value register.<br>00b - Compare disabled.<br>01b - Reserved<br>10b - Compare enabled. Store on true.<br>11b - Compare enabled. Repeat channel acquisition (sample/convert/compare) until true. |

## 46.4.16 LPADC Command High Buffer Register (CMDH5 - CMDH 15)

### 46.4.16.1 Offset

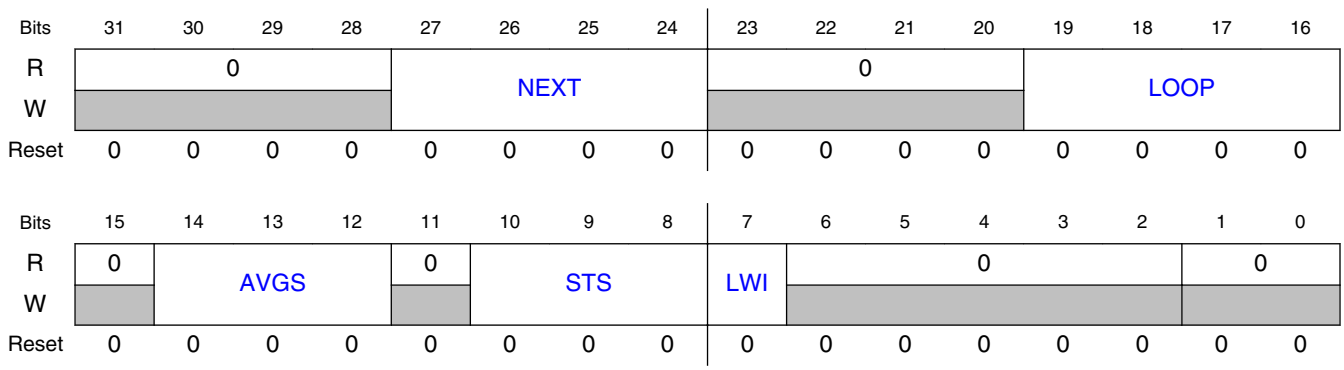
For a = 5 to 15:

| Register | Offset |
|---|---|
| CMDHa | FCh + (a × 8h) |

### 46.4.16.2 Function

There are 15 command buffers (CMD*a*), each constructed from two 32-bit registers (CMDL*a*:CMDH*a*) that can be configured for different channel select and varying conversion options. Any of the command buffers is selected and used as the controlling command by association to a trigger event via configuration of the TCTRL*a*[TCMD] bitfield. When the LPADC is actively executing commands, only one of the CMD buffers is actively controlling ADC conversions. The actively controlling CMD buffer cannot be updated while the LPADC is active. A write to a CMD buffer while that CMD buffer is controlling ADC operation is ignored.

### 46.4.16.3 Diagram




### 46.4.16.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 27-24<br>NEXT | Next Command Select<br>Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order and the order of execution is strictly controlled by the NEXT field (for example, a sequence of commands could be CMD2, CMD1, CMD3). Unending circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Lower priority trigger events cannot be serviced until a higher priority triggered command (or sequence of commands) completes.<br>0000b - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger.<br>0001b - Select CMD1 command buffer register as next command.<br>0010-1110b - Select corresponding CMD command buffer register as next command |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1111b - Select CMD15 command buffer register as next command. |
| 23-20<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 19-16<br>LOOP | Loop Count Select<br>Selects how many times this command executes (and stores conversion result to RESFIFO) before finish and transition to the next command or Idle state. LWI field controls whether a single channel is converted on each iteration or auto channel increment results in channel scanning functionality.<br>0000b - Looping not enabled. Command executes 1 time.<br>0001b - Loop 1 time. Command executes 2 times.<br>0010b - Loop 2 times. Command executes 3 times.<br>0011-1110b - Loop corresponding number of times. Command executes LOOP+1 times.<br>1111b - Loop 15 times. Command executes 16 times. |
| 15<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 14-12<br>AVGS | Hardware Average Select<br>Selects how many ADC conversions are averaged to create the ADC result ($2^{AVGS}$). An internal storage buffer is used to capture temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for more detailed description on usage of AVGS, LOOP, and NEXT bitfields in command execution sequencing.<br>000b - Single conversion.<br>001b - 2 conversions averaged.<br>010b - 4 conversions averaged.<br>011b - 8 conversions averaged.<br>100b - 16 conversions averaged.<br>101b - 32 conversions averaged.<br>110b - 64 conversions averaged.<br>111b - 128 conversions averaged. |
| 11<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 10-8<br>STS | Sample Time Select<br>When programmed to 000 the minimum sample time of 3 ADCK cycles is selected. When STS is programmed to a non-zero value the sample time is ($3 + 2^{STS}$) ADCK cycles. The shortest sample time maximizes conversion speed for lower impedance inputs. Extending sample time allows higher impedance inputs to be accurately sampled. Longer sample times can also be used to lower overall power consumption when command looping and sequencing is configured and high conversion rates are not required.<br>000b - Minimum sample time of 3 ADCK cycles.<br>001b - $3 + 2^1$ ADCK cycles; 5 ADCK cycles total sample time.<br>010b - $3 + 2^2$ ADCK cycles; 7 ADCK cycles total sample time.<br>011b - $3 + 2^3$ ADCK cycles; 11 ADCK cycles total sample time.<br>100b - $3 + 2^4$ ADCK cycles; 19 ADCK cycles total sample time.<br>101b - $3 + 2^5$ ADCK cycles; 35 ADCK cycles total sample time.<br>110b - $3 + 2^6$ ADCK cycles; 67 ADCK cycles total sample time.<br>111b - $3 + 2^7$ ADCK cycles; 131 ADCK cycles total sample time. |
| 7<br>LWI | Loop with Increment<br>When LWI is clear, the LOOP field selects the number of times the selected channel is converted consecutively. When LWI is set, auto channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution.<br>Example 1: LOOP = 0x8, LWI = 0b0, ABSEL = 0b0, ADCH = 0xD. Convert on channel 13A 8 times. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Example 2: LOOP = 0x8, LWI= 0b1, ABSEL = 0b0, ADCH = 0xD. Run channels 13A, 14A, 15A, ... 20A each one time.<br>Maximum channel scanning using a single command buffer then is defined by the maximum value of the LOOP field (16).<br>Note, when LWI is set, the minimum sample length must be extended to CMDHa[STS] = 0x3.<br>0b - Auto channel increment disabled<br>1b - Auto channel increment enabled |
| 6-2<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 1-0<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |

## 46.4.17 Compare Value Register (CV1 - CV4)

### 46.4.17.1 Offset

| Register | Offset |
|---|---|
| CV1 | 200h |
| CV2 | 204h |
| CV3 | 208h |
| CV4 | 20Ch |

### 46.4.17.2 Function

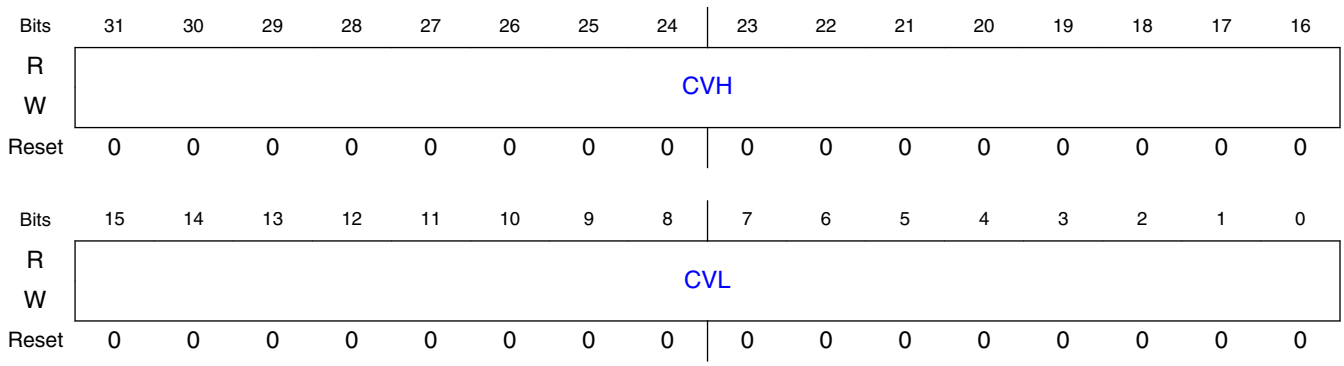
The compare value registers (CV*a*) contain values used to compare the conversion result when the compare function is enabled. This register is formatted in the same way as the D field in the RESFIFO registers in different modes of operation for both bit position definition and value format using unsigned or signed 2's complement. There is a direct association of each compare value register to a specific command buffer register (i.e., CV1 is only used during execution of CMD1 command).

**NOTE**

Not all Command Buffers have an associated Compare Value register. The compare function is only available on Command Buffers that have a corresponding Compare Value register.

When the LPADC is actively executing commands, the CV*a* register associated with the active command (CMD*a*) cannot be updated and writes to associated CV*a* register during this time are ignored.

### 46.4.17.3 Diagram




### 46.4.17.4 Fields

| Field | Function |
|---|---|
| 31-16<br>CVH | Compare Value High.<br>The compare function can be configured to check whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. After the input is sampled and converted and any averaging iterations are performed, the compare values in CVL and CVH are optionally used in a compare operation on the result. See Compare function for a description on how CVH is used. |
| 15-0<br>CVL | Compare Value Low.<br>The compare function can be configured to check whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. After the input is sampled and converted and any averaging iterations are performed, the compare values in CVL and CVH are optionally used in a compare operation on the result. See Compare function for a description on how CVL is used. |

## 46.4.18 LPADC Data Result FIFO Register (RESFIFO)

### 46.4.18.1 Offset

| Register | Offset |
|---|---|
| RESFIFO | 300h |

### 46.4.18.2 Function

The data result FIFO register (RESFIFO) is a 16 entry FIFO that stores the data result of ADC conversions. In addition, several tag fields of source command and trigger information are stored along with the data. FCTRL[FCOUNT] indicates how many valid datawords are stored in the RESFIFO. Reading RESFIFO provides the oldest unread dataword entry in the FIFO and decrements FCOUNT. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

The following table describes the format of data in the result FIFO.

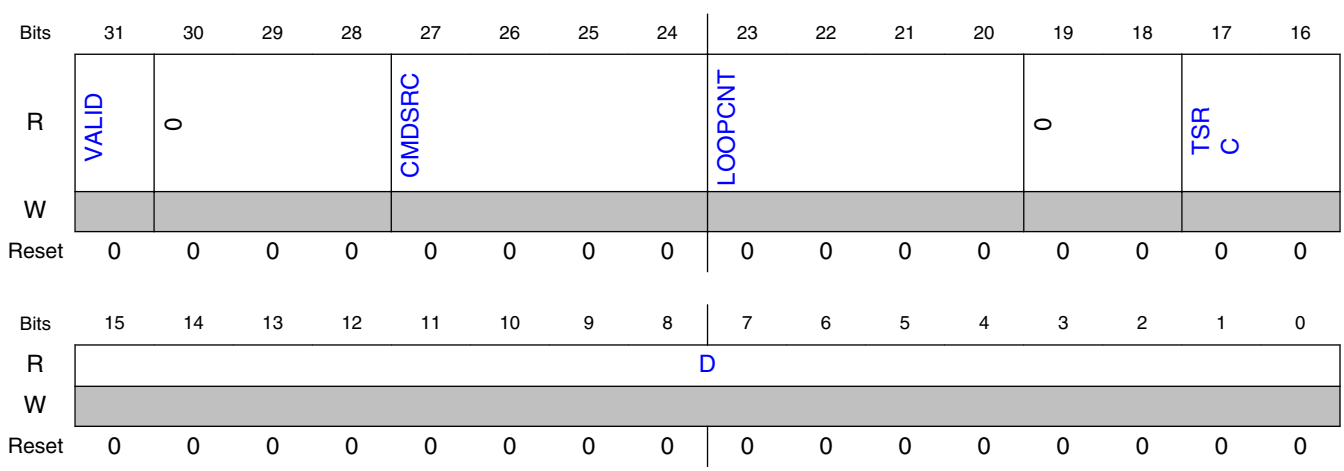
**Table 46-5. Data result register format description**

| Conversion mode | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Format |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12-bit single-ended | 0 | D | D | D | D | D | D | D | D | D | D | D | D | 0 | 0 | 0 | Unsigned, zero in D[15] and D[2:0] |

**NOTE**

D: Data

### 46.4.18.3 Diagram



### 46.4.18.4 Fields

| Field | Function |
|---|---|
| 31<br>VALID | FIFO entry is valid<br>Indicate the FIFO entry is valid. When the FIFO is holding data the VALID bit is set. When the FIFO is empty the VALID bit is clear.<br>0b - FIFO is empty. Discard any read from RESFIFO.<br>1b - FIFO record read from RESFIFO is valid. |
| 30-28<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 27-24<br>CMDSRC | Command Buffer Source<br>Indicates the command buffer being executed that generated this result.<br>0000b - Not a valid value CMDSRC value for a dataword in RESFIFO. 0x0 is only found in initial FIFO state prior to an ADC conversion result dataword being stored to a RESFIFO buffer.<br>0001b - CMD1 buffer used as control settings for this conversion.<br>0010-1110b - Corresponding command buffer used as control settings for this conversion.<br>1111b - CMD15 buffer used as control settings for this conversion. |
| 23-20<br>LOOPCNT | Loop count value<br>Indicates the loop count value during command execution that generated this result. When CMDH*a*[LOOP] is non-zero, the command execution stores off a result multiple times during command execution (at the loop boundary).<br>0000b - Result is from initial conversion in command.<br>0001b - Result is from second conversion in command.<br>0010-1110b - Result is from LOOPCNT+1 conversion in command.<br>1111b - Result is from 16$^{th}$ conversion in command. |
| 19-18<br>— | Reserved<br>This read-only field is reserved and always has the value 0. |
| 17-16<br>TSRC | Trigger Source<br>Indicate the trigger source that initiated a conversion and generated this result. When multiple commands are chained together using the NEXT field, the TSRC field for all datawords stored to the RESFIFO indicate the trigger source that started the sequence of commands.<br>00b - Trigger source 0 initiated this conversion.<br>01b - Trigger source 1 initiated this conversion.<br>10b - Trigger source 2 initiated this conversion.<br>11b - Trigger source 3 initiated this conversion. |
| 15-0<br>D | Data result<br>The formatting for the data in D is summarized in Table 46-5. |

## 46.5 Functional description

The LPADC module performs analog-to-digital conversions on any of the software selectable analog input channels by a successive approximation algorithm. The module initializes to its lowest power state during reset and in Low-Power Stop mode. The LPADC analog circuits can optionally be pre-enabled for faster starts to conversions at the expense of higher idle currents. Conversions are initiated by selectable trigger events

from software or hardware sources. The trigger detect logic includes a configurable enable and priority scheme for the available trigger sources. The module includes multiple command buffers that provide configurable flexibility for channel scanning and independent channel selections for different trigger sources. Multiple command buffers also allow variable option selection such as sample time and averaging on a per-channel basis.

This module optionally averages the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting CMDH*a*[AVGS] bitfield to a non-zero value and operates in any of the conversion modes and configurations.

When the conversion and averaging loops are completed, the resulting data is placed in a FIFO data buffer along with other tag information associated with the result. A configurable watermark level supports interrupt or DMA requests when the number of stored datawords exceeds the setting.

The LPADC module optionally compares the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range or outside-range detection. The compare function operates in any of the conversion modes and configurations.

The sequencing of an ADC command is summarized in the flow diagram shown below.

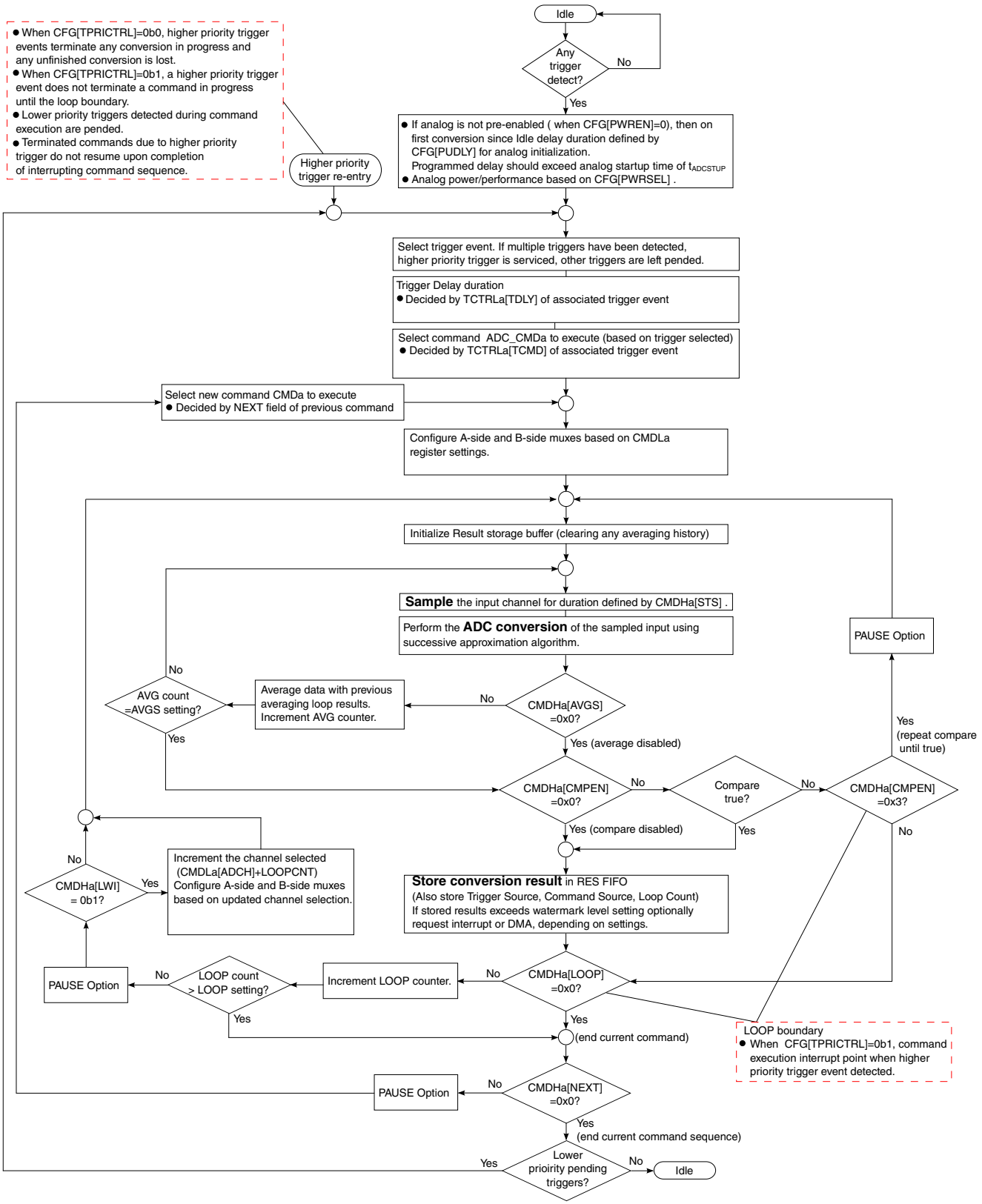



**Figure 46-2. ADC command execution flow diagram**

### 46.5.1 Voltage reference

The voltage reference high ($V_{REFH}$) used by the LPADC is supplied from either an on-chip voltage reference source or from an off-chip source supplied through external pins. $V_{REFL}$ is always from an external pin and must be at the same voltage potential as $V_{SSA}$. See the chip configuration information on the voltage reference options specific to this packaged device.

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via the CFG[REFSEL] field). See the chip configuration information on the voltage reference options specific to this packaged device.

### 46.5.2 Power control

The default setting for the ADC analog circuits is disabled while the LPADC is in its Idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled and require a period of initialization before the first conversion cycle. The CFG[PUDLY] should be programmed such that a delay duration longer than $t_{ADCSTUP}$ is incurred. Accuracy of the initial conversion(s) after activation is degraded if CFG[PUDLY] is set to too small a value.

Faster conversion startup times can be achieved by optionally setting the CFG[PWREN] field to pre-enable the analog circuits of the LPADC at the expense of power consumption even while the module is in an idle state. When CFG[PWREN] is set, the Power Enable timer is activated and enforces the minimum startup (controlled by the PUDLY register field) before detected triggers are allowed to initiate ADC conversions.

This module also has configuration options for controlling power and performance summarized in the table below. See the device datasheet for specification of the available power modes.

**Table 46-6. Power option settings**

| CFG[PWRSEL] | Description |
|---|---|
| 0b00 (Default) | Level 1. Slowest speed/Lowest power setting. |
| 0b01 | Level 2. |
| 0b10 | Level 3. |
| 0b11 | Level 4. Fastest speed/highest power setting. |

In the two highest power mode settings (CFG[PWRSEL]=0x2 or CFG[PWRSEL]=0x3) the LPADC will delay start of sampling for 16 ADC clock cycles on the initial conversion since being in its Idle state. This delay is in addition to any configured Power Up delay or trigger delay.

### 46.5.3 Clock operation

The LPADC operates from the ADCK clock input provided from an on-chip clock select block and is used by the SAR conversion control sequencing logic and the FIFO storage buffer. The ADCK frequency must fall within the specified frequency range for ADCK and will vary based on configuration of CFG[PWRSEL]. See the device datasheet for supported frequency ranges.

The LPADC continues operating in stop and wait modes provided the Doze Enable bit (CTRL[DOZEN]) is clear and the on-chip clock select block continues to supply an ADCK clock source. When the CTRL[DOZEN] bit is set, the module will wait for the current averaging iteration/FIFO storage to complete before acknowledging stop or wait mode entry.

### 46.5.4 Trigger detect and command execution

See Figure 46-2 for a flow diagram of command execution sequencing.

LPADC command execution is initiated from up to 4 trigger sources. Each trigger can be software generated by writing 0b1 to the corresponding SWTRIG[SWT*n*] bitfield. Alternatively, hardware triggers can be generated from asynchronous input sources at the periphery of the module. The number and sources of hardware triggers implemented is device-specific. See the chip configuration information for description of available hardware trigger sources for this device.

Each hardware trigger source is enabled by setting the associated enable bit (TCTRL*a*[HTEN]). Each trigger source is assigned a priority via the associated priority control field (TCTRL*a*[TPRI]). Each of the trigger sources is configurably associated with a command in the command buffer via the associated command select field (TCTRL*a*[TCMD]).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising-edge of the associated hardware trigger source.

If a trigger event occurs that is configured for a higher priority than the trigger source associated with the currently executing command, the current command is handled in one of two ways depending on the CFG[TPRICTRL] bitfield setting. When

CFG[TPRICTRL]=0b0, a higher priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started. When CFG[TPRICTRL]=0b1, a higher priority triggers allows the current conversion to complete (including averaging and compare functions) before the higher priority command is initiated.

If a lower priority trigger occurs (i.e., a trigger event occurs that is configured for a lower priority than the trigger source associated with the currently executing command), the trigger detect is pended until completion of the current command sequence.

At the completion of a command, the LPADC does not resume a previously interrupted command (or series of commands). The module begins command execution associated with the processing of lower priority pending triggers. If no trigger events are pending, the LPADC transitions to its idle state.

When a conversion is completed (including hardware averaging when AVGS is non-zero), the result is placed in the RESFIFO buffer. When an ADC command selects looping (when LOOP is non-zero) a command will store multiple conversion results to the FIFO during execution of that command.

At the end of command execution, the NEXT field of the command selects the next command to be executed. Multiple commands can be executed sequentially by configuration of each command's NEXT field. Setting the next command to 0x0 causes conversions to terminate at the completion of the command. Unending circular command execution is allowed by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Lower priority trigger events cannot be serviced until a higher priority triggered command (or command sequence) completes.

Disabling the LPADC by writing 0b0 to the CTRL[ADCEN] bitfield terminates any active ADC command processing. Writing 0b0 to the CTRL[ADCEN] bitfield causes the current command (or command sequence) to terminate, clears any pending triggers, and sends the module to an Idle state.

### 46.5.5 Pause option

When the maximum conversion rate is not required by an application the effective conversion rate can be reduced by implementing periodic trigger events to initiate ADC conversions or by selecting a reduced frequency clock as the ADACK source. Both of these options are chip specific and are dependent on ADC triggering and clocking options external to the LPADC module. The latency associated with ADC analog power up delays results in a limit on the maximum conversion rate when using periodic triggering.

Another means of reducing conversion rates is by inserting a pause of a programmable duration between LOOP iterations, between commands in a sequence, and between conversions when command is executing in the "Compare Until True" configuration. When PAUSE[PAUSEEN] is set, the PAUSE[PAUSEDLY] field controls the duration of pausing during command exection sequencing. The pause delay is a count of (PAUSEDLY*4) ADCK cycles. The PAUSE register cannot be changed while the CTRL[ADCEN] bit is set. Writes to the PAUSE register while CTRL[ADCEN] is set are ignored.

See Figure 46-2 for the places during command execution sequencing where the pause is optionally inserted.

### 46.5.6 Result FIFO operation

The LPADC includes a 16 entry FIFO in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command and the loop count are also stored along with the data. FCTRL[FCOUNT] indicates how many valid datawords are stored in the RESFIFO.

A programmable watermark threshold supports configurable notification of data availability. When FCOUNT is greater than FCTRL[FWMARK], the STAT[RDY] flag is asserted. When IE[FWMIE] is set, a watermark interrupt request is issued. When DE[FWMDE] is set, a DMA request is issued. Reading RESFIFO provides the oldest unread dataword entry in the FIFO and decrements FCOUNT. When FCOUNT falls equal to or below FWMARK, the RDY flag is cleared.

The FIFO can be emptied by successive reads of RESFIFO. When the RESFIFO[VALID] bit is 1, the associated FIFO entry is valid. Reading RESFIFO when the FIFO is empty (when RESFIFO[VALID] is clear and FCOUNT=0x0) provides an undefined dataword. The FIFO is reset by writing 0b1 to the CTRL[RSTFIFO] bit.

If the LPADC attempts to store a dataword to the FIFO when the FIFO is full, the FIFO overflow flag (FCTRL[FOF]) is set. When IE[FOFIE] is set, an overflow interrupt request is issued. The STAT[FOF] flag is cleared by writing 1 to FOF. On overflow events, no new data is stored and the data associated with the store that triggered the overflow is lost, the current ADC command (sequence) is aborted and all pending trigger events are discarded. No new triggers are detected until the overflow flag is cleared.

## 46.5.7 Compare function

After the input is sampled and converted and any averaging iterations are performed, the CMDH*a*[CMPEN] field guides operation of the automatic compare function to optionally only store when the compare operation is true. There are multiple options on command sequencing related to the compare function as summarized in the table below.

**NOTE**

Not all Command Buffers have an associated Compare Value register. The compare function is only available on Command Buffers that have a corresponding Compare Value register.

**Table 46-7. Compare modes**

| CMDH*a*[CMPEN] | Compare Function | Description |
|---|---|---|
| 0b00 | Compare disabled | Do not perform compare operation. Always store the conversion result to the FIFO. |
| 0b01 | Reserved | |
| 0b10 | Store on true | Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. If compare is false, do not store the result to the FIFO. In either the true or false condition, the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required. |
| 0b11 | Repeat compare until true | Perform compare operation. Store conversion result to FIFO at end of averaging only if compare is true. Once the true condition is found the LOOP setting is considered and increments the LOOP counter before deciding whether the current command has completed or additional LOOP iterations are required. If the compare is false do not store the result to the FIFO. The conversion is repeated without consideration of LOOP setting and does not increment the LOOP counter. |

Depending on CV*a*[CVH] and CV*a*[CVL] values programmed, the compare operation checks whether the result is less than, greater than, or if the result falls within or outside a range determined by two compare values. The compare values are used as described in the following table.

**Table 46-8. Compare operations**

| CV*a*[CVL] vs. CV*a*[CVH] | Operation | Description |
|---|---|---|
| set CVL < CVH | Outside range (General form) | Compare true if the result is less than CVL value or greater than CVH value. |
| set CVH to max value<br>set CVL to compare point | Less than | Compare true if the result is less than CVL value. |
| set CVL to min value<br>set CVH to compare point | Greater than | Compare true if the result is greater than CVH value. |

*Table continues on the next page...*

**Table 46-8. Compare operations (continued)**

| CV*a*[CVL] vs. CV*a*[CVH] | Operation | Description |
|---|---|---|
| set CVL > CVH | Inside range | Compare true if the result is less than CVL value and greater than CVH value. |

**Note**

In low power modes where the LPADC continues to operate, the compare function can monitor the voltage and only wake the device when the compare condition is met.

# Chapter 47
# Low Power Comparator (LPCMP)

## 47.1 Chip-specific LPCMP information

### 47.1.1 LPCMP input connections

The following table shows the fixed internal connections to the LPCMP.

**Table 47-1. LPCMP input connections**

| LPCMP Inputs | LPCMP0 | LPCMP1 |
|---|---|---|
| IN0 | LPCMP0_IN0 | LPCMP1_IN0 |
| IN1 | LPCMP0_IN1 | LPCMP1_IN1 |
| IN2 | LPCMP0_IN2 | LPCMP1_IN2 |
| IN3 | LPCMP0_IN3 / DAC0_OUT (12-bit) | LPCMP1_IN3 / DAC0_OUT (12-bit) |
| IN4 | — | LPCMP1_IN4 |
| IN5 | LPCMP0_IN5 / VREF Output | LPCMP1_IN5 / VREF Output |
| IN6 | PMC 1 V Bandgap | PMC 1 V Bandgap |
| IN7 | DAC0 Reference (6-bit) | DAC1 Reference (6-bit) |

### 47.1.2 LPCMP external references

The 6-bit DAC sub-block supports the selection of two reference sources. For this device, the references are:

- $V_{REFH_EXT}$ - $V_{DDIO1}$ (LPCMP0), $V_{DDIO2}$ (LPCMP1). LPCMPx[VRSEL] = 1
- $V_{REFH_INT}$ -$V_{SS_BULK}$, LPCMPx[VRSEL] = 0

## 47.2 Introduction

The low power comparator (LPCMP) module provides a circuit for comparing two analog input voltages. It comprises a comparator (CMP), a DAC and an analog mux (ANMUX).

The CMP circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The DAC is a 64-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. The 64-tap resistor ladder network divides the supply reference $V_{in}$ into 64 voltage levels. A 6-bit digital signal input selects the output voltage level, which varies from $V_{in}$ to $V_{in}/64$. $V_{in}$ can be selected from two voltage sources, vrefh_int and vrefh_ext. The DAC from a comparator is available as an on-chip internal signal only and is not available to output externally to a pin.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One channel is provided by the DAC. Refer to the chip-specific information section for details on which device resources are connected to other channels. The mux circuit is designed to operate across the full range of the supply voltage.

## 47.3 Features

The features of the LPCMP module include:

- Two 8-to-1 channel MUXes to select input signal from eight channels
- Multiple operation modes to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally Filtered
- Selectable performance levels: nano mode, normal mode, high speed mode
- Programmable hysteresis control
- Selectable inversion on comparator output
- External hysteresis can be used at the same time that the output filter is used for internal functions

- Interrupt and DMA support
- Includes a 6-bit resolution DAC
- Selectable supply reference source for DAC
- Configurable low power mode or high speed mode for DAC

## 47.4 LPCMP Block Diagram

The following figure shows the LPCMP block diagram includes the CMP, DAC, and ANMUX modules.

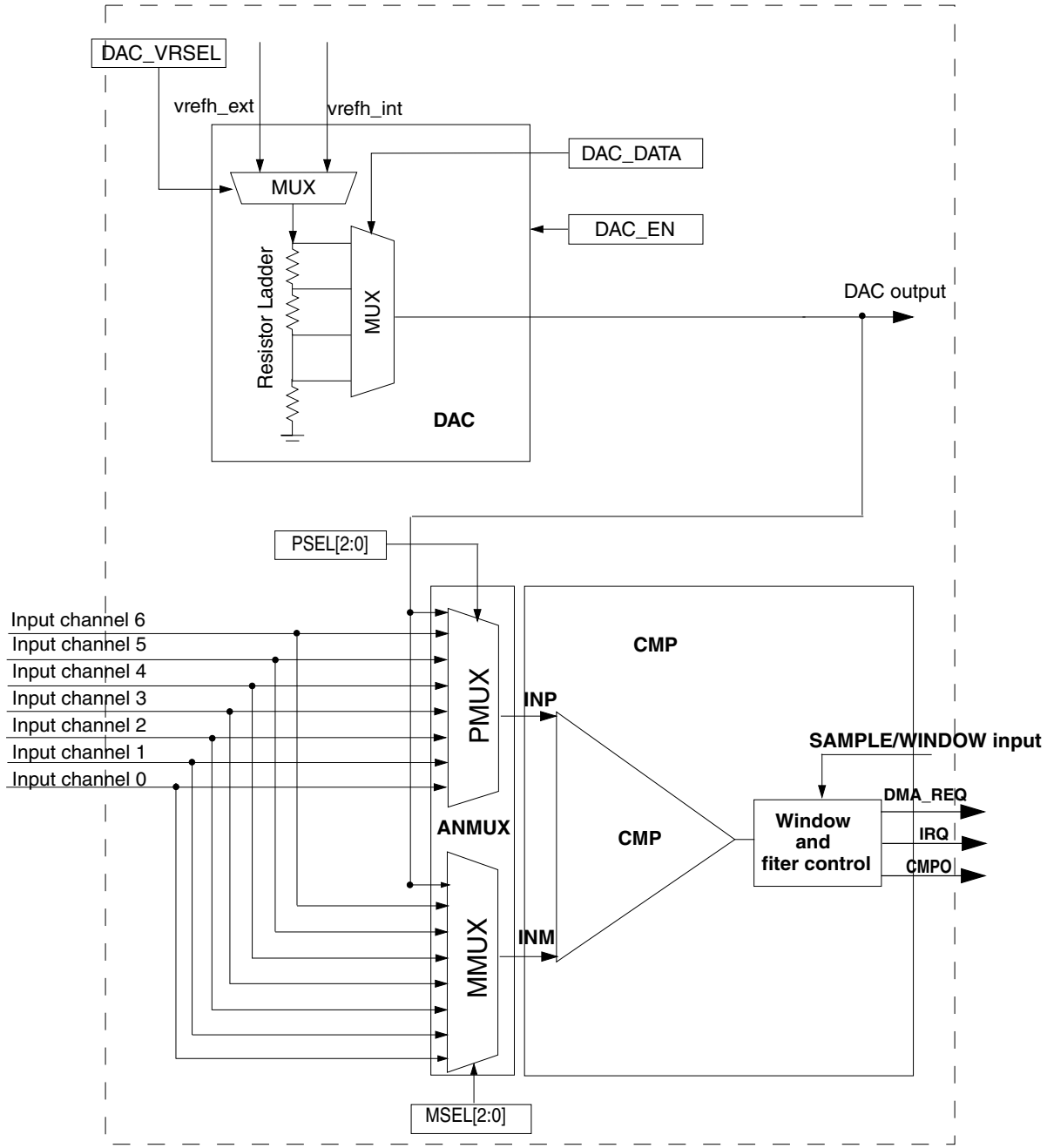



**Figure 47-1. LPCMP Block Diagram**

## 47.5 LPCMP0 register descriptions

The memory map comprises of 32-bit aligned registers which can be accessed via 8-, 16- or 32-bit reads and 32-bit write. Attempted accesses using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error. Read access to reserved address will also generate a transfer error and the read data bus will show all 0s.

## 47.5.1 LPCMP0 Memory map

LPCMP0 base address: 4004_B000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0000_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0001h |
| 8h | Comparator Control Register 0 (CCR0) | 32 | RW | 0000_0002h |
| Ch | Comparator Control Register 1 (CCR1) | 32 | RW | 0000_0000h |
| 10h | Comparator Control Register 2 (CCR2) | 32 | RW | 0000_0000h |
| 18h | DAC Control Register (DCR) | 32 | RW | 0000_0000h |
| 1Ch | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 20h | Comparator Status Register (CSR) | 32 | W1C | 0000_0000h |

## 47.5.2 Version ID Register (VERID)

### 47.5.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 47.5.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 47.5.2.3 Fields

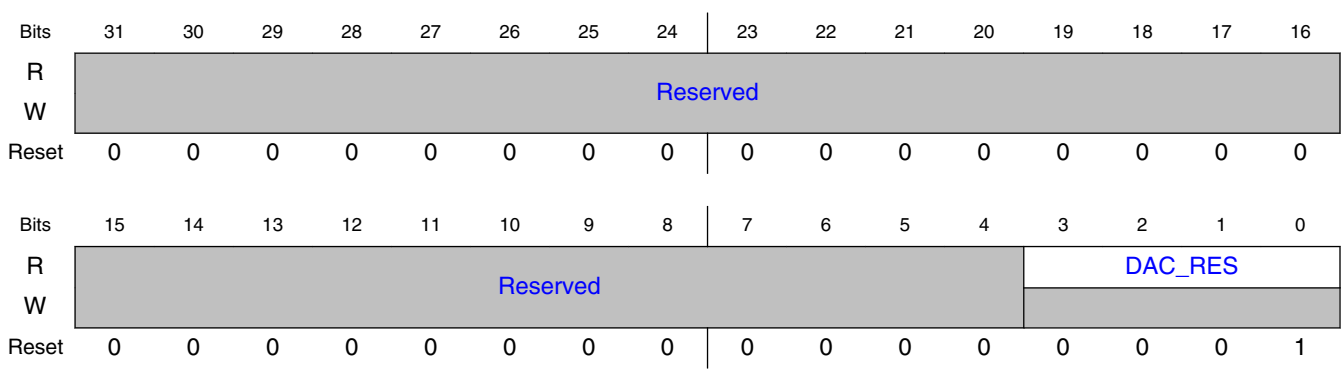
| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number. This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number. This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Round robin feature |

## 47.5.3 Parameter Register (PARAM)

### 47.5.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 47.5.3.2 Diagram




### 47.5.3.3 Fields

| Field | Function |
|---|---|
| 31-4 | Reserved. |

*Table continues on the next page...*

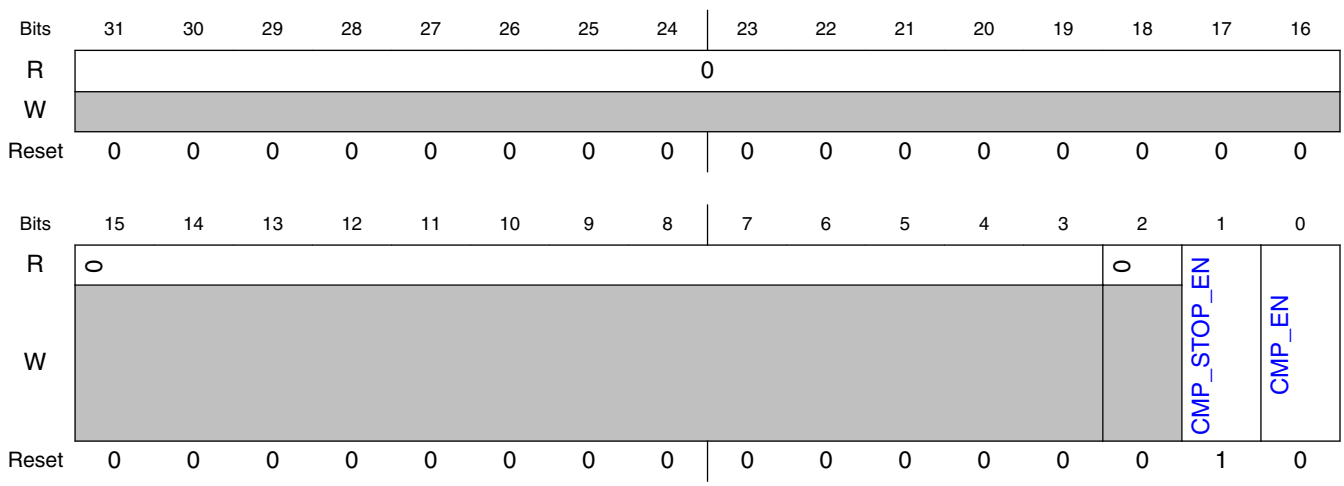
| Field | Function |
|---|---|
| — | |
| 3-0<br>DAC_RES | DAC resolution<br>This read only field indicates the DAC resolution supported by this implementation.<br>0000b - 4 bit DAC<br>0001b - 6 bit DAC<br>0010b - 8 bit DAC<br>0011b - 10 bit DAC<br>0100b - 12 bit DAC<br>0101b - 14 bit DAC<br>0110b - 16 bit DAC |

## 47.5.4 Comparator Control Register 0 (CCR0)

### 47.5.4.1 Offset

| Register | Offset |
|---|---|
| CCR0 | 8h |

### 47.5.4.2 Diagram



### 47.5.4.3 Fields

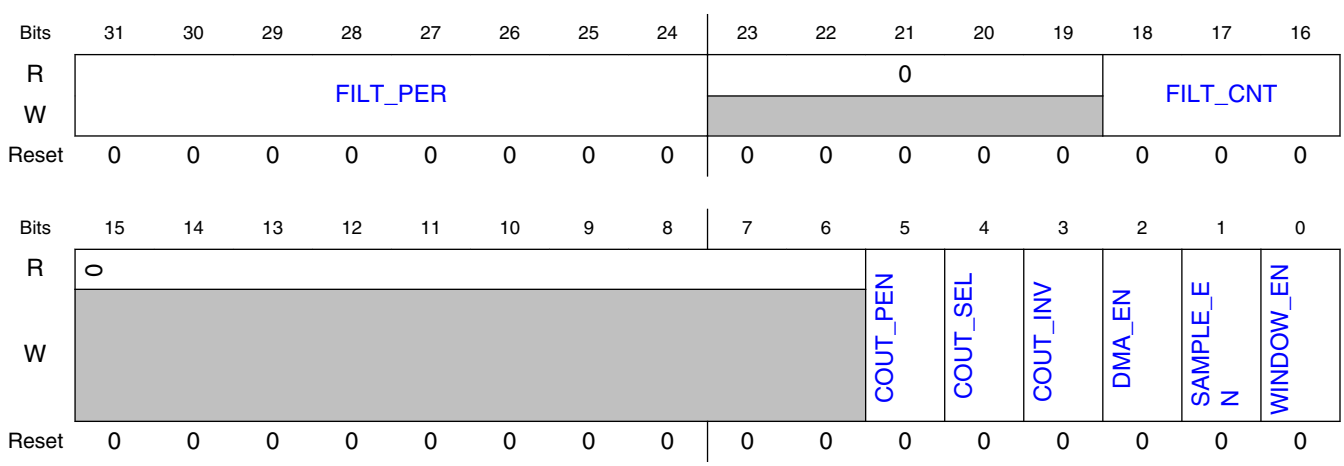
| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>CMP_STOP_EN | Comparator Module STOP Mode Enable<br>This bit decides if user can enable the analog comparator or the DAC when in STOP modes.<br>0b - Comparator is disabled in STOP modes regardless of CMP_EN.<br>1b - Comparator is enabled in STOP mode if CMP_EN is active |
| 0<br>CMP_EN | Comparator Module Enable<br>This bit enables the analog comparator module. When the module is not enabled, the analog part remains in the off state, and consumes no power.<br>0b - Analog Comparator is disabled.<br>1b - Analog Comparator is enabled. |

## 47.5.5 Comparator Control Register 1 (CCR1)

### 47.5.5.1 Offset

| Register | Offset |
|---|---|
| CCR1 | Ch |

### 47.5.5.2 Diagram



### 47.5.5.3 Fields

| Field | Function |
|---|---|
| 31-24<br>FILT_PER | Filter Sample Period<br>Specifies the sampling period (in bus clock cycles) of the comparator output filter, when CCR1[SAMPLE_EN] is 0. Setting FILT_PER to 0x00 bypass the filter when CCR1[SAMPLE_EN] is 0. Filter programming and latency details are provided in the functional description charpter. This field has no effect when CCR1[SAMPLE_EN]= 1. In that case, the external SAMPLE signal is used to determine the sampling period. |
| 23-19<br>— | Reserved |
| 18-16<br>FILT_CNT | Filter Sample Count<br>This field specifies the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, please see the functional description charpter.<br>000b - Filter is disabled. If SAMPLE_EN = 1, then COUT is a logic zero (this is not a legal state in Table 47-2, and is not recommended). If SAMPLE_EN = 0, COUT = COUTA.<br>001b - 1 consecutive sample must agree (comparator output is simply sampled).<br>010b - 2 consecutive samples must agree.<br>011b - 3 consecutive samples must agree.<br>100b - 4 consecutive samples must agree.<br>101b - 5 consecutive samples must agree.<br>110b - 6 consecutive samples must agree.<br>111b - 7 consecutive samples must agree. |
| 15-6<br>— | Reserved |
| 5<br>COUT_PEN | Comparator Output Pin Enable<br>The COUT_PEN bit enables the path from the comparator output to a selected pin.<br>0b - When COUT_PEN is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin.<br>1b - When COUT_PEN is 1, and if the software has configured the comparator to own a packaged pin, the comparator output is available in a packaged pin. |
| 4<br>COUT_SEL | Comparator Output Select<br>0b - Set CMPO to equal COUT (filtered comparator output).<br>1b - Set CMPO to equal COUTA (unfiltered comparator output). |
| 3<br>COUT_INV | Comparator invert<br>This bit allows selecting the polarity of the analog comparator function. It is also driven to the COUT output (on both the device pin and as CSR[COUT]) when CCR0[CMP_EN] is 0.<br>0b - Does not invert the comparator output.<br>1b - Inverts the comparator output. |
| 2<br>DMA_EN | DMA Enable<br>Enables the DMA transfer triggered from the LPCMP module. When this field and the corresponding interrupt enable bit are set, a DMA request is asserted when CFR or CFF is set.<br>0b - DMA is disabled.<br>1b - DMA is enabled. |
| 1<br>SAMPLE_EN | Sample Enable<br>If a write to this register attempts to set both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN is set. |

*Table continues on the next page...*

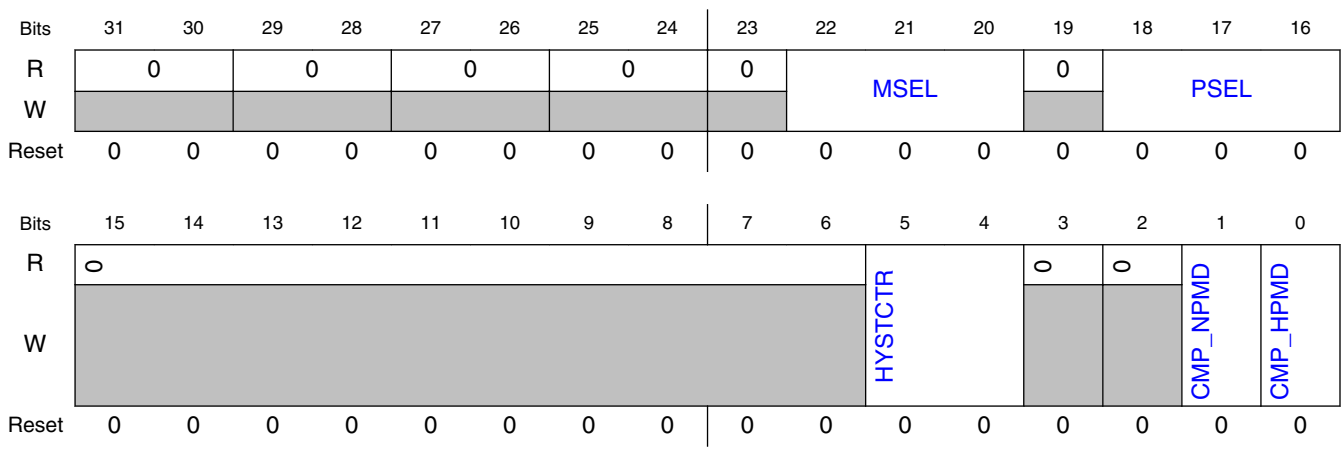
| Field | Function |
|---|---|
| | 0b - Sampling mode is not selected.<br>1b - Sampling mode is selected. |
| 0<br>WINDOW_EN | Windowing Enable<br>If a write to this register attempts to set both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN is set.<br>0b - Windowing mode is not selected.<br>1b - Windowing mode is selected. |

## 47.5.6 Comparator Control Register 2 (CCR2)

### 47.5.6.1 Offset

| Register | Offset |
|---|---|
| CCR2 | 10h |

### 47.5.6.2 Diagram




### 47.5.6.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27-26<br>— | Reserved |
| 25-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-20<br>MSEL | Minus Input MUX Control<br>These bits determine which input is selected for the negative mux. Refer to the chip configuation chapters to get the detailed connections.<br>000b - Input 0<br>001b - Input 1<br>010b - Input 2<br>011b - Input 3<br>100b - Input 4<br>101b - Input 5<br>110b - Input 6<br>111b - Internal DAC output |
| 19<br>— | Reserved |
| 18-16<br>PSEL | Plus Input MUX Control<br>These bits determine which input is selected for the positive mux. Refer to the chip configuation chapters to get the detailed connections.<br>000b - Input 0<br>001b - Input 1<br>010b - Input 2<br>011b - Input 3<br>100b - Input 4<br>101b - Input 5<br>110b - Input 6<br>111b - Internal DAC output |
| 15-6<br>— | Reserved |
| 5-4<br>HYSTCTR | Comparator hard block hysteresis control. See chip data sheet to get the actual hystersis value with each level<br>00b - The hard block output has level 0 hysteresis internally.<br>01b - The hard block output has level 1 hysteresis internally.<br>10b - The hard block output has level 2 hysteresis internally.<br>11b - The hard block output has level 3 hysteresis internally. |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>CMP_NPMD | CMP Nano Power Mode Select<br>0b - Nano Power Comparator is not enabled (mode is determined by CMP_HPMD)<br>1b - Nano Power Comparator is enabled |
| 0 | CMP High Power Mode Select |

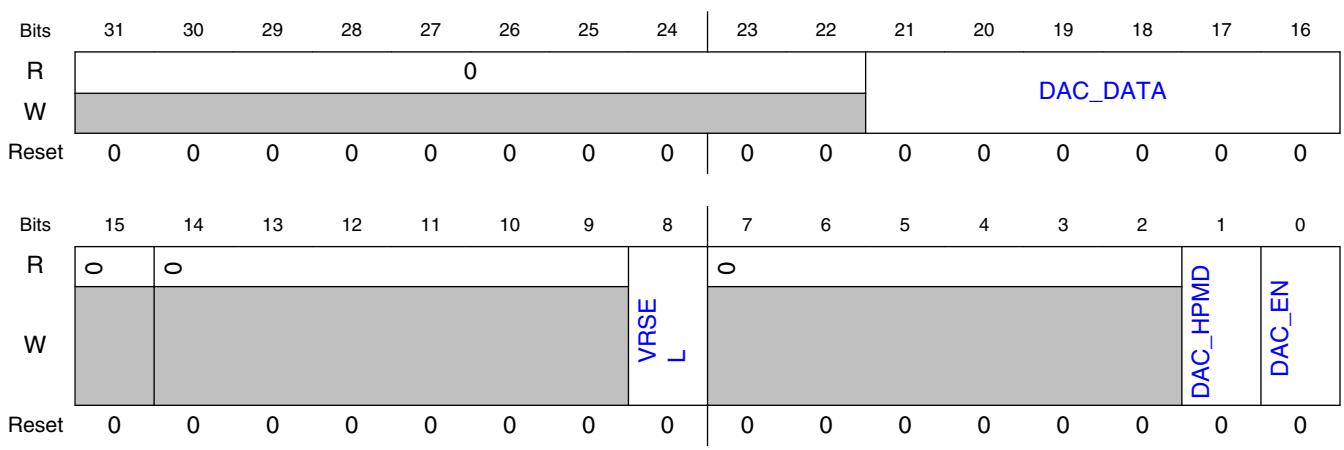
| Field | Function |
|---|---|
| CMP_HPMD | 0b - Low speed comparison mode is selected.(when CMP_NPMD is 0)<br>1b - High speed comparison mode is selected.(when CMP_NPMD is 0) |

## 47.5.7 DAC Control Register (DCR)

### 47.5.7.1 Offset

| Register | Offset |
|---|---|
| DCR | 18h |

### 47.5.7.2 Diagram




### 47.5.7.3 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |
| 21-16<br>DAC_DATA | DAC Output Voltage Select<br>This bit selects an output voltage from one of 64 distinct levels. DACO = (Vin/64) * (DAC_DATA + 1), so the DACO range is from Vin/64 to Vin. |
| 15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14-9<br>— | Reserved |
| 8<br>VRSEL | Supply Voltage Reference Source Select<br>0b - vrefh_int is selected as resistor ladder network supply reference Vin.<br>1b - vrefh_ext is selected as resistor ladder network supply reference Vin. |
| 7-2<br>— | Reserved |
| 1<br>DAC_HPMD | DAC High Power Mode Select<br>This bit is used to enable the DAC high power mode.<br>0b - DAC high power mode is not enabled.<br>1b - DAC high power mode is enabled. |
| 0<br>DAC_EN | DAC Enable<br>This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power.<br>0b - DAC is disabled.<br>1b - DAC is enabled. |

## 47.5.8 Interrupt Enable Register (IER)

### 47.5.8.1 Offset

| Register | Offset |
|---|---|
| IER | 1Ch |

### 47.5.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | CFF_IE | CFR_IE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 47.5.8.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>CFF_IE | Comparator Flag Falling Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFF is set.<br>0b - CFF interrupt is disabled.<br>1b - CFF interrupt is enabled. |
| 0<br>CFR_IE | Comparator Flag Rising Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFR is set.<br>0b - CFR interrupt is disabled.<br>1b - CFR interrupt is enabled. |

## 47.5.9 Comparator Status Register (CSR)

### 47.5.9.1 Offset

| Register | Offset |
|---|---|
| CSR | 20h |

### 47.5.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | COUT | 0 | | | | | | CFF | CFR |
| W | | | | | | | | | | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 47.5.9.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>COUT | Analog Comparator Output<br>Returns the current value of the analog comparator output, when read. The field is reset to 0 and will read as CCR1[COUT_INV] when the analog comparator module is disabled, that is, when CCR0[CMP_EN] = 0. Writing to this field is ignored. |
| 7-2<br>— | Reserved |
| 1<br>CFF | Analog Comparator Flag Falling<br>The CFF bit is set when a falling edge on COUT has been detected. CFF is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFF is cleared when CCR0[CMP_EN] is not set.<br>0b - A falling edge has not been detected on COUT.<br>1b - A falling edge on COUT has occurred. |
| 0<br>CFR | Analog Comparator Flag Rising<br>The CFR bit is set when a rising edge on COUT has been detected. CFR is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFR is cleared when CCR0[CMP_EN] is not set.<br>0b - A rising edge has not been detected on COUT.<br>1b - A rising edge on COUT has occurred. |

## 47.6 LPCMP1 register descriptions

The memory map comprises of 32-bit aligned registers which can be accessed via 8-, 16- or 32-bit reads and 32-bit write. Attempted accesses using unsupported write data sizes, writes to read-only resources, or to reserved spaces are terminated with an error. Read access to reserved address will also generate a transfer error and the read data bus will show all 0s.

### 47.6.1 LPCMP1 Memory map

LPCMP1 base address: 4103_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0000_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0001h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | Comparator Control Register 0 (CCR0) | 32 | RW | 0000_0002h |
| Ch | Comparator Control Register 1 (CCR1) | 32 | RW | 0000_0000h |
| 10h | Comparator Control Register 2 (CCR2) | 32 | RW | 0000_0000h |
| 18h | DAC Control Register (DCR) | 32 | RW | 0000_0000h |
| 1Ch | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 20h | Comparator Status Register (CSR) | 32 | W1C | 0000_0000h |

## 47.6.2 Version ID Register (VERID)

### 47.6.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 47.6.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 47.6.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number. This read only field returns the major version number for the module specification. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>MINOR | Minor Version Number. This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Round robin feature |

## 47.6.3 Parameter Register (PARAM)

### 47.6.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 47.6.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | DAC_RES | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 47.6.3.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved. |
| 3-0<br>DAC_RES | DAC resolution<br>This read only field indicates the DAC resolution supported by this implementation.<br>0000b - 4 bit DAC |

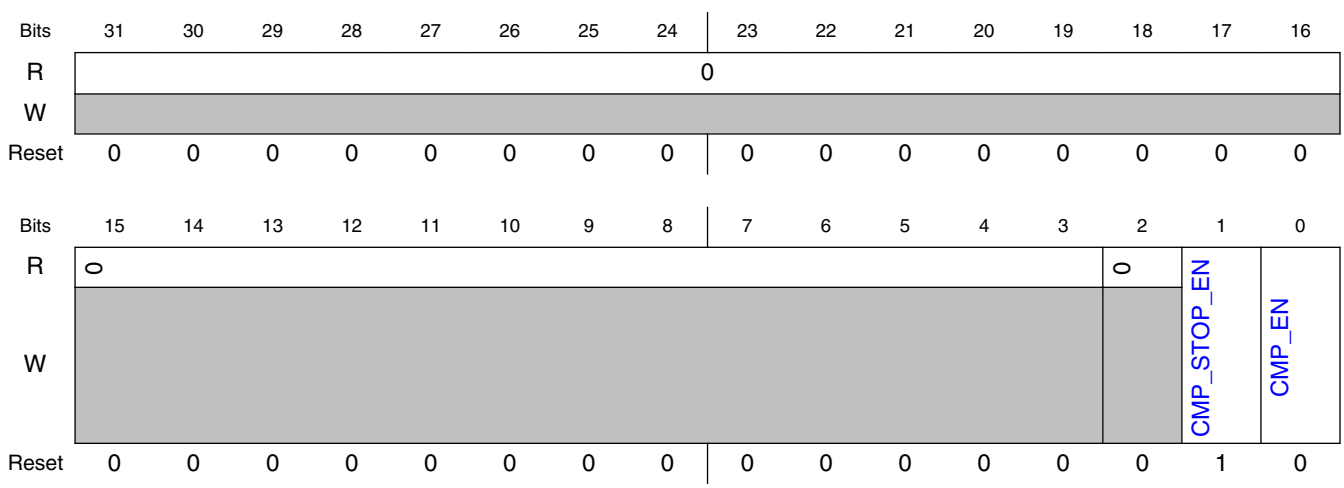
| Field | Function |
|---|---|
| | 0001b - 6 bit DAC<br>0010b - 8 bit DAC<br>0011b - 10 bit DAC<br>0100b - 12 bit DAC<br>0101b - 14 bit DAC<br>0110b - 16 bit DAC |

## 47.6.4 Comparator Control Register 0 (CCR0)

### 47.6.4.1 Offset

| Register | Offset |
|---|---|
| CCR0 | 8h |

### 47.6.4.2 Diagram




### 47.6.4.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |

*Table continues on the next page...*

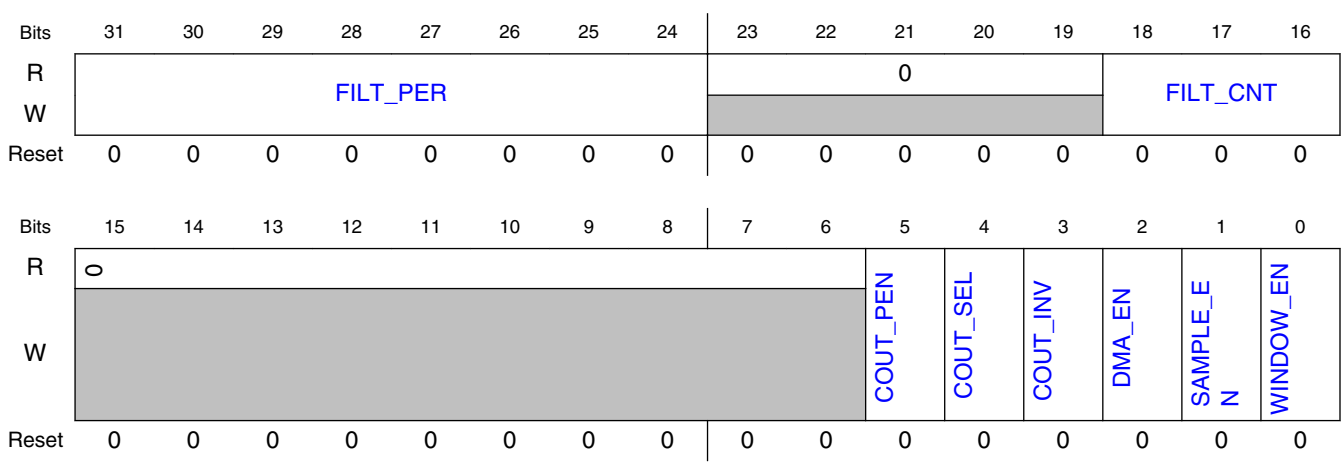
| Field | Function |
|---|---|
| 2<br>— | Reserved |
| 1<br>CMP_STOP_EN | Comparator Module STOP Mode Enable<br>This bit decides if user can enable the analog comparator or the DAC when in STOP modes.<br>0b - Comparator is disabled in STOP modes regardless of CMP_EN.<br>1b - Comparator is enabled in STOP mode if CMP_EN is active |
| 0<br>CMP_EN | Comparator Module Enable<br>This bit enables the analog comparator module. When the module is not enabled, the analog part remains in the off state, and consumes no power.<br>0b - Analog Comparator is disabled.<br>1b - Analog Comparator is enabled. |

## 47.6.5 Comparator Control Register 1 (CCR1)

### 47.6.5.1 Offset

| Register | Offset |
|---|---|
| CCR1 | Ch |

### 47.6.5.2 Diagram



### 47.6.5.3 Fields

| Field | Function |
|---|---|
| 31-24<br>FILT_PER | Filter Sample Period<br>Specifies the sampling period (in bus clock cycles) of the comparator output filter, when CCR1[SAMPLE_EN] is 0. Setting FILT_PER to 0x00 bypass the filter when CCR1[SAMPLE_EN] is 0. Filter programming and latency details are provided in the functional description charpter. This field has no effect when CCR1[SAMPLE_EN]= 1. In that case, the external SAMPLE signal is used to determine the sampling period. |
| 23-19<br>— | Reserved |
| 18-16<br>FILT_CNT | Filter Sample Count<br>This field specifies the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, please see the functional description charpter.<br>000b - Filter is disabled. If SAMPLE_EN = 1, then COUT is a logic zero (this is not a legal state in Table 47-2, and is not recommended). If SAMPLE_EN = 0, COUT = COUTA.<br>001b - 1 consecutive sample must agree (comparator output is simply sampled).<br>010b - 2 consecutive samples must agree.<br>011b - 3 consecutive samples must agree.<br>100b - 4 consecutive samples must agree.<br>101b - 5 consecutive samples must agree.<br>110b - 6 consecutive samples must agree.<br>111b - 7 consecutive samples must agree. |
| 15-6<br>— | Reserved |
| 5<br>COUT_PEN | Comparator Output Pin Enable<br>The COUT_PEN bit enables the path from the comparator output to a selected pin.<br>0b - When COUT_PEN is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin.<br>1b - When COUT_PEN is 1, and if the software has configured the comparator to own a packaged pin, the comparator output is available in a packaged pin. |
| 4<br>COUT_SEL | Comparator Output Select<br>0b - Set CMPO to equal COUT (filtered comparator output).<br>1b - Set CMPO to equal COUTA (unfiltered comparator output). |
| 3<br>COUT_INV | Comparator invert<br>This bit allows selecting the polarity of the analog comparator function. It is also driven to the COUT output (on both the device pin and as CSR[COUT]) when CCR0[CMP_EN] is 0.<br>0b - Does not invert the comparator output.<br>1b - Inverts the comparator output. |
| 2<br>DMA_EN | DMA Enable<br>Enables the DMA transfer triggered from the LPCMP module. When this field and the corresponding interrupt enable bit are set, a DMA request is asserted when CFR or CFF is set.<br>0b - DMA is disabled.<br>1b - DMA is enabled. |
| 1<br>SAMPLE_EN | Sample Enable<br>If a write to this register attempts to set both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN is set.<br>0b - Sampling mode is not selected.<br>1b - Sampling mode is selected. |

*Table continues on the next page...*

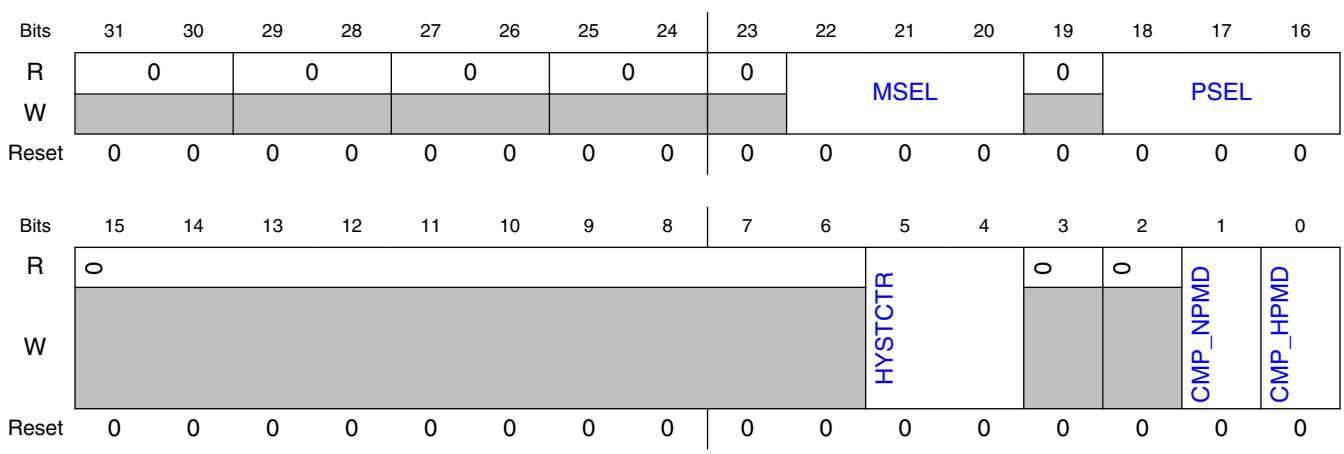
| Field | Function |
|---|---|
| 0<br>WINDOW_EN | Windowing Enable<br>If a write to this register attempts to set both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN is set.<br>0b - Windowing mode is not selected.<br>1b - Windowing mode is selected. |

## 47.6.6 Comparator Control Register 2 (CCR2)

### 47.6.6.1 Offset

| Register | Offset |
|---|---|
| CCR2 | 10h |

### 47.6.6.2 Diagram




### 47.6.6.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 27-26<br>— | Reserved |
| 25-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-20<br>MSEL | Minus Input MUX Control<br>These bits determine which input is selected for the negative mux. Refer to the chip configuation chapters to get the detailed connections.<br>000b - Input 0<br>001b - Input 1<br>010b - Input 2<br>011b - Input 3<br>100b - Input 4<br>101b - Input 5<br>110b - Input 6<br>111b - Internal DAC output |
| 19<br>— | Reserved |
| 18-16<br>PSEL | Plus Input MUX Control<br>These bits determine which input is selected for the positive mux. Refer to the chip configuation chapters to get the detailed connections.<br>000b - Input 0<br>001b - Input 1<br>010b - Input 2<br>011b - Input 3<br>100b - Input 4<br>101b - Input 5<br>110b - Input 6<br>111b - Internal DAC output |
| 15-6<br>— | Reserved |
| 5-4<br>HYSTCTR | Comparator hard block hysteresis control. See chip data sheet to get the actual hystersis value with each level<br>00b - The hard block output has level 0 hysteresis internally.<br>01b - The hard block output has level 1 hysteresis internally.<br>10b - The hard block output has level 2 hysteresis internally.<br>11b - The hard block output has level 3 hysteresis internally. |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>CMP_NPMD | CMP Nano Power Mode Select<br>0b - Nano Power Comparator is not enabled (mode is determined by CMP_HPMD)<br>1b - Nano Power Comparator is enabled |
| 0<br>CMP_HPMD | CMP High Power Mode Select<br>0b - Low speed comparison mode is selected.(when CMP_NPMD is 0) |

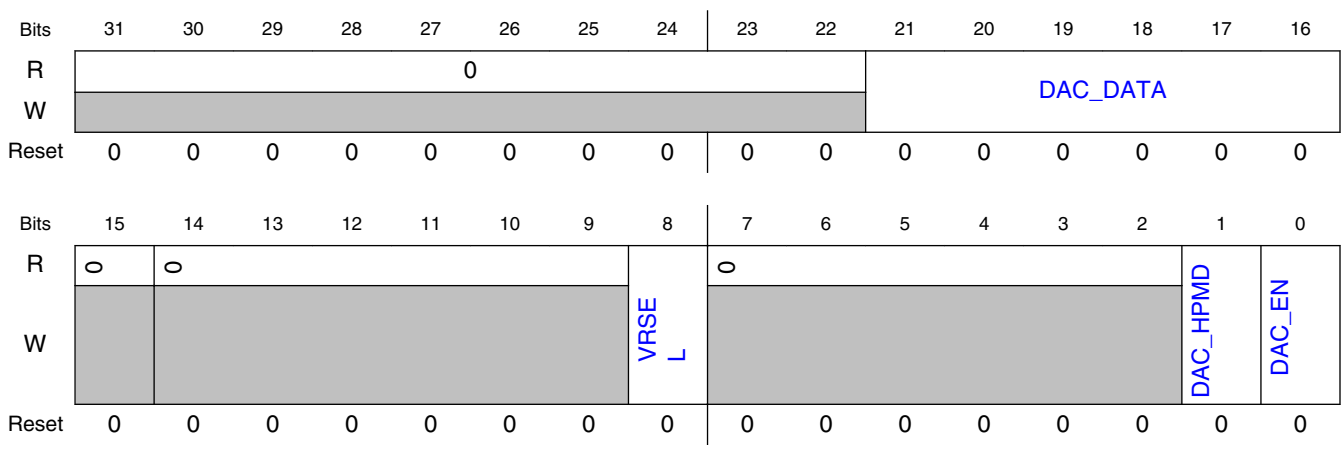
| Field | Function |
|---|---|
| | 1b - High speed comparison mode is selected.(when CMP_NPMD is 0) |

## 47.6.7 DAC Control Register (DCR)

### 47.6.7.1 Offset

| Register | Offset |
|---|---|
| DCR | 18h |

### 47.6.7.2 Diagram




### 47.6.7.3 Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |
| 21-16<br>DAC_DATA | DAC Output Voltage Select<br>This bit selects an output voltage from one of 64 distinct levels. DACO = (Vin/64) * (DAC_DATA + 1), so the DACO range is from Vin/64 to Vin. |
| 15<br>— | Reserved |

*Table continues on the next page...*

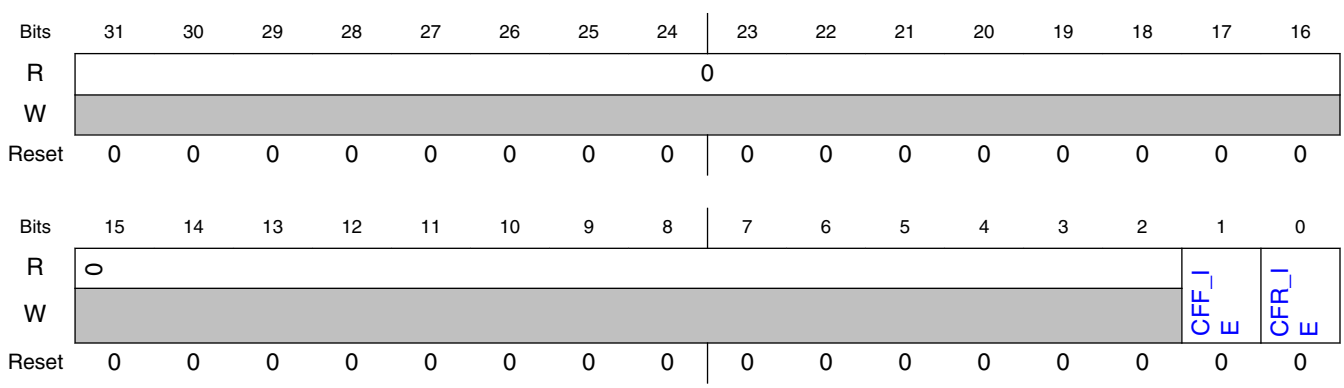
| Field | Function |
|---|---|
| 14-9<br>— | Reserved |
| 8<br>VRSEL | Supply Voltage Reference Source Select<br>0b - vrefh_int is selected as resistor ladder network supply reference Vin.<br>1b - vrefh_ext is selected as resistor ladder network supply reference Vin. |
| 7-2<br>— | Reserved |
| 1<br>DAC_HPMD | DAC High Power Mode Select<br>This bit is used to enable the DAC high power mode.<br>0b - DAC high power mode is not enabled.<br>1b - DAC high power mode is enabled. |
| 0<br>DAC_EN | DAC Enable<br>This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power.<br>0b - DAC is disabled.<br>1b - DAC is enabled. |

## 47.6.8 Interrupt Enable Register (IER)

### 47.6.8.1 Offset

| Register | Offset |
|---|---|
| IER | 1Ch |

### 47.6.8.2 Diagram



### 47.6.8.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>CFF_IE | Comparator Flag Falling Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFF is set.<br>0b - CFF interrupt is disabled.<br>1b - CFF interrupt is enabled. |
| 0<br>CFR_IE | Comparator Flag Rising Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFR is set.<br>0b - CFR interrupt is disabled.<br>1b - CFR interrupt is enabled. |

## 47.6.9 Comparator Status Register (CSR)

### 47.6.9.1 Offset

| Register | Offset |
|---|---|
| CSR | 20h |

### 47.6.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | COUT | 0 | | | | | | CFF | CFR |
| W | | | | | | | | | | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 47.6.9.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>COUT | Analog Comparator Output<br>Returns the current value of the analog comparator output, when read. The field is reset to 0 and will read as CCR1[COUT_INV] when the analog comparator module is disabled, that is, when CCR0[CMP_EN] = 0. Writing to this field is ignored. |
| 7-2<br>— | Reserved |
| 1<br>CFF | Analog Comparator Flag Falling<br>The CFF bit is set when a falling edge on COUT has been detected. CFF is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFF is cleared when CCR0[CMP_EN] is not set.<br>0b - A falling edge has not been detected on COUT.<br>1b - A falling edge on COUT has occurred. |
| 0<br>CFR | Analog Comparator Flag Rising<br>The CFR bit is set when a rising edge on COUT has been detected. CFR is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFR is cleared when CCR0[CMP_EN] is not set.<br>0b - A rising edge has not been detected on COUT.<br>1b - A rising edge on COUT has occurred. |

## 47.7 LPCMP Function Description

### 47.7.1 Functional Block Diagram

The following figure shows the functional block diagram.

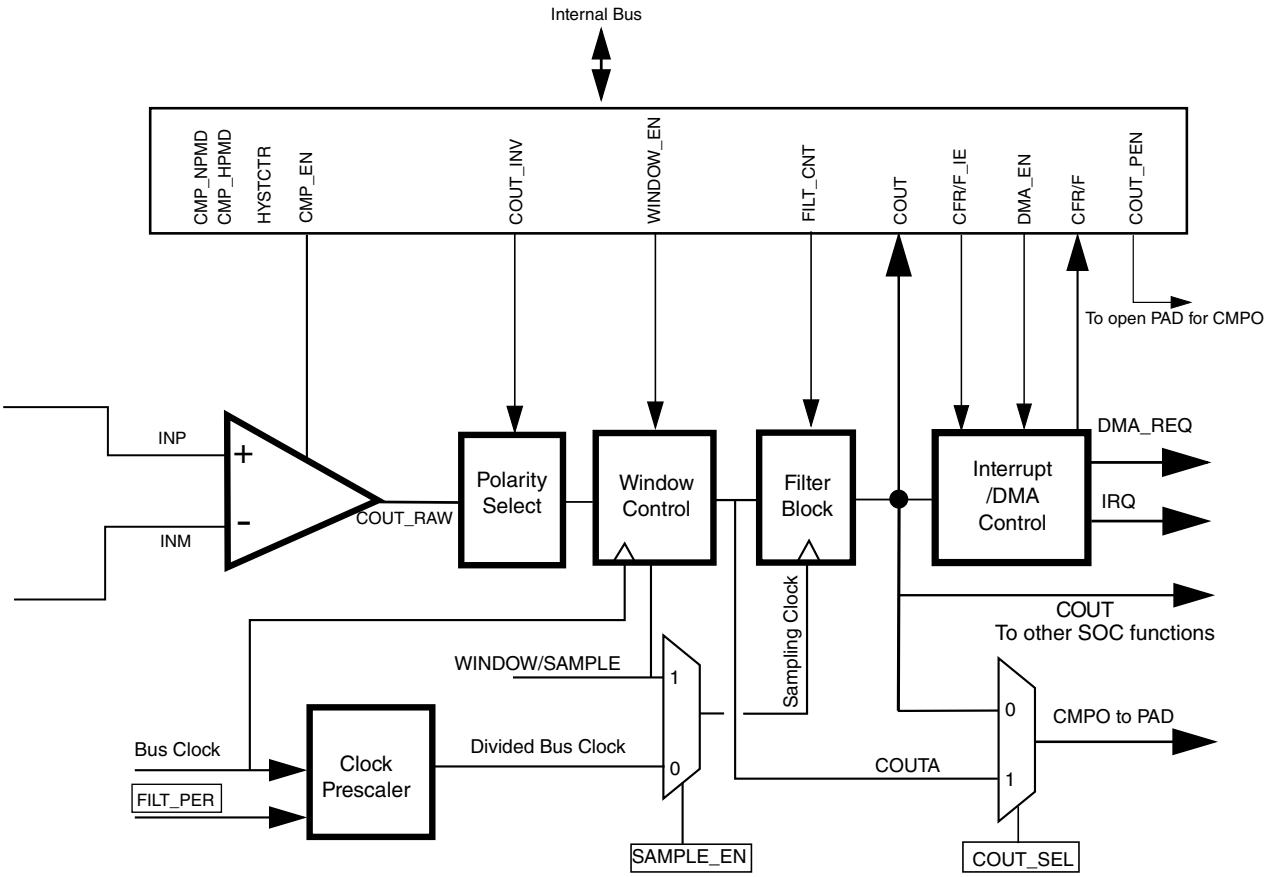



**Figure 47-2. Functional Block Diagram**

In the block diagram:

- Two analog input voltages applied to INP and INM are compared. COUT_RAW is high when the INP input voltage is greater than the INM input, and is low when the INP input voltage is less than the INM input.
- This COUT_RAW signal can be inverted if CCR1[COUT_INV] is enabled.
- If CCR1[WINDOW_EN] is enabled, the optionally inverted comparator output COUT_RAW is sampled on every bus clock when window is enabled to generate COUTA. Sampling does NOT occur when window is disabled. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.
- The window control block can be bypassed if CCR1[WINDOW_EN] is disabled.
- The filter block acts as a simple sampler if CCR1[FILT_CNT] is set to 0x01.
- The filter block acts as a filter based on multiple samples when CCR1[FILT_CNT] is set greater than 0x01.

- If CCR1[SAMPLE_EN] = 1, the external SAMPLE input is used as the sampling clock.
- If CCR1[SAMPLE_EN] = 0, the divided bus clock is used as the sampling clock.

- The filter block can be bypassed when not in use.

```
Bypass_Filter_Block = ((FILT_CNT == 0x00) | (FILT_PER == 0x00)) & ~SAMPLE_EN
```

- Both COUTA and COUT can be configured as module output CMPO by configuring the CCR1[COUT_SEL] bit and are used for different purposes within the system.
- The optionally filtered COUT can be read directly through register bit CSR[COUT].

**NOTE**

See the chip configuration section for the source of SAMPLE/ WINDOW input.

## 47.7.2 Functional Modes

The comparator window and filter features can be combined as shown in the following table. Individual modes are discussed below the table.

**Table 47-2. Functional modes**

| Mode # | CMP_EN | WINDOW_EN | SAMPLE_EN | FILT_CNT | FILT_PER | Operation | Maximum latency[1] |
|---|---|---|---|---|---|---|---|
| 1 | 0 | X | X | X | X | See the Disabled Mode (#1). | N/A |
| 2A | 1 | 0 | 0 | 0x00 | X | See the Continuous Mode (#2A & 2B). | $T_{PD}$ |
| 2B | 1 | 0 | 0 | X | 0x00 | | |
| 3A | 1 | 0 | 1 | 0x01 | X | See the Sampled, Non-Filtered Mode (#3A & 3B). | $T_{PD} + T_{SAMPLE} + T_{per}$ |
| 3B | 1 | 0 | 0 | 0x01 | > 0x00 | | $T_{PD} + (FILT_PER * T_{per}) + T_{per}$ |
| 4A | 1 | 0 | 1 | > 0x01 | X | See the Sampled, Filtered Mode (#4A & 4B). | $T_{PD} + (FILT_CNT * T_{SAMPLE}) + T_{per}$ |
| 4B | 1 | 0 | 0 | > 0x01 | > 0x00 | | $T_{PD} + (FILT_CNT * FILT_PER \times T_{per}) + T_{per}$ |
| 5A | 1 | 1 | 0 | 0x00 | X | See the Windowed Mode (#5A & 5B). | $T_{PD} + 2T_{per}$ |
| 5B | 1 | 1 | 0 | X | 0x00 | | |
| 6 | 1 | 1 | 0 | 0x01 | > 0x00 | See the Windowed/ Resampled Mode (#6). | $T_{PD} + (FILT_PER * T_{per}) + 3T_{per}$ |
| 7 | 1 | 1 | 0 | > 0x01 | > 0x00 | See the Windowed/Filtered Mode (#7). | $T_{PD} + (FILT_CNT * FILT_PER \times T_{per}) + 3T_{per}$ |
| All other combinations of CMP_EN, WINDOW_EN, SAMPLE_EN, FILT_CNT, FILT_PER are illegal. | | | | | | | |

1. $T_{PD}$ represents the intrinsic delay of the analog component plus the polarity select logic. $T_{SAMPLE}$ is the clock period of the external sample clock. $T_{per}$ is the period of the bus clock.

### 47.7.2.1 Disabled Mode (#1)

In disabled mode, the analog comparator is non-functional and consumes no power. CSR[COUT] and CMPO is same as CCR1[COUT_INV] in this mode.

### 47.7.2.2 Continuous Mode (#2A & 2B)

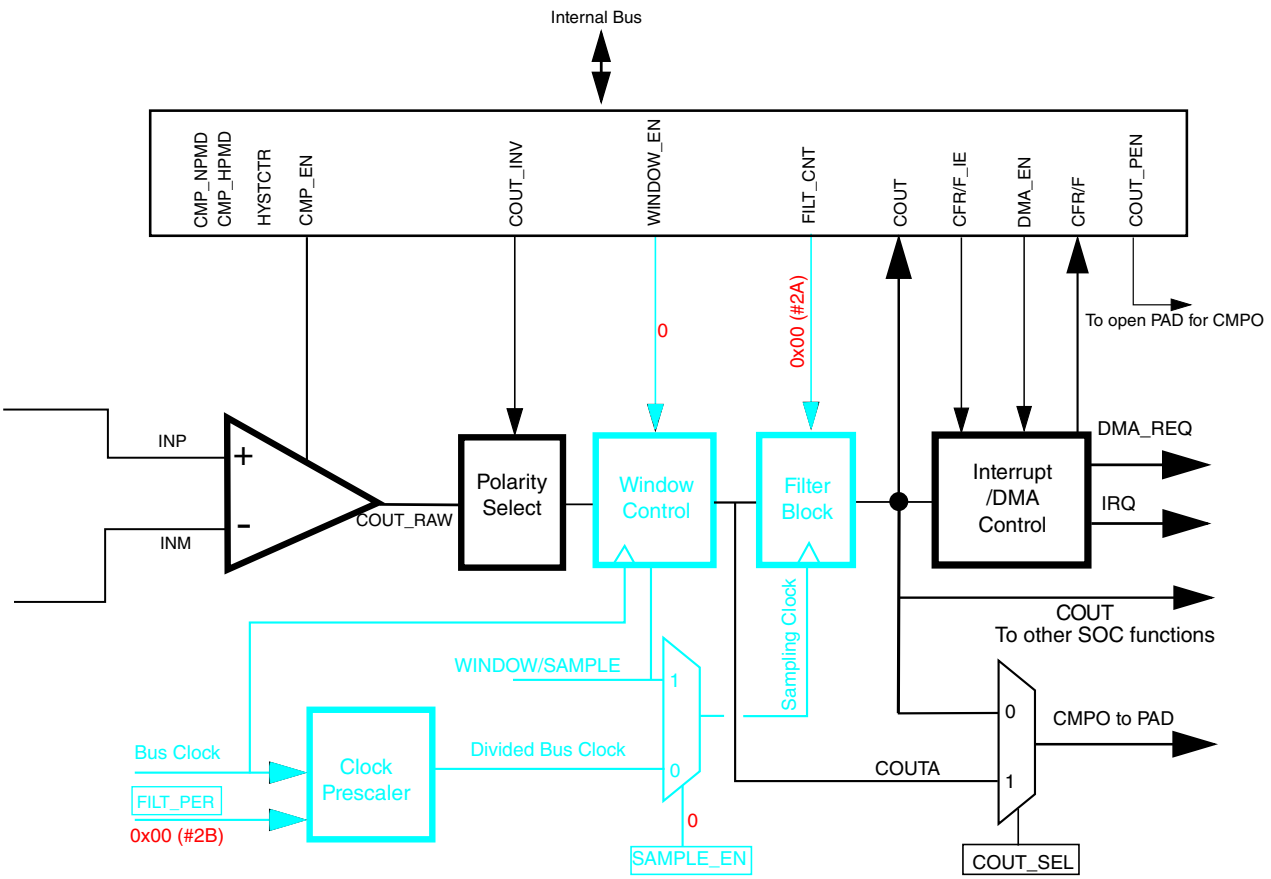



**Figure 47-3. Comparator Operation in Continuous Mode**

In this mode, COUT_RAW may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. CSR[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational(unclocked) mode. COUT and COUTA are identical.

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it should generally be configured to operate in continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

### 47.7.2.3 Sampled, Non-Filtered Mode (#3A & 3B)

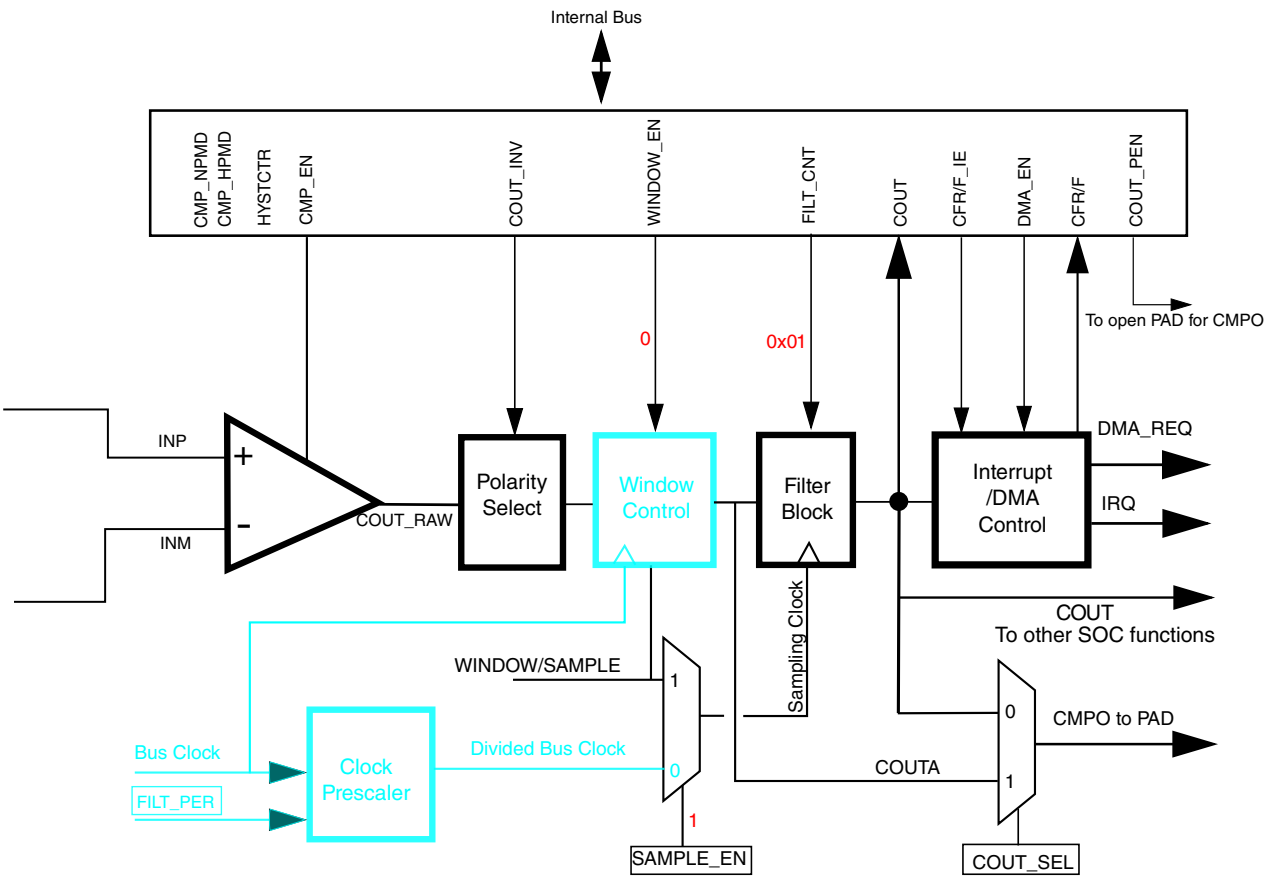



**Figure 47-4. Sampled, Non-Filtered (#3A): Sampling Point Externally Driven**

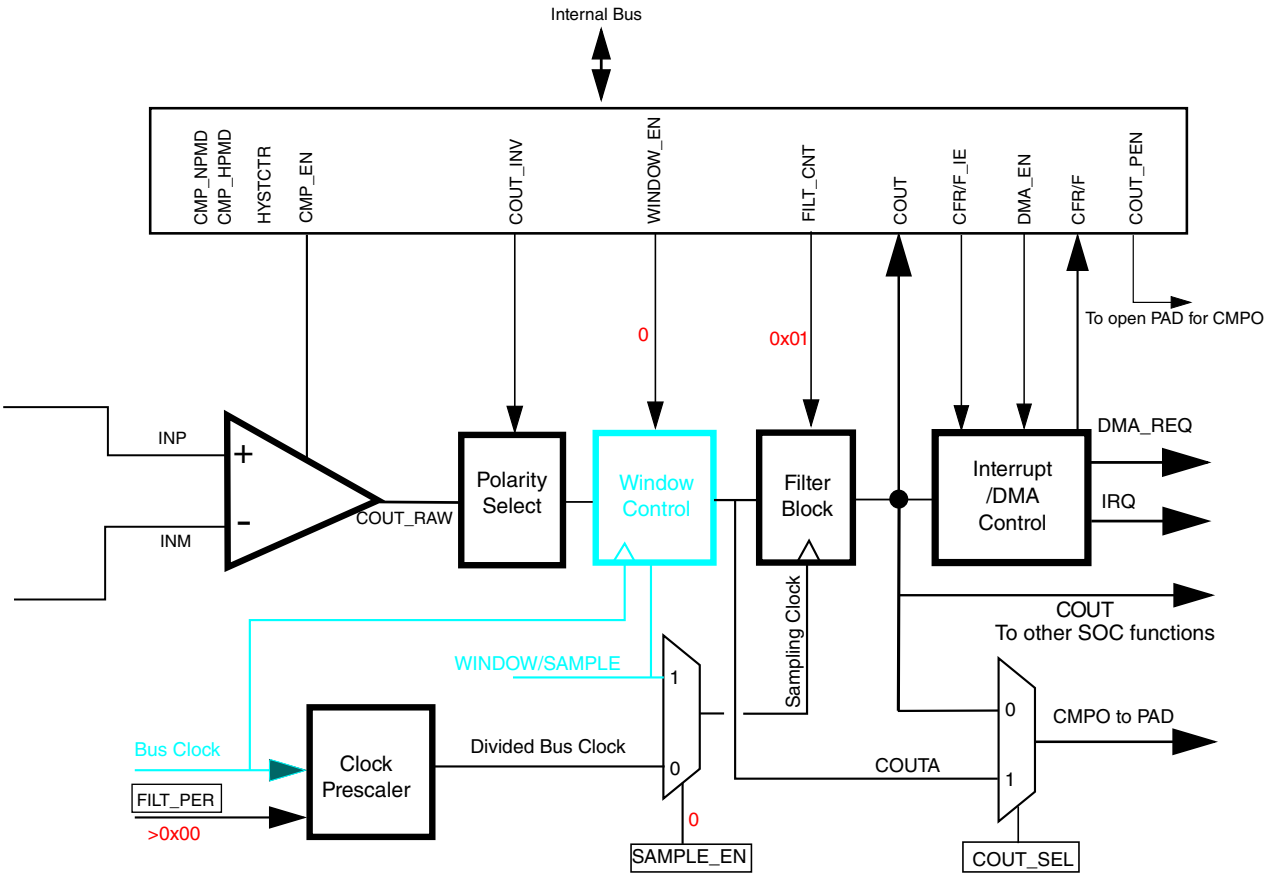



**Figure 47-5. Sampled, Non-Filtered (#3B): Sampling Interval Internally Derived**

In this mode, the path from analog inputs to COUTA is combinational(unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) and sampled, non-filtered (#3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The filter block has no other function than sample/hold of the comparator output in this mode.

## 47.7.2.4 Sampled, Filtered Mode (#4A & 4B)

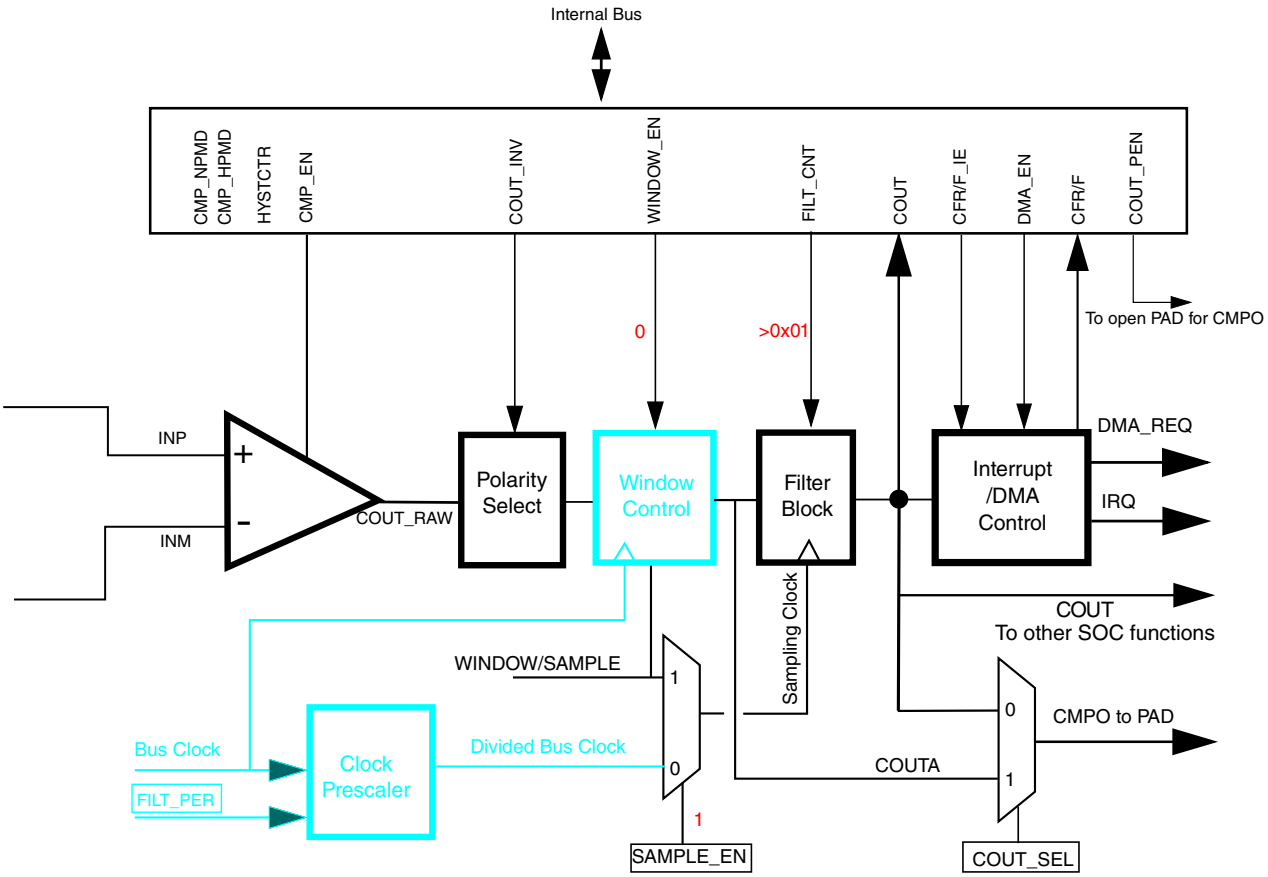



**Figure 47-6. Sampled, Filtered (#4A): Sampling Point Externally Driven**

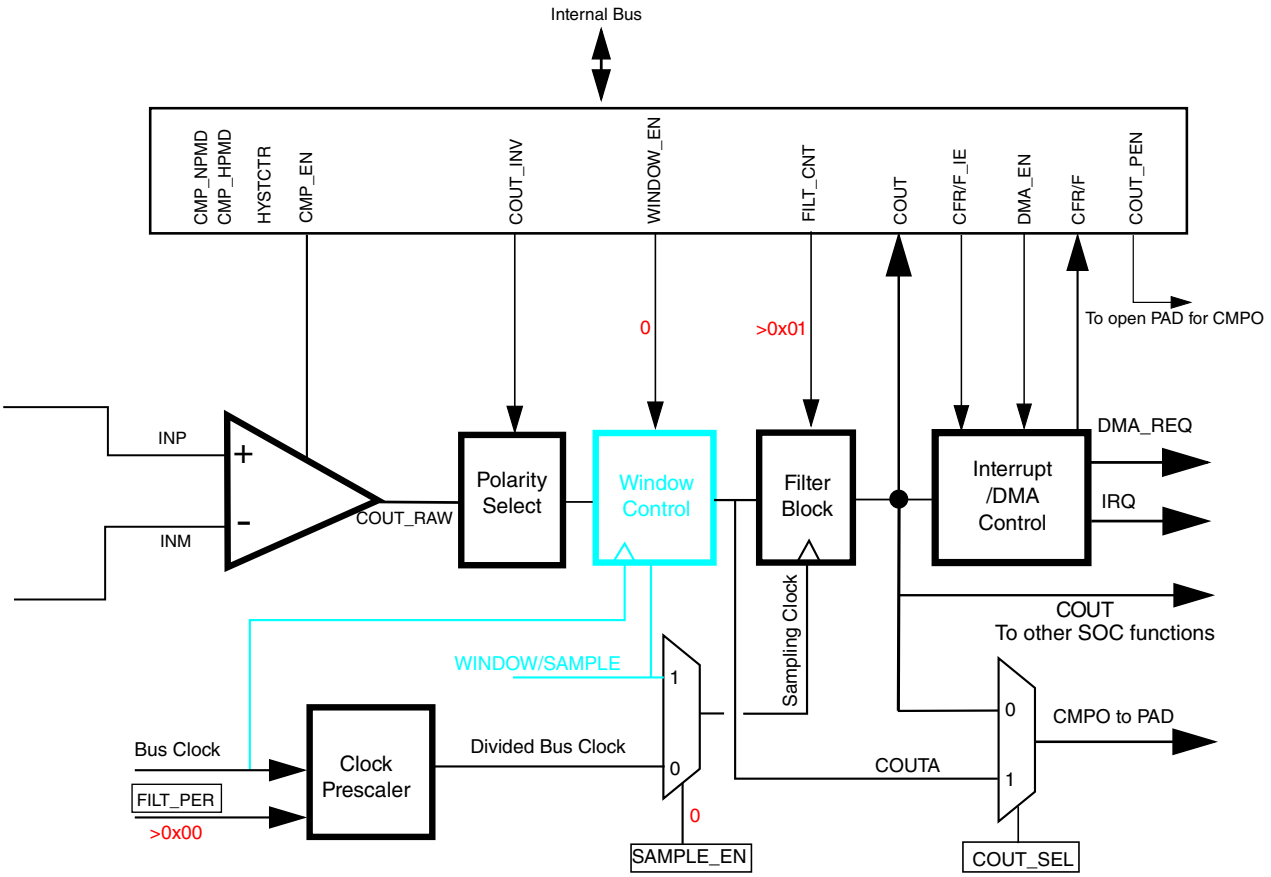



**Figure 47-7. Sampled, Filtered (#4B): Sampling Point Internally Derived**

In this mode, the path from analog inputs to COUTA is combinational(unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) and sampled, filtered (#4A) is that, now, CCR1[FILT_CNT]>1, which activates filter operation.

The only difference in operation between sampled, non-filtered (#3B) and sampled, filtered (#4B) is that now, CCR1[FILT_CNT]>1, which activates filter operation.

## 47.7.2.5 Windowed Mode (#5A & 5B)

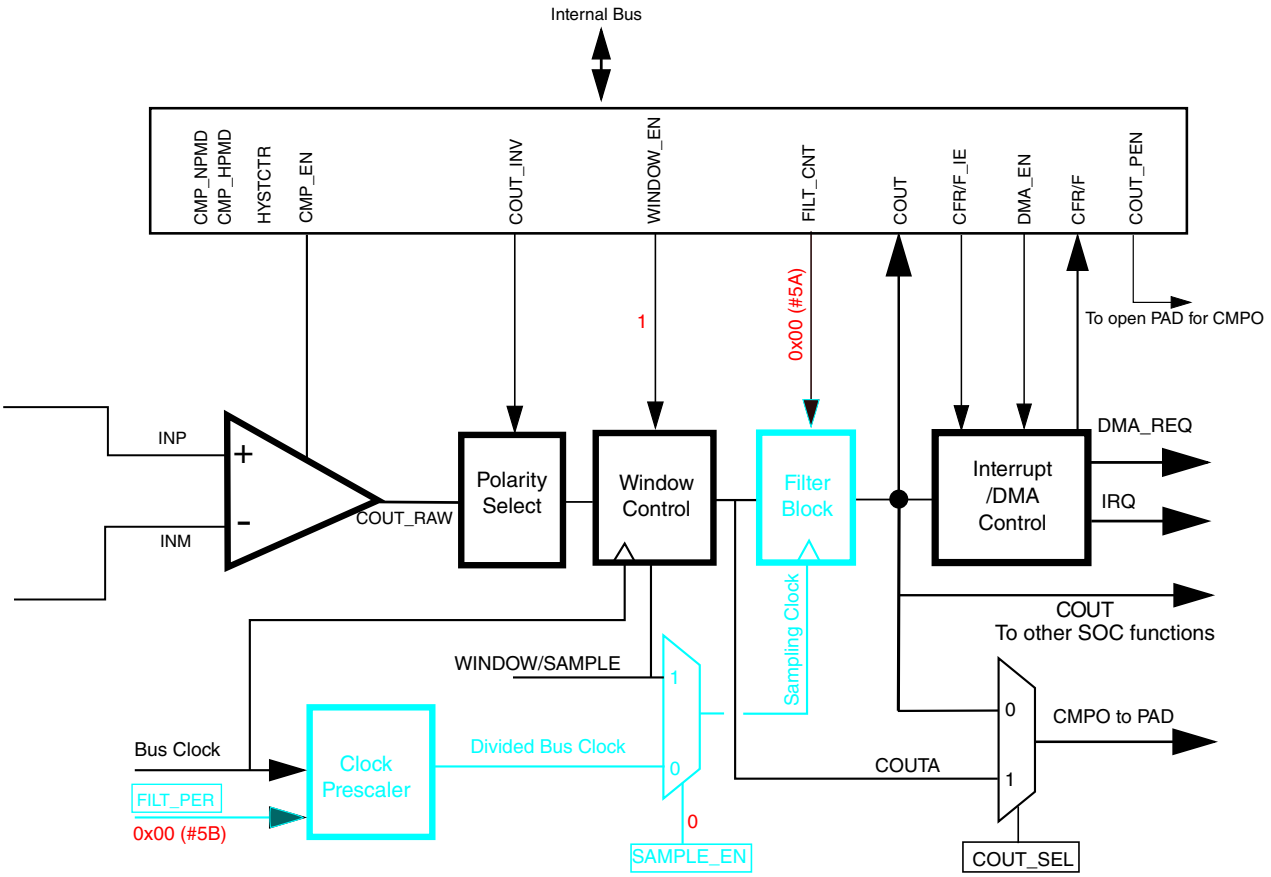



**Figure 47-8. Windowed Mode**

In this mode, COUTA is clocked by the bus clock whenever window is enabled. The last latched value is held when window is disabled. The filter block is bypassed.

The following figure illustrates comparator operation in this mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

In actual operation, COUTA may lag the analog inputs by up to two bus clock cycles plus the combinational path delay through the comparator and polarity select logic.

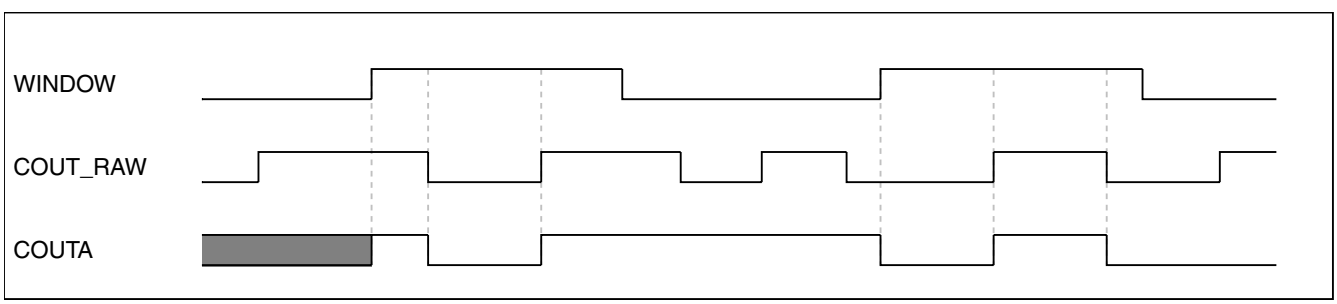



**Figure 47-9. Windowed Mode Timing Diagram**

### 47.7.2.6 Windowed/Resampled Mode (#6)

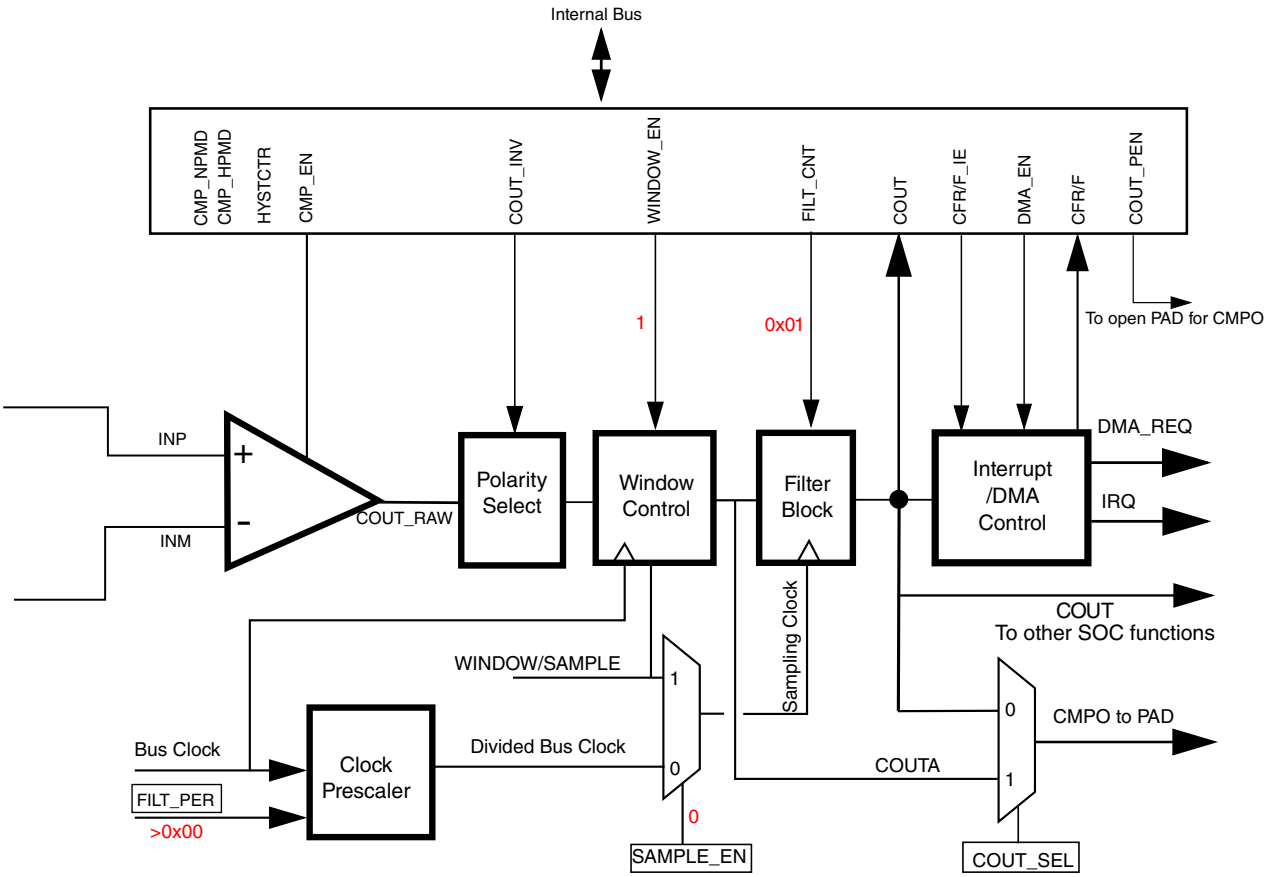



**Figure 47-10. Windowed/Resampled Mode**

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by CCR1[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the windowed/filtered mode shown in the next section. The only difference is that the value of CCR1[FILT_CNT] in this mode must be 1.

The following figure uses the same input stimulus shown in Figure 47-9, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

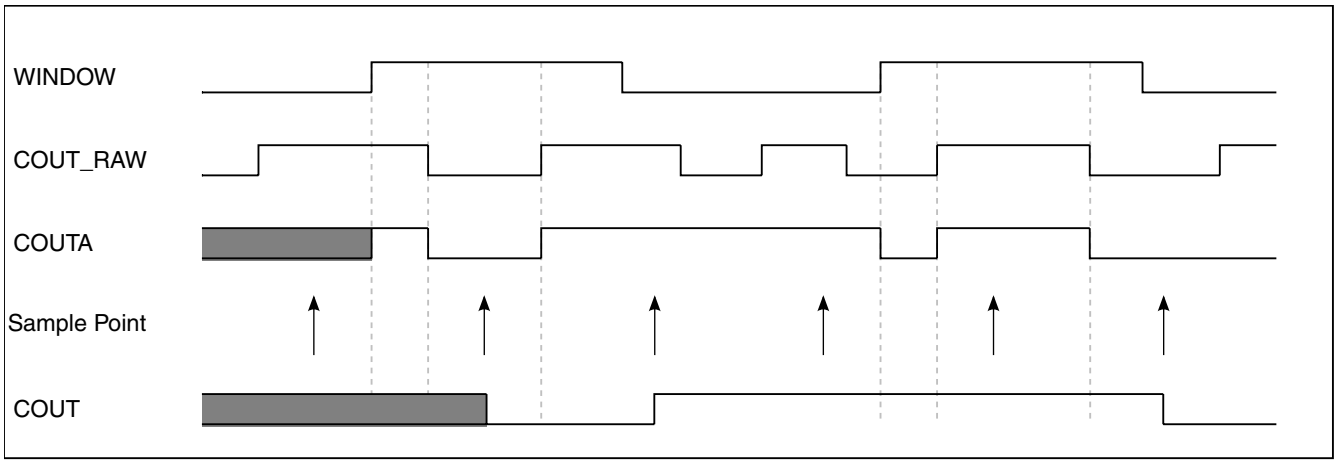



**Figure 47-11. Windowed/Resampled Mode Operation**

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

### 47.7.2.7 Windowed/Filtered Mode (#7)

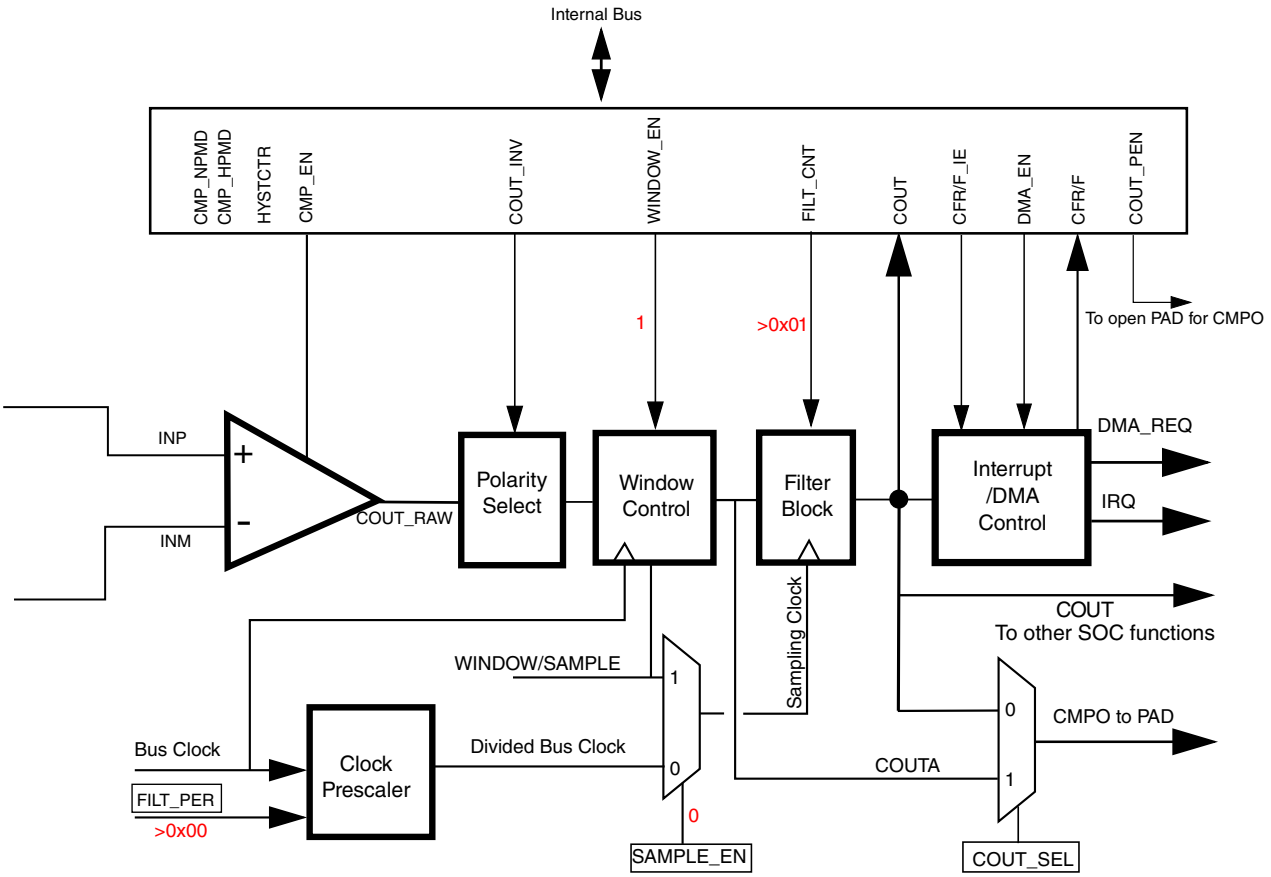



**Figure 47-12. Windowed/Filtered Mode**

The only difference in operation between Windowed/Resampled Mode (#6) and Windowed/Filtered Mode (#7) is that, now, CCR1[FILT_CNT]>1, which activates filter operation.

This is the most complex mode of operation for the comparator block, as it utilizes both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 2 peripheral clock synchronization in the window function + ((CCR1[FILT_CNT] X CCR1[FILT_PER]) + 1) X peripheral clock for the filter function.

## 47.7.3 Hysteresis

The following diagram illustrates implementation of an external hysteresis resistor bridge between the asynchronous comparator output and the INP input of the Comparator. Since positive feedback is required, CCR1[COUT_INV] must be set to 0 when the hysteresis resistor bridge is added to the INP input of the comparator. CCR1[COUT_INV] must be set to 1 when the hysteresis resistor bridge is added to the INM input of the comparator.

The option of adding an external resistor bridge for the purpose of adding hysteresis to the comparator and the amount of hysteresis depends on the user's individual requirements. Hysteresis can be important in some system designs. In the absence of hysteresis, the continuous comparison of nearly identical analog inputs may add noise and waste power by generating high-frequency oscillations at CMPO.

If external hysteresis is added to the comparator, the bridge must be designed to consider other issues than simply how much hysteresis is needed. The resistor values must be sufficiently high so that they do not cause the drive strength of the digital output driver in the CMPO IO cell to be exceeded. Also, if any digital function other than CMPO must operate on the CMPO pad in the presence of such a bridge, the resistor values must be sufficiently high so that the IO cell can function appropriately in its digital role.

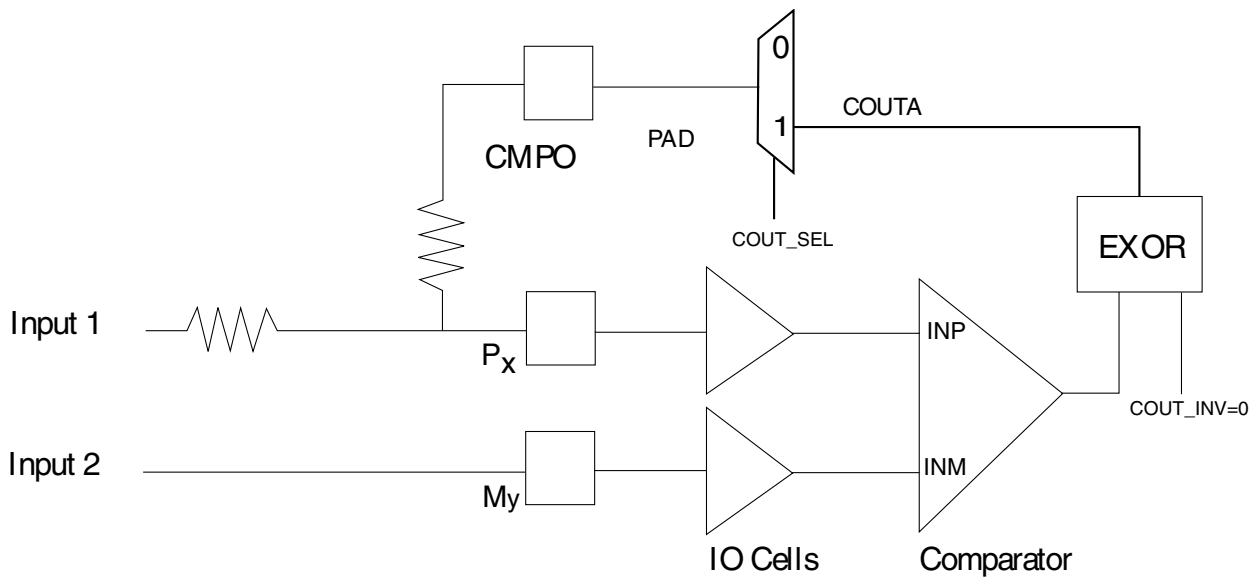



**Figure 47-13. External Hysteresis Circuit**

## 47.7.4 Initialization

To enable the LPCMP module, it is recommended to configure the control registers(CCR1, CCR2, DCR, etc) before setting the CCR0[CMP_EN] bit. To disable the module, just clear the CCR0[CMP_EN] bit. Switch operation modes or change control register bits on the fly(when CCR0[CMP_EN] is 1) may cause some noise on the COUT or COUTA. So it is recommended to disable the module before mode switch or any control bits change.

Interrupts should be disabled during power up, recovery from partial-power-down, and while re-enabling the module. Failure to do so could result in spurious interrupts.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter. See the datasheet for power-on delays of the comparators. The delay caused by windowing and filter function is specified in the Table 47-2.

During operation, the propagation delay of the selected data paths must always be considered. It can take many bus clock cycles for COUT and CSR[CFR]/CSR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT will initially be equal to zero until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic one.

## 47.7.5 Low-Pass Filter

The low-pass filter operates on the unfiltered, optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by CCR1[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

### 47.7.5.1 Enabling Filter Modes

Filter modes can be enabled by:

- Setting CCR1[FILT_CNT] > 0x01 and
- Setting CCR1[FILT_PER] to a nonzero value or setting CCR1[SAMPLE_EN]=1

If using the divided bus clock to drive the filter, it samples COUTA every CCR1[FILT_PER] bus clock cycles.

The filter output is at logic zero when first initalized, and subsequently changes when all the consecutive CCR1[FILT_CNT] samples agree that the output value has changed. In other words, CMP_C0[COUT] is zero for some initial period, even when COUTA is at logic one.

If CCR1[SAMPLE_EN]=1, the filter samples COUTA on each positive transition of the SAMPLE input. The output state of the filter changes when all the consecutive CCR1[FILT_CNT] samples agree that the output value has changed.

#### 47.7.5.2 Latency Issues

The value of CCR1[FILT_PER] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CCR1[FILT_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CCR1[FILT_CNT].

The values of CCR1[FILT_PER] or SAMPLE period and CCR1[FILT_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CCR1[FILT_CNT].

Table 47-2 summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency is restarted each time an actual output transition is masked by noise.

### 47.7.6 Interrupts

When corresponding interrupt enable register bit is set, flags CSR[CFR], CSR[CFF] can generate interrupt, assuming the DMA enable bit is not set. The interrupt is de-asserted by clearing either the flag or the interrupt enable register bit.

### 47.7.7 DMA Support

When DMA support is enabled by setting CCR1[DMA_EN] and the interrupt is enabled by setting IER[CFR_IE], IER[CFF_IE], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has

completed the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes if CCR0[CMP_STOP_EN] is set. A DMA transfer request will wake up the system from STOP modes. After the data transfer has finished, the system will go back again to STOP modes. Refer to the DMA chapters in the device reference manual for the asynchronous DMA function for details.

## 47.7.8 Low Power Mode Operation

The following table shows the WAIT mode and the STOP mode operation of the LPCMP module.

**Table 47-3. Low Power Modes Operation**

| Modes of Operation | Description |
|---|---|
| WAIT | The LPCMP module can operate normally with window and filter function. |
| STOP | The LPCMP module can only operate in continuous mode. |

# Chapter 48
# 12-bit Low Power Digital-to-Analog Converter (LPDAC)

## 48.1 Chip-specific LPDAC information

### 48.1.1 12-bit LPDAC Overview

This device contains 1 instance of a 12-bit low-power digital-to-analog converter (LPDAC) with programmable reference generator output. The LPDAC includes a FIFO for DMA support.

### 48.1.2 12-bit LPDAC Output

The output of the LPDAC can be placed on an external pin or set as one of the inputs to the analog comparator or LPADC.

### 48.1.3 12-bit LPDAC Reference

For this device VREF_OUT and VDDA are selectable as the LPDAC reference. VREF_OUT is connected to the VREFH_INT input and VDDA is connected to the VREFH_EXT input. Use LPDAC_GCR[DACRFS] control bit to select between these two options.

Be aware that if the LPDAC and ADC use the VREF_OUT reference simultaneously, some degradation of ADC accuracy is to be expected due to LPDAC switching.

## 48.2 Introduction

The LPDAC is a low-power, 12-bit general-purpose digital-to-analog converter. The output of the LPDAC can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

## 48.3 Features

The features of the LPDAC module include:

- On-chip programmable reference generator output. The voltage output range is from 1⁄4096 $V_{in}$ to $V_{in}$, and the step is 1⁄4096 $V_{in}$, where $V_{in}$ is the input reference voltage.
- $V_{in}$ can be selected from two reference sources.
- 16-word depth FIFO supported with configurable watermark.
- Multiple operation modes.
  - Buffer mode
  - FIFO mode
  - Swing back mode
- Software trigger and hardware trigger supported.
- Selectable performance levels: low power mode and high power mode.
- Interrupt and DMA support.

## 48.4 Block diagram

The block diagram of the LPDAC module is as follows:

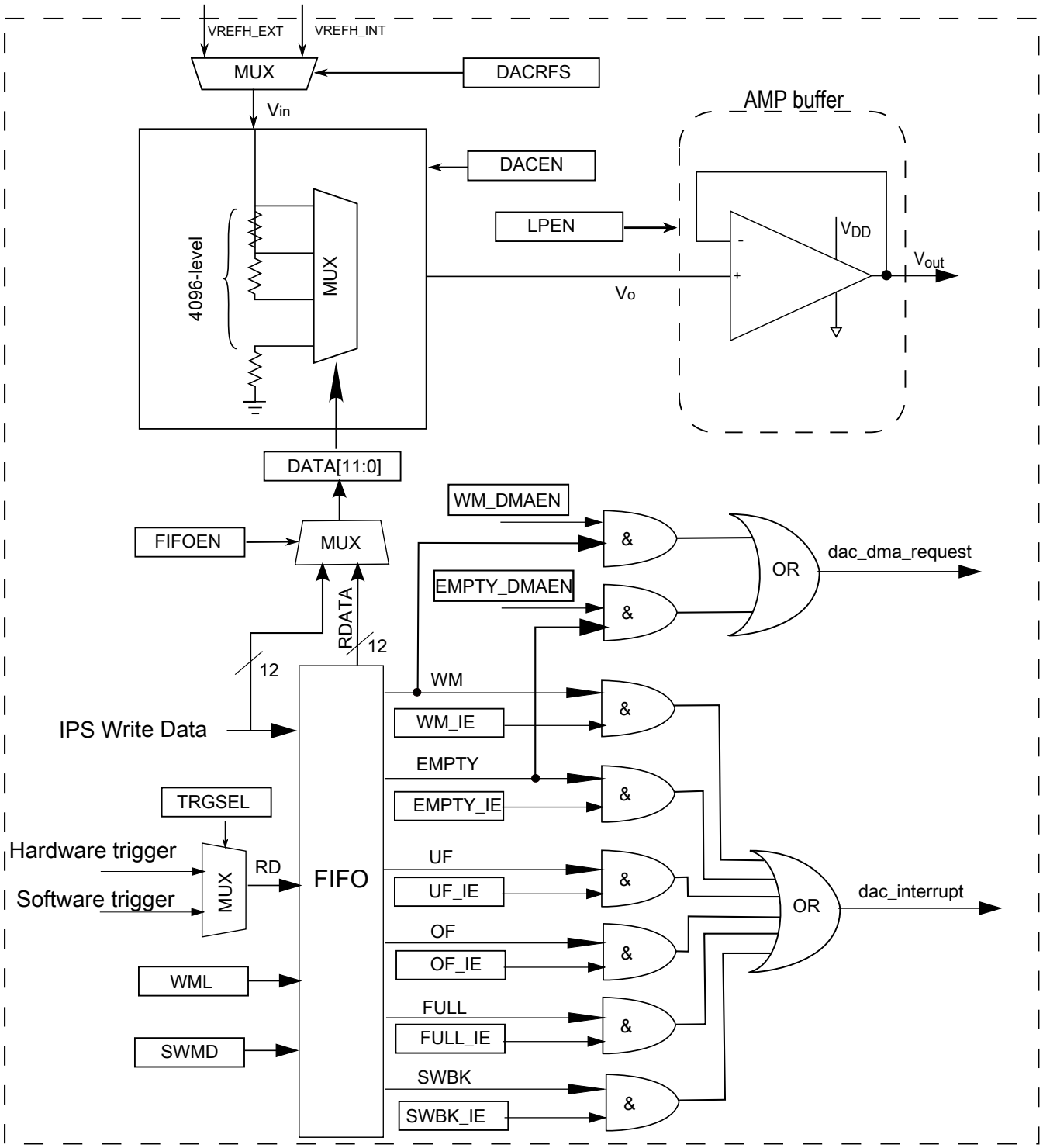



**Figure 48-1. LPDAC Block Diagram**

## 48.5 LPDAC register descriptions

Access to an address outside the valid memory map will generate a bus error.

## 48.5.1 LPDAC Memory map

LPDAC0 base address: 4004_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version Identifier Register (VERID) | 32 | RO | 0000_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0003h |
| 8h | DAC Data Register (DATA) | 32 | WORZ | 0000_0000h |
| Ch | DAC Global Control Register (GCR) | 32 | RW | 0000_0000h |
| 10h | DAC FIFO Control Register (FCR) | 32 | RW | 0000_0000h |
| 14h | DAC FIFO Pointer Register (FPR) | 32 | RO | 0000_0000h |
| 18h | FIFO Status Register (FSR) | 32 | W1C | 0000_0002h |
| 1Ch | DAC Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 20h | DAC DMA Enable Register (DER) | 32 | RW | 0000_0000h |
| 24h | DAC Reset Control Register (RCR) | 32 | RW | 0000_0000h |
| 28h | DAC Trigger Control Register (TCR) | 32 | WORZ | 0000_0000h |

## 48.5.2 Version Identifier Register (VERID)

### 48.5.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 48.5.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 48.5.2.3 Fields

| Field | Function |
|---|---|
| 31-24 | Major version number |
| MAJOR | This read-only field returns the major version number for the module specification. |
| 23-16 | Minor version number |
| MINOR | This read-only field returns the minor version number for the module specification. |
| 15-0 | Feature Identification Number |
| FEATURE | This read-only field returns the feature set number. |

## 48.5.3 Parameter Register (PARAM)

### 48.5.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

## 48.5.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | FIFOSZ | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

## 48.5.3.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>FIFOSZ | FIFO size<br>The FIFO depth is 2^(FIFOSZ+1) words.<br>000b - Reserved<br>001b - FIFO depth is 4<br>010b - FIFO depth is 8<br>011b - FIFO depth is 16<br>100b - FIFO depth is 32<br>101b - FIFO depth is 64<br>110b - FIFO depth is 128<br>111b - FIFO depth is 256 |

# 48.5.4 DAC Data Register (DATA)

## 48.5.4.1 Offset

| Register | Offset |
|---|---|
| DATA | 8h |

### 48.5.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | 0 | | | | | | | | | | | |
| W | | | | | DATA | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 48.5.4.3 Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>DATA | In FIFO mode or swing back mode, this is the FIFO data entry. In buffer mode, write to this field will push the data to analog without trigger support. This field is write only and always read zero. |

## 48.5.5 DAC Global Control Register (GCR)

### 48.5.5.1 Offset

| Register | Offset |
|---|---|
| GCR | Ch |

## 48.5.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | 0 | | | | 0 | 0 | TRGSEL | SWMD | FIFOEN | LPEN | DACRFS | DACEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 48.5.5.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8 | Reserved |

*Table continues on the next page...*

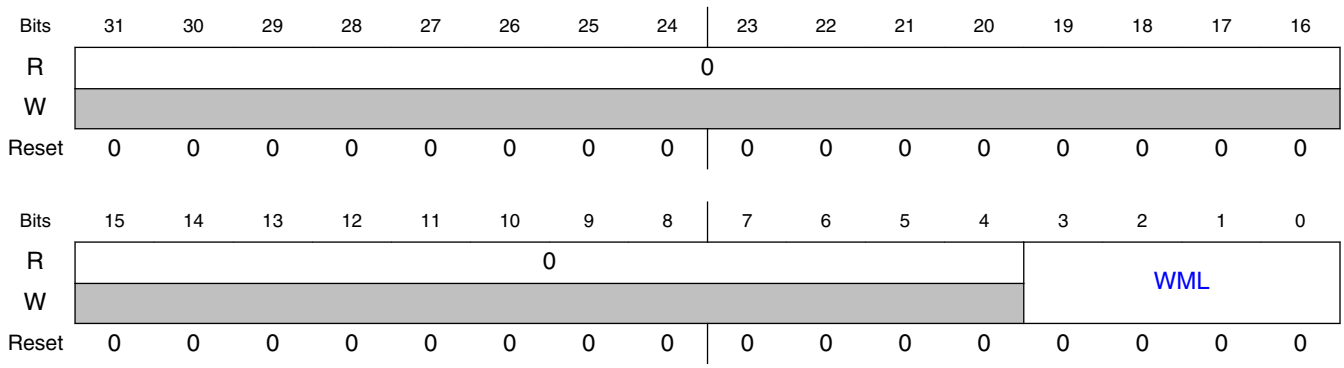
| Field | Function |
|---|---|
| — | |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>TRGSEL | DAC Trigger Select<br>0b - The DAC hardware trigger is selected.<br>1b - The DAC software trigger is selected. |
| 4<br>SWMD | Swing Back Mode<br>0b - Swing back mode disable<br>1b - Swing back mode enable |
| 3<br>FIFOEN | FIFO Enable<br>0b - FIFO mode is disabled and buffer mode is enabled. Any data written to DATA[DATA] goes to buffer then goes to conversion.<br>1b - FIFO mode is enabled. Data will be first read from FIFO to buffer then goes to conversion |
| 2<br>LPEN | Low Power Enable<br>**NOTE:** See the DAC electrical characteristics of the device data sheet for details on the impact of the modes below.<br>0b - High-Power mode<br>1b - Low-Power mode |
| 1<br>DACRFS | DAC Reference Select<br>0b - The DAC selects VREFH_INT as the reference voltage.<br>1b - The DAC selects VREFH_EXT as the reference voltage. |
| 0<br>DACEN | DAC Enable<br>Starts the Programmable Reference Generator operation.<br>**NOTE:** For 8-bit or 16-bit register write, it is recommended to enable this bit after configuring other bytes of GCR.<br>0b - The DAC system is disabled.<br>1b - The DAC system is enabled. |

## 48.5.6 DAC FIFO Control Register (FCR)

### 48.5.6.1 Offset

| Register | Offset |
|---|---|
| FCR | 10h |

### 48.5.6.2 Diagram




### 48.5.6.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>WML | Watermark Level<br>When the number of data remain in FIFO is equal to or less than this level, the WM flag will be set.<br>**NOTE:** This field should not be configured to 0, otherwise the watermark feature is not enabled. |

## 48.5.7 DAC FIFO Pointer Register (FPR)

### 48.5.7.1 Offset

| Register | Offset |
|---|---|
| FPR | 14h |

### 48.5.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FIFO_WPT | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FIFO_RPT | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 48.5.7.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>FIFO_WPT | FIFO Write Pointer |
| 15-4<br>— | Reserved |
| 3-0<br>FIFO_RPT | FIFO Read Pointer |

## 48.5.8 FIFO Status Register (FSR)

### 48.5.8.1 Offset

| Register | Offset |
|---|---|
| FSR | 18h |

## 48.5.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | UF | OF | 0 | | SWBK | WM | EMPTY | FULL |
| W | | | | | | | | | W1C | W1C | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

## 48.5.8.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>UF | FIFO Underflow Flag<br>This flag means that there is a new trigger after EMPTY flag is set. The FIFO read pointer will not increase in this case and the data sent to DAC analog conversion will not changed. This flag is cleared by writing a 1 to it.<br>0b - No underflow has occurred since the last time the flag was cleared.<br>1b - At least one trigger underflow has occurred since the last time the flag was cleared. |
| 6<br>OF | FIFO Overflow Flag<br>This flag indicates that data is intended to write into FIFO after FULL flag is set. The writer pointer will not increase in this case. The extra data will not be written into the FIFO. This flag is cleared by writing a 1 to it.<br>0b - No overflow has occurred since the last time the flag was cleared.<br>1b - At least one FIFO overflow has occurred since the last time the flag was cleared. |
| 5-4<br>— | Reserved |
| 3<br>SWBK | Swing Back One Cycle Complete Flag<br>This flag indicates that the DAC has completed one period of conversion in swing back mode. It means that the read pointer has increased to the top (write pointer) once and then decreased to zero once. For example, after three data is written to FIFO, the writer pointer is now 3. Then, if continually triggered, the read pointer will swing like: 0-1-2-1-0-1-2-, and so on. After the fourth trigger, the flag is set. This flag is cleared by writing a 1 to it.<br>0b - No swing back cycle has completed since the last time the flag was cleared.<br>1b - At least one swing back cycle has occurred since the last time the flag was cleared. |

*Table continues on the next page...*

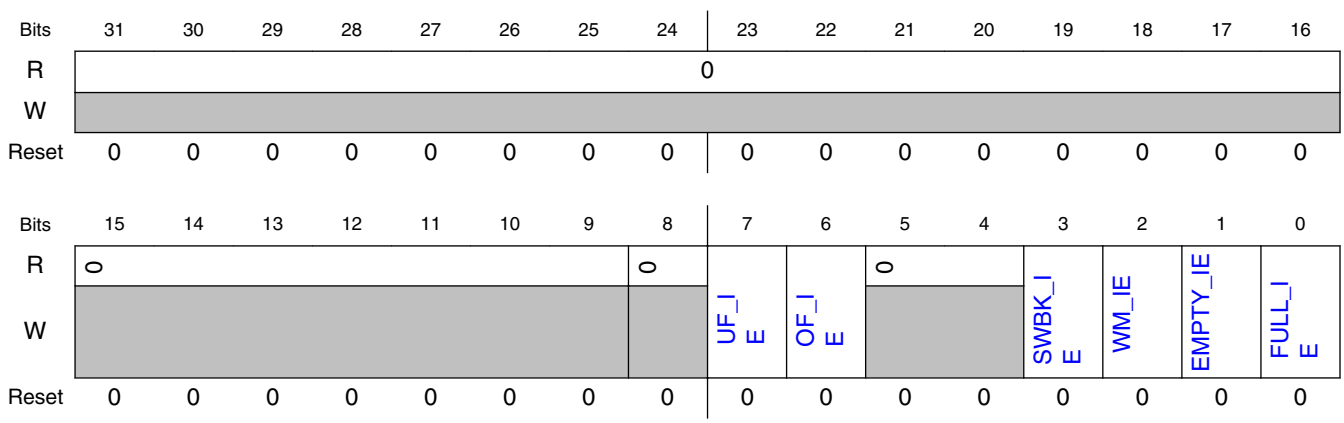
| Field | Function |
|---|---|
| 2<br>WM | FIFO Watermark Status Flag<br>This field is set if the remaining data in FIFO is less than or equal to the FCR[WML]. By writing data into FIFO by DMA or CPU, this flag is cleared automatically when the data in FIFO is more than FCR[WML].<br>0b - Data in FIFO is more than watermark level<br>1b - Data in FIFO is less than or equal to watermark level |
| 1<br>EMPTY | FIFO Empty Flag<br>The EMPTY flag is set when there is no data in FIFO. It is automatically cleared when data is written to the FIFO by CPU or DMA.<br>0b - FIFO is not empty<br>1b - FIFO is empty |
| 0<br>FULL | FIFO Full Flag<br>The FULL flag is set when FIFO is full of data. The FULL flag is automatically cleared by a FIFO read with software trigger or hardware trigger.<br>0b - FIFO is not full<br>1b - FIFO is full |

## 48.5.9 DAC Interrupt Enable Register (IER)

### 48.5.9.1 Offset

| Register | Offset |
|---|---|
| IER | 1Ch |

### 48.5.9.2 Diagram



### 48.5.9.3 Fields

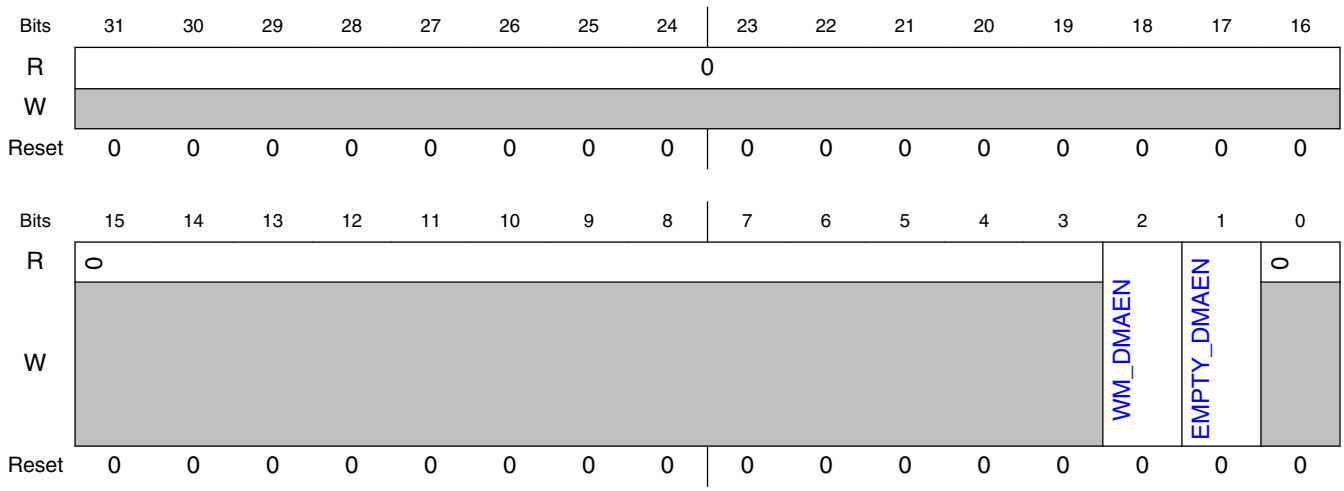
| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>UF_IE | FIFO Underflow Interrupt Enable<br>0b - Underflow interrupt is disabled.<br>1b - Underflow interrupt is enabled. |
| 6<br>OF_IE | FIFO Overflow Interrupt Enable<br>0b - Overflow interrupt is disabled<br>1b - Overflow interrupt is enabled. |
| 5-4<br>— | Reserved |
| 3<br>SWBK_IE | Swing back One Cycle Complete Interrupt Enable<br>0b - Swing back one time complete interrupt is disabled.<br>1b - Swing back one time complete interrupt is enabled. |
| 2<br>WM_IE | FIFO Watermark Interrupt Enable<br>0b - Watermark interrupt is disabled.<br>1b - Watermark interrupt is enabled. |
| 1<br>EMPTY_IE | FIFO Empty Interrupt Enable<br>0b - FIFO Empty interrupt is disabled.<br>1b - FIFO Empty interrupt is enabled. |
| 0<br>FULL_IE | FIFO Full Interrupt Enable<br>0b - FIFO Full interrupt is disabled.<br>1b - FIFO Full interrupt is enabled. |

## 48.5.10 DAC DMA Enable Register (DER)

### 48.5.10.1 Offset

| Register | Offset |
|---|---|
| DER | 20h |

### 48.5.10.2 Diagram




### 48.5.10.3 Fields

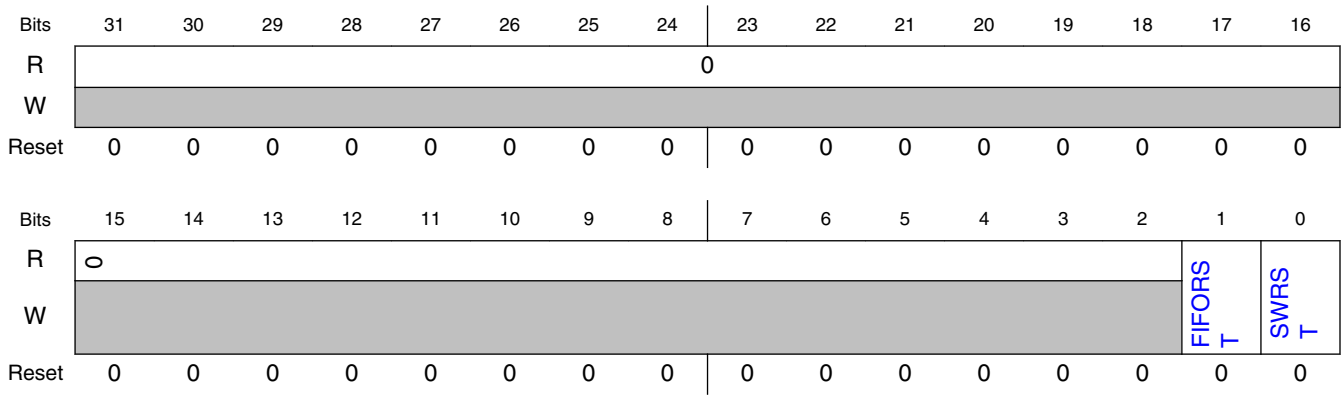
| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>WM_DMAEN | FIFO Watermark DMA Enable<br>0b - Watermark DMA request is disabled.<br>1b - Watermark DMA request is enabled. |
| 1<br>EMPTY_DMAEN | FIFO Empty DMA Enable<br>0b - FIFO Empty DMA request is disabled.<br>1b - FIFO Empty DMA request is enabled. |
| 0<br>— | Reserved |

## 48.5.11 DAC Reset Control Register (RCR)

### 48.5.11.1 Offset

| Register | Offset |
|---|---|
| RCR | 24h |

## 48.5.11.2 Diagram




## 48.5.11.3 Fields

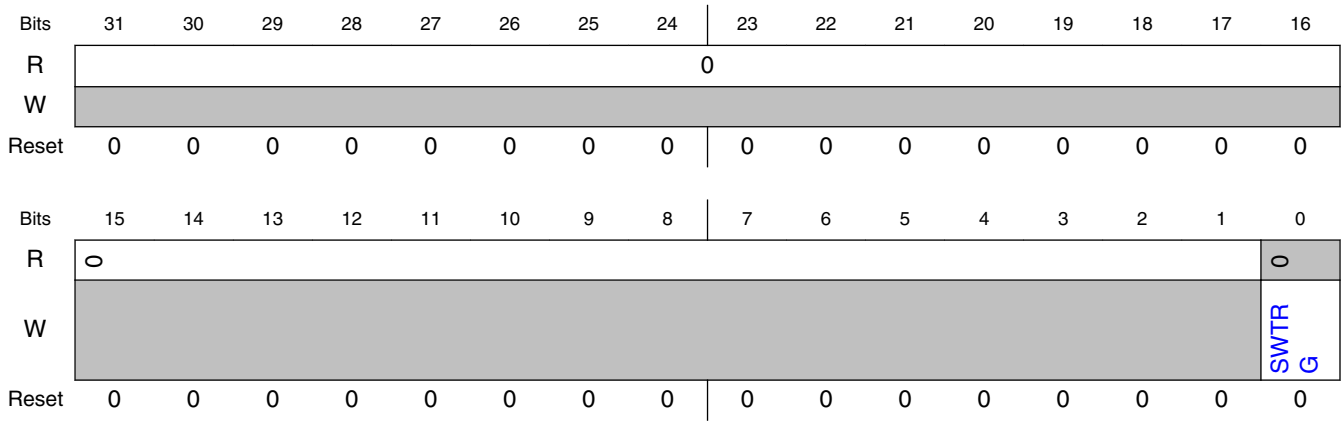
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>FIFORST | FIFO Reset<br>Resets the FIFO pointers and flags in register FSR. Remains set until cleared by software.<br>0b - No effect<br>1b - FIFO reset |
| 0<br>SWRST | Software Reset<br>Resets all DAC registers and internal logic. Remains set until cleared by software.<br>0b - No effect<br>1b - Software reset |

# 48.5.12 DAC Trigger Control Register (TCR)

## 48.5.12.1 Offset

| Register | Offset |
|---|---|
| TCR | 28h |

### 48.5.12.2 Diagram




### 48.5.12.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>SWTRG | Software Trigger<br>Active high. This is a write-only field, which always reads 0. If GCR[TRGSEL] is set and FIFO is enabled and not empty, writing 1 to this field will advance the FIFO read pointer once.<br>**NOTE:** It is not suggested to adopt hardware trigger and software trigger in mixed use, during continuous conversions.<br>0b - The DAC soft trigger is not valid.<br>1b - The DAC soft trigger is valid. |

## 48.6 Functional description

The LPDAC can select one of the two reference inputs — VREFH_INT and VREFH_EXT as the reference voltage, by GCR[DACRFS]. See the chip-specific LPDAC information to determine the source options for VREFH_INT and VREFH_EXT.

When enabled, the LPDAC converts a 12 bit data to a stepped analog output voltage. The output voltage range is from $V_{in}$ to $V_{in}/4096$, and the step is $V_{in}/4096$.

The LPDAC can be configured to FIFO mode, swing back mode or buffer mode.

**NOTE**

User should configure the LPDAC before any conversion. During conversion or FIFO write operation, user should not change the configuration or disable the GCR[DACEN] bit. If need to switch to another mode, user should first reset the LPDAC by writing a 1 to RCR[SWRST] or just clear the GCR[DACEN] bit.

### 48.6.1 Buffer mode

Buffer mode is enabled with GCR[DACEN].

In buffer mode, any write to the DATA[DATA] will push the data to analog conversion without trigger support.

**NOTE**

The data write speed should be not higher than the analog conversion speed.

### 48.6.2 FIFO mode

FIFO mode is enabled with GCR[FIFOEN] and GCR[DACEN].

In FIFO mode, for a 32/16-bit write to DATA[DATA], the data is put into the FIFO, and the write pointer is automatically incremented. For an 8-bit write, only if both the bytes are written, the write pointer will increase. User needs to ensure that an 8-bit write is done in pair, and both the upper and lower bytes are written. There is no priority for which byte is written to first.

When a software trigger or hardware trigger comes, the read pointer pointed data in FIFO will be pushed to analog conversion, and the read pointer is automatically incremented. If the trigger comes when FIFO is empty, the data to analog keeps unchanged and an underflow error flag is set.

### 48.6.3 Swing back mode

Swing back mode is enabled with GCR[FIFOEN], GCR[SWMD] and GCR[DACEN].

In swing back mode, the read pointer swings between the writer pointer and zero. That is, the trigger increases the read pointer till the writer pointer and decreases the read pointer till zero, and so on. The FIFO EMPTY/FULL/WM flags have no meaning and should be ignored during swing back mode.

**NOTE**

In swing back mode, the FIFO should first be written at least two data, otherwise this mode will not work correctly. User should not write the FIFO when in swing back process.

### 48.6.4 Interrupts

When corresponding interrupt enable register bit is set, warning flags and error flags can generate interrupt. The warning flags are EMPTY, FULL, WM, SWBK. The error flags are OF and UF.

### 48.6.5 DMA operation

When corresponding DMA enable register bit is set, DMA requests can be generated automatically when WM or EMPTY flag is set. When DMA is enabled, interrupt can only be generated from the error flags.

### 48.6.6 Low-power mode operation

The LPDAC can be functional with FIFO mode/swing back mode during STOP/WAIT modes, using the hardware trigger initiating the conversion.

# Chapter 49
# Voltage Reference (VREF)

## 49.1 Chip-specific VREF information

### 49.1.1 VREF Overview

This device includes a voltage reference (VREF) to supply an accurate 1.2 V or 2.1V voltage output.

The voltage reference can provide a reference voltage to external peripherals or a reference to analog peripherals, such as the LPADC, LPDAC, or LPCMP.

**NOTE**

SPM_CORELPCNFG[BGEN] bit must be set if the VREF regulator is required to remain operating in VLPx modes.

**NOTE**

For either an internal or external reference if the VREF_OUT functionality is being used, VREF_OUT signal must be connected to an output load capacitor. Refer the device data sheet for more details.

## 49.2 Introduction

The VREF can be used in applications to provide a reference voltage to external devices, or used internally in the device as a reference to analog peripherals (such as the ADC, DAC, or CMP). The Voltage Reference (VREF) can supply an accurate voltage output that can be trimmed in 0.5 mV steps (for 1.2 V output) or 1.5 mV steps (for 2.1 V output). The voltage reference has 3 operating modes that provide different levels of supply rejection and power consumption.

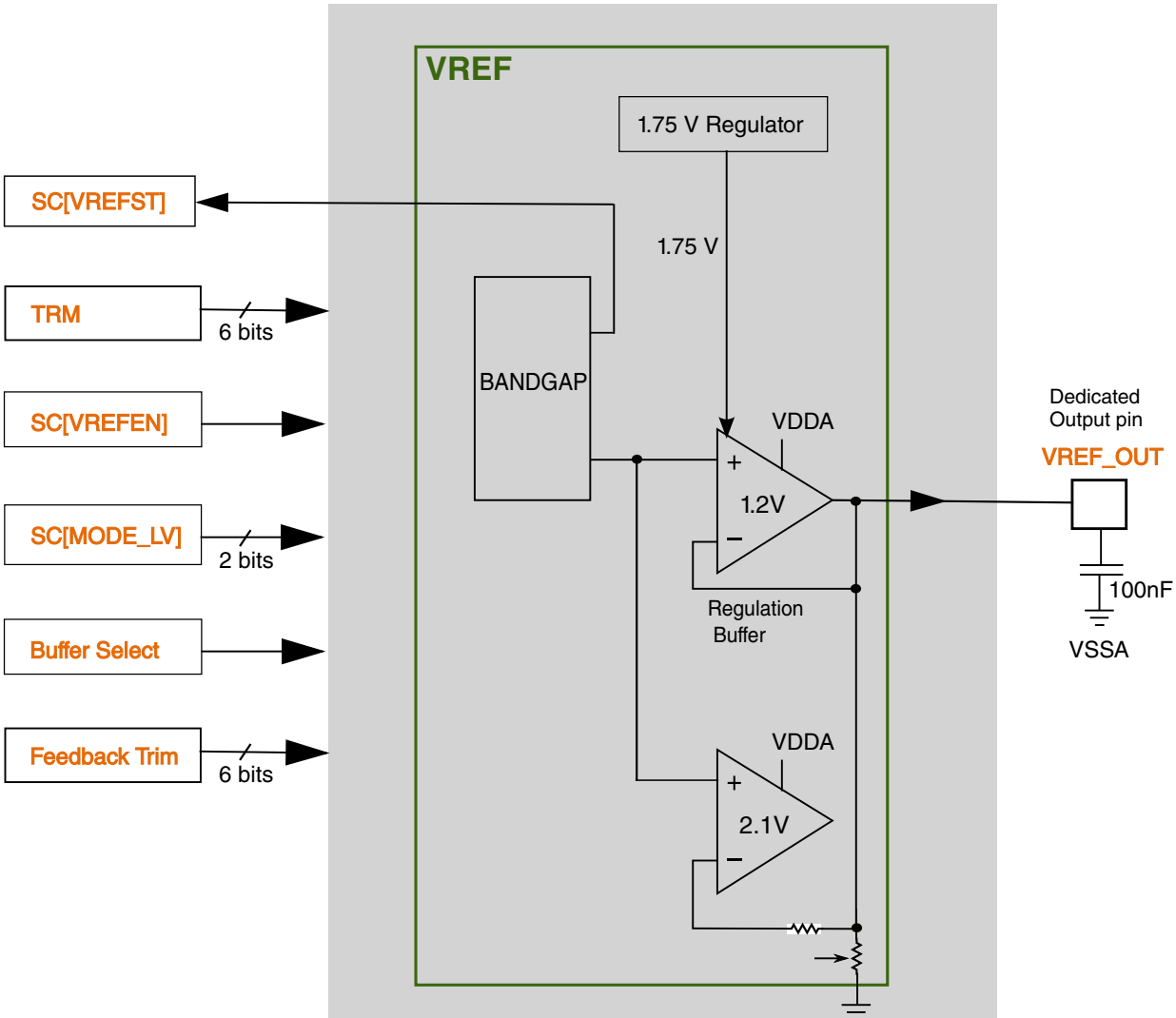



**Figure 49-1. Voltage reference block diagram**

## 49.2.1 Overview

The Voltage Reference provides a buffered reference voltage for use as an external reference, which can be set to either 1.2V or 2.1V. In addition, the buffered reference is available internally for use with on-chip peripherals (such as ADCs and DACs). Refer to the chip configuration details. When the VREF is enabled, the reference voltage is placed on a dedicated output pin.

- In 1.2V mode, the Voltage Reference output can be trimmed with 0.5mV resolution, using the TRM register TRIM[5:0] bitfield.
- In 2.1V mode, the Voltage Reference output can be trimmed with 1.5mV resolution, using the TRM register VREF_TRM4 TRIM2V1[5:0] bitfield.

## 49.2.2 Features

- Programmable trim register with 0.5 mV steps in 1.2V mode and 1.5mV steps in 2.1V mode; upon reset, the trim register is automatically loaded with a factory-trimmed value
- Programmable buffer mode selection:
  - Off
  - Bandgap enabled/standby (output buffer disabled)
  - Low power buffer mode (output buffer enabled)
  - High power buffer mode (output buffer enabled)
- Selectable 1.2V or 2.1V output at room temperature
- Dedicated output pin, VREF_OUT

## 49.2.3 Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, and Stop modes. The Voltage Reference can also run in Very Low Power Run (VLPR), Very Low Power Wait (VLPW) and Very Low Power Stop (VLPS). If it is desired to use the VREF regulator and/or the chop oscillator in the very low power modes, the system reference voltage (also referred to as the bandgap voltage reference) must be enabled in these modes. Refer to the chip configuration details for information on enabling this mode of operation. Having the VREF regulator enabled does increase current consumption. In very low power modes it may be desirable to disable the VREF regulator to minimize current consumption. Note however that the accuracy of the output voltage will be reduced (by as much as several mVs) when the VREF regulator is not used.

**NOTE**

The assignment of module modes to core modes is chip-specific. For module-to-core mode assignments, see the chapter that describes how modules are configured.

### 49.2.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

**Table 49-1. VREF Signal Descriptions**

| Signal | Description | I/O |
|---|---|---|
| VREF_OUT | Internally-generated Voltage Reference output | O |

**NOTE**

When the VREF output buffer is disabled, the status of the VREF_OUT signal is high-impedence.

## 49.3 Memory Map and Register Definition

### 49.3.1 VREF register descriptions

#### 49.3.1.1 VREF Memory map

VREF base address: 4004_D000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | VREF Trim Register (TRM) | 8 | RW | See description. |
| 1h | VREF Status and Control Register (SC) | 8 | RW | 00h |
| 5h | VREF Trim 2.1V Register (TRM4) | 8 | RW | See description. |

#### 49.3.1.2 VREF Trim Register (TRM)

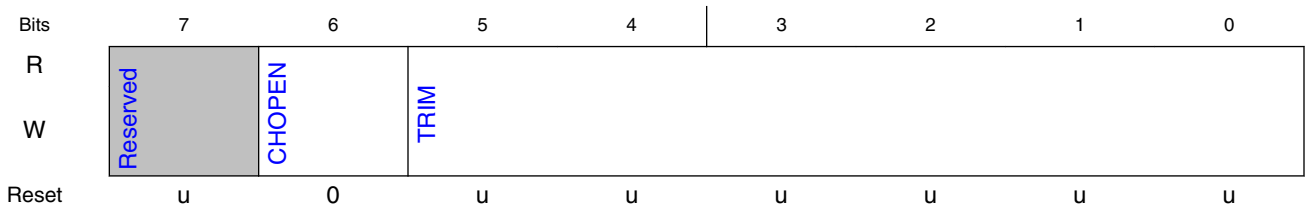
##### 49.3.1.2.1 Offset

| Register | Offset |
|---|---|
| TRM | 0h |

#### 49.3.1.2.2 Function

This register contains the trim data for the Voltage Reference.

#### 49.3.1.2.3 Diagram




#### 49.3.1.2.4 Fields

| Field | Function |
|---|---|
| 7<br>— | Reserved<br>Upon reset, this value is loaded with a factory trim value. |
| 6<br>CHOPEN | Chop oscillator enable. When set, the internal chopping operation is enabled and the internal analog offset will be minimized.<br>CHOPEN bit is set during factory trimming of the VREF voltage. CHOPEN bit should be written to 1 to achieve the performance stated in the data sheet.<br>If the internal voltage regulator is being used (REGEN bit is set to 1), then the chop oscillator must also be enabled.<br>If the chop oscillator will be used in very low power modes, then the system (bandgap) voltage reference must also be enabled. See the chip-specific VREF information (also known as "chip configuration" details) to see how this can be achieved.<br>0b - Chop oscillator is disabled.<br>1b - Chop oscillator is enabled. |
| 5-0<br>TRIM | Trim bits<br>These bits change the resulting VREF by approximately ± 0.5 mV for each step. The trim value is stored in IFR and loaded out of reset.<br>**NOTE:** Min = minimum and Max = maximum voltage reference output. For minimum and maximum voltage reference output values, refer to the Data Sheet of this chip.<br>000000b - Min<br>111111b - Max |

### 49.3.1.3 VREF Status and Control Register (SC)

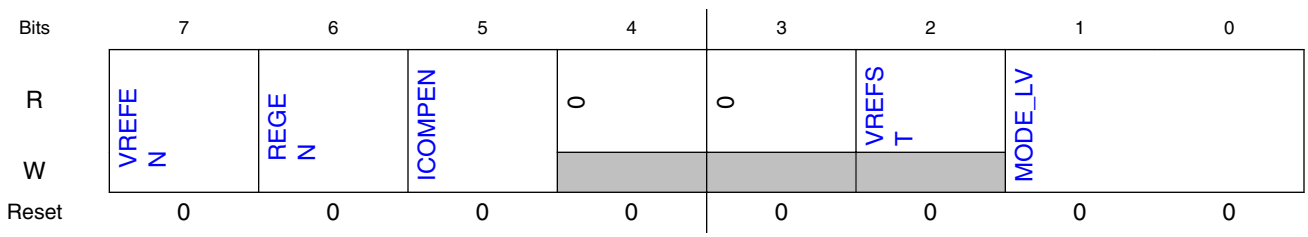
### 49.3.1.3.1 Offset

| Register | Offset |
|---|---|
| SC | 1h |

### 49.3.1.3.2 Function

This register contains the control bits that enable the internal voltage reference and select the buffer mode to be used.

### 49.3.1.3.3 Diagram




### 49.3.1.3.4 Fields

| Field | Function |
|---|---|
| 7<br>VREFEN | Internal Voltage Reference enable<br>Enables the bandgap reference within the Voltage Reference module.<br>**NOTE:** After the VREF is enabled, turning off the clock to the VREF module via the corresponding clock gate register will not disable the VREF. VREF must be disabled using this VREFEN bit.<br>0b - The module is disabled.<br>1b - The module is enabled. |
| 6<br>REGEN | Regulator enable<br>Enables the internal 1.75 V regulator to produce a constant internal voltage supply in order to reduce the sensitivity to external supply noise and variation. If it is desired to keep the regulator enabled in very low power modes, refer to the Chip Configuration details to see how this can be achieved.<br>REGEN bit should be written to 1 to achieve the performance stated in the data sheet.<br>**NOTE:** See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.<br>0b - Internal 1.75 V regulator is disabled.<br>1b - Internal 1.75 V regulator is enabled. |
| 5<br>ICOMPEN | Second order curvature compensation enable<br>ICOMPEN bit should be written to 1 to achieve the performance stated in the data sheet.<br>0b - Disabled<br>1b - Enabled |
| 4 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 3<br>—<br> | Reserved |
| 2<br>VREFST | Internal Voltage Reference stable<br>Indicates that the bandgap reference within the Voltage Reference module has finished its startup and stabilization.<br>**NOTE:** VREFST bit is valid only when the chop oscillator is not being used.<br>0b - The module is disabled or not stable.<br>1b - The module is stable. |
| 1-0<br>MODE_LV | Buffer Mode selection<br>Selects the buffer modes for the Voltage Reference module.<br>00b - Bandgap on only, for stabilization and startup<br>01b - High power buffer mode enabled<br>10b - Low-power buffer mode enabled<br>11b - Reserved |

## 49.3.1.4 VREF Trim 2.1V Register (TRM4)

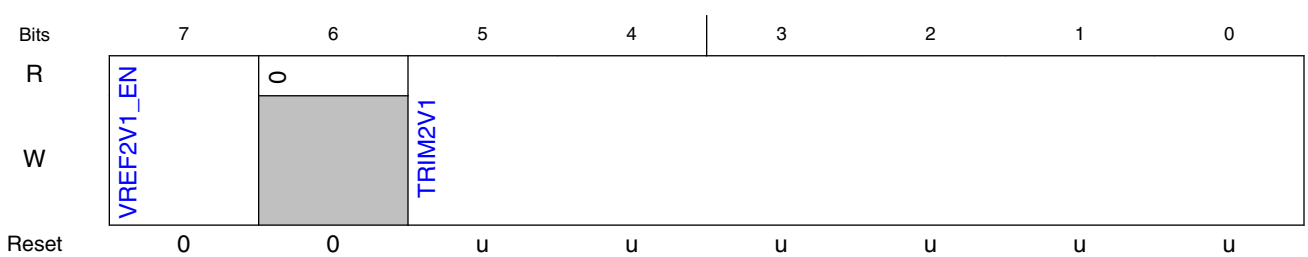
### 49.3.1.4.1 Offset

| Register | Offset |
|---|---|
| TRM4 | 5h |

### 49.3.1.4.2 Function

This register contains the enable and trim for VREF 2.1V generation.

### 49.3.1.4.3 Diagram



#### 49.3.1.4.4 Fields

| Field | Function |
|---|---|
| 7<br>VREF2V1_EN | Internal Voltage Reference (2.1V) Enable<br>Enables the bandgap reference within the Voltage Reference module for generating the 2.1V output<br>0b - VREF 2.1V is enabled<br>1b - VREF 2.1V is disabled |
| 6<br>— | RESERVED |
| 5-0<br>TRIM2V1 | VREF 2.1V Trim Bits<br>These bits change the resulting 2.1 V VREF by approximately ± 1.5 mV for each step. The trim value is stored in IFR and loaded out of reset.<br>**NOTE:** Min = minimum and Max = maximum voltage reference output. For minimum and maximum voltage reference output values, refer to the Data Sheet of this chip.<br>000000b - Max<br>111111b - Min |

## 49.4 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The VREF_OUT signal can be used by both internal and external peripherals in low and high power buffer mode. A 100 nF capacitor must always be connected between VREF_OUT and VSSA if the VREF is being used.

The following table shows all possible function configurations of the Voltage Reference.

**Table 49-2. Voltage Reference function configurations**

| SC[VREFEN] | SC[MODE_LV] | Configuration | Functionality |
|---|---|---|---|
| 0 | X | Voltage Reference disabled | Off |
| 1 | 00 | Voltage Reference enabled, bandgap on only | Startup and standby |
| 1 | 01 | Voltage Reference enabled, high-power buffer on | VREF_OUT available for internal and external use. 100 nF capacitor is required. |
| 1 | 10 | Voltage Reference enabled, low power buffer on | VREF_OUT available for internal and external use. 100 nF capacitor is required. |
| 1 | 11 | Reserved | Reserved |

### 49.4.1 Voltage Reference Disabled, SC[VREFEN] = 0

When SC[VREFEN] = 0, the Voltage Reference is disabled, the VREF bandgap and the output buffers are disabled. The Voltage Reference is in off mode.

### 49.4.2 Voltage Reference Enabled, SC[VREFEN] = 1

When SC[VREFEN] = 1, the Voltage Reference is enabled, and different modes should be set by the SC[MODE_LV] bits.

#### 49.4.2.1 SC[MODE_LV]=00

The internal VREF bandgap is enabled to generate an accurate 1.2 V output that can be trimmed with the TRM register's TRIM[5:0] bitfield. The bandgap requires some time for startup and stabilization. SC[VREFST] can be monitored to determine if the stabilization and startup is complete when the chop oscillator is not enabled.

If the chop oscillator is being used, the internal bandgap reference voltage settles within the chop oscillator start up time, Tchop_osc_stup.

The output buffer is disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode. If this mode is first selected and the low power or high power buffer mode is subsequently enabled, there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet.

#### 49.4.2.2 SC[MODE_LV] = 01

The internal VREF bandgap is on. The high power buffer is enabled to generate a buffered 1.2 V voltage to VREF_OUT. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (SC[MODE_LV] = 00, SC[VREFEN] = 1) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of Tstup or until SC[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

#### 49.4.2.3 SC[MODE_LV] = 10

The internal VREF bandgap is on. The low power buffer is enabled to generate a buffered 1.2 V voltage to VREF_OUT. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (SC[MODE_LV] = 00, SC[VREFEN] = 1) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of Tstup or until SC[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

#### 49.4.2.4 SC[MODE_LV] = 11

Reserved

### 49.4.3 Internal voltage regulator

The VREF module contains an internal voltage regulator that can be enabled to provide additional supply noise rejection. It is recommended that when possible, this regulator be enabled to provide the optimum VREF performance.

If the internal voltage regulator is being used, the chop oscillator must also be enabled. A specific sequence must be followed when enabling the internal regulator as follows:

1. Enable the chop oscillator (VREF_TRM[CHOPEN] = 1)
2. Configure the VREF_SC register to the desired settings with the internal regulator disabled, VREF_SC[REGEN] = 0
3. Wait > 300ns
4. Enable the internal regulator by setting VREF_SC[REGEN] to 1

## 49.5 Initialization/Application Information

The Voltage Reference requires some time for startup and stabilization. After SC[VREFEN] = 1, SC[VREFST] can be monitored to determine if the stabilization and startup is completed when the chop oscillator is not enabled. When the chop oscillator is enabled, the settling time of the internal bandgap reference is defined by Tchop_osc_stup (chop oscillator start up time). You must wait this time (Tchop_osc_stup) after the internal bandgap has been enabled to ensure the VREF internal reference voltage has stabilized.

When the Voltage Reference is already enabled and stabilized, changing SC[MODE_LV] will not clear SC[VREFST] but there will be some startup time before the output voltage at the VREF_OUT pin has settled. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. Also, there will be some settling time when a step change of the load current is applied to the VREF_OUT pin. When the 1.75V VREF regulator is disabled, the VREF_OUT voltage will be more sensitive to supply voltage variation. It is recommended to use this regulator to achieve optimum VREF_OUT performance.

The TRM[CHOPEN], SC[REGEN] and SC[ICOMPEN] bits must be written to 1 to achieve the performance stated in the device data sheet.

**NOTE**

See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.

# Chapter 50
# Low Power Interrupt Timer (LPIT)

## 50.1 Introduction

### 50.1.1 Overview

LPIT is a low power periodic interval timer with multiple timer channels. After a timer reaches a programmed count, the respective timer channel will generate pre-trigger and trigger output signals, and these pre-trigger and trigger outputs can be used to trigger other modules on the device.

- Timers can be configured to be controlled using:
  - external triggers (triggers from outside the LPIT module)
  - or internal triggers (triggers from other timer channels inside the LPIT module).
- The timer channels can be chained together, to form a larger width timer.
- Depending on the timer modes, the timer channels may reload and count again, or stop after reaching the programmed count.

The next figure shows the interface of an LPIT module to the other modules on the SoC.

- The CPU interface provides the clock, reset, register read/write bus interface and handles interrupts from an LPIT.
- The trigger output signals from an LPIT may trigger other modules on the SoC, like the DMA, ADC, and other modules.
- Similarly, other timer modules may provide trigger inputs to an LPIT module, to control when an LPIT timer channel starts.

For the exact module interactions, see the SoC Configuration chapter in your device's hardware Reference Manual (RM).

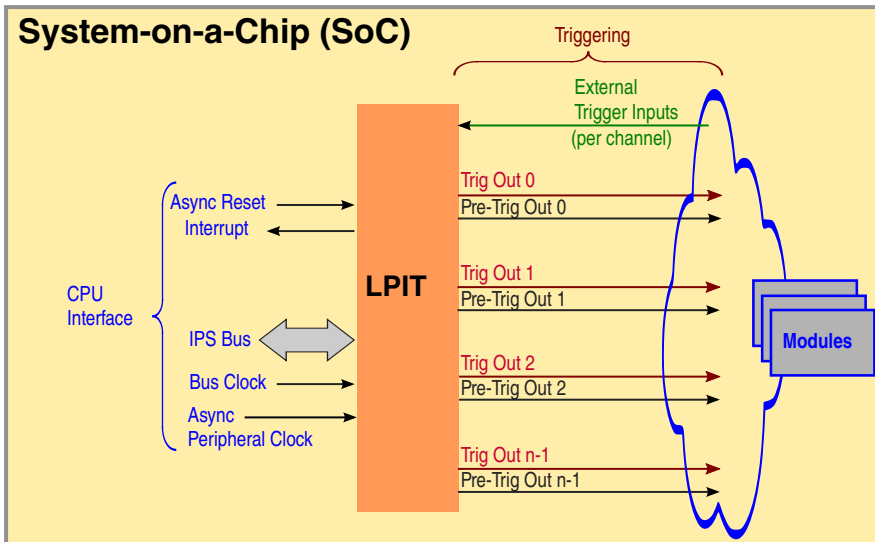



**Figure 50-1. LPIT Interface in SoC**

Each timer channel in LPIT can be configured to work in either compare modes or capture modes.

- **In compare mode:** the timers decrement when enabled and generate an output pre-trigger and trigger output. The trigger output is 1 clock cycle delayed of the pre-trigger pulse. Each timer channel start, reload and restart can be controlled via control bits. The timer can be configured to always decrement from a programmed start value, or decrement on selected trigger inputs or previous channel timeout (when channels are chained). By chaining timer channels, applications can achieve larger timeout durations.
- **In capture mode:** the timer can be used to perform measurements as the timer value is captured (in the timer value register) when a selected trigger input is asserted. The timer can support once-off or multiple measurements (like frequency measurements).

The timer channels operate on an asynchronous clock, which is independent from the register read/write access clock. Clock synchronization between the clock domains ensures normal operations.

## 50.1.2 Block Diagram

The next figure shows a detailed block diagram for an LPIT module. The programming model comprises of a global register set (common to all timer channels), and registers for each timer channel (that control their respective timer channels). Access to these registers gets synchronized to the asynchronous peripheral clock, then affects the timer channel registers.

- Each timer channel contains a 32-bit counter that loads the starting value and down counts on every peripheral clock's positive edge.

- After reaching a zero value (a channel timer timeout), a trigger output is generated.
- The counter enable is controlled using a timer enable register control bit, external or internal triggers or via previous channel timeout (when using timer chaining).
- After a channel timer timeout, an interrupt bit is also set, to tell the CPU about the timer timeout.

For more details, see the functional description section.

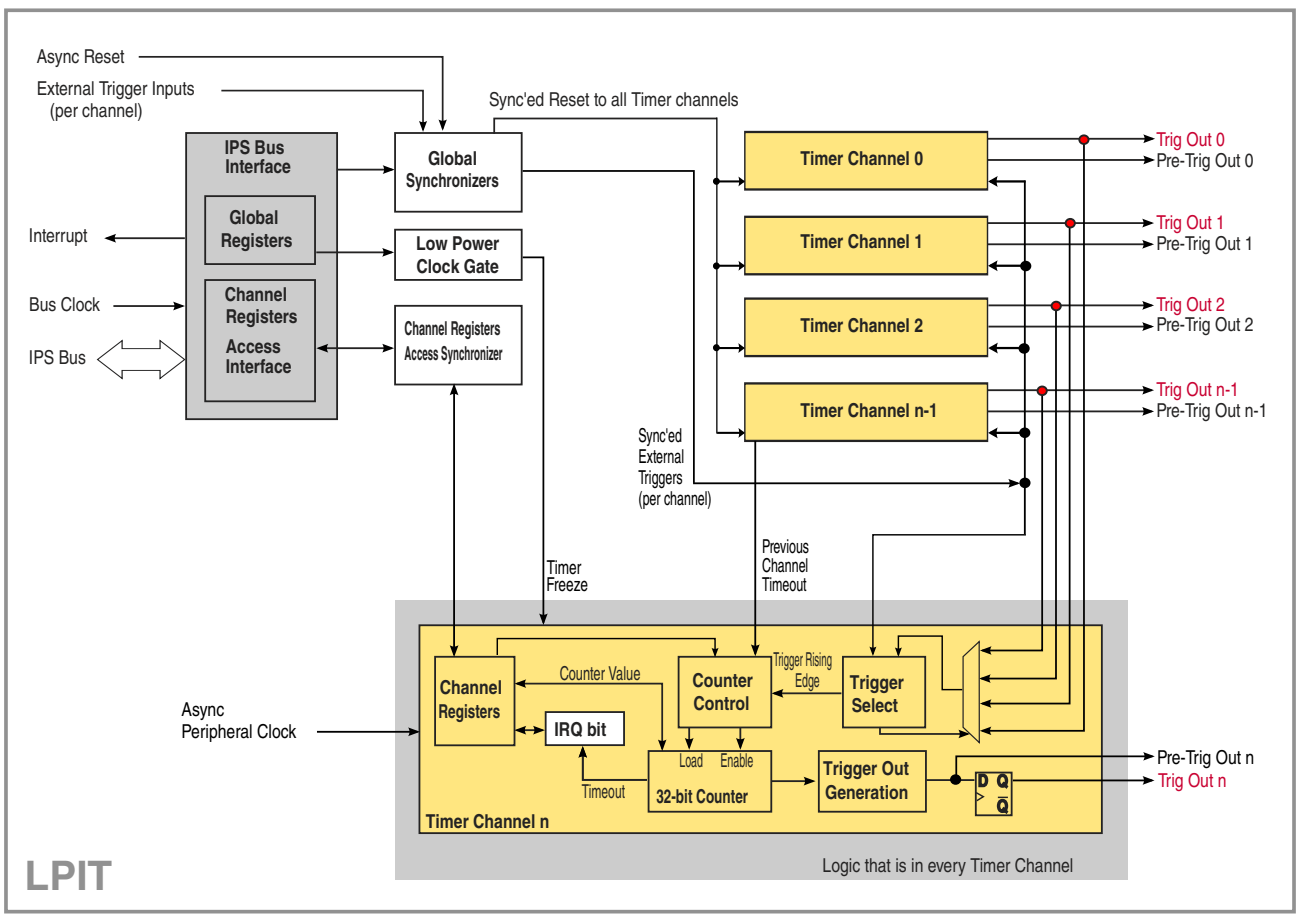



**Figure 50-2. Detailed Block Diagram**

## 50.2 Modes of operation

The LPIT module supports the following chip modes.

**Table 50-1. Chip modes supported by the LPIT module**

| Chip mode | LPIT Operation |
|---|---|
| Run | Normal operation |

*Table continues on the next page...*

**Table 50-1. Chip modes supported by the LPIT module (continued)**

| Chip mode | LPIT Operation |
|---|---|
| Stop/Wait | Can continue operating, if the Doze Enable bit (MCR[DOZE_EN]) is set and the LPIT is using an external or internal clock source that remains operating during stop/wait modes. |
| Low Power Stop (also called Low Leakage Stop or LLS) | The Doze Enable (MCR[DOZE_EN]) bit is ignored and the LPIT will be disabled for the duration of low power mode. |
| Debug | Can continue operating provided the Debug Enable bit (MCR[DBG_EN]) is set. |

# 50.3 Memory Map and Registers

## 50.3.1 LPIT0 register descriptions

The memory map comprises of 32-bit aligned registers, which can be accessed via 8-bit, 16-bit, or 32-bit accesses.

- Write access to reserved locations will generate a transfer error.
- Read access to reserved locations will also generate a transfer error, and the read data bus will show all 0s.

**NOTE**

- The Memory Map and complete module is in Big Endian format.
- The LPIT module will not check if programmed values in the registers are correct; software must ensure that correct programmed values are being written.

### 50.3.1.1 LPIT0 Memory map

LPIT0 base address: 4003_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0404h |
| 8h | Module Control Register (MCR) | 32 | RW | 0000_0000h |
| Ch | Module Status Register (MSR) | 32 | W1C | 0000_0000h |
| 10h | Module Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 14h | Set Timer Enable Register (SETTEN) | 32 | RW | 0000_0000h |
| 18h | Clear Timer Enable Register (CLRTEN) | 32 | WORZ | 0000_0000h |
| 20h | Timer Value Register (TVAL0) | 32 | RW | 0000_0000h |
| 24h | Current Timer Value (CVAL0) | 32 | RO | FFFF_FFFFh |
| 28h | Timer Control Register (TCTRL0) | 32 | RW | 0000_0000h |
| 30h | Timer Value Register (TVAL1) | 32 | RW | 0000_0000h |
| 34h | Current Timer Value (CVAL1) | 32 | RO | FFFF_FFFFh |
| 38h | Timer Control Register (TCTRL1) | 32 | RW | 0000_0000h |
| 40h | Timer Value Register (TVAL2) | 32 | RW | 0000_0000h |
| 44h | Current Timer Value (CVAL2) | 32 | RO | FFFF_FFFFh |
| 48h | Timer Control Register (TCTRL2) | 32 | RW | 0000_0000h |
| 50h | Timer Value Register (TVAL3) | 32 | RW | 0000_0000h |
| 54h | Current Timer Value (CVAL3) | 32 | RO | FFFF_FFFFh |
| 58h | Timer Control Register (TCTRL3) | 32 | RW | 0000_0000h |

## 50.3.1.2 Version ID Register (VERID)

### 50.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 50.3.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 50.3.1.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module design specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module design specification. Read-only field. |
| 15-0<br>FEATURE | Feature Number<br>Returns the feature set number. Read-only field. |

### 50.3.1.3 Parameter Register (PARAM)

#### 50.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 50.3.1.3.2 Function

Provides details on the parameter settings that were used while incorporating this module into the device.

#### 50.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EXT_TRIG | | | | | | | | CHANNEL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 50.3.1.3.4 Fields

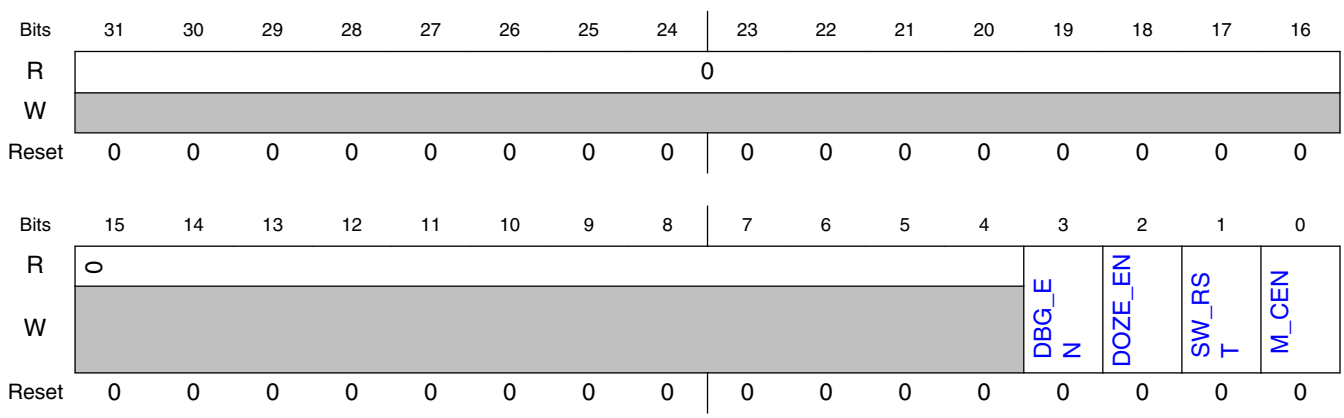
| Field | Function |
|---|---|
| 31-16<br>— | This read-only field is reserved and always has the value 0 |
| 15-8<br>EXT_TRIG | Number of External Trigger Inputs<br>Number of external triggers implemented in this device |
| 7-0<br>CHANNEL | Number of Timer Channels<br>Number of timer channels implemented in this device |

### 50.3.1.4 Module Control Register (MCR)

#### 50.3.1.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 8h |

#### 50.3.1.4.2 Diagram




#### 50.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3 | Debug Enable Bit |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DBG_EN | Stops the timer channels when the device enters Debug mode<br>0b - Stop timer channels in Debug mode<br>1b - Allow timer channels to continue to run in Debug mode |
| 2<br>DOZE_EN | DOZE Mode Enable Bit<br>Stops the timer channels when the device enters DOZE mode<br>0b - Stop timer channels in DOZE mode<br>1b - Allow timer channels to continue to run in DOZE mode |
| 1<br>SW_RST | Software Reset Bit<br>Resets all channels and registers, except the Module Control Register. The Software Reset bit remains set until cleared by software.<br>**NOTE:** Before clearing the Software Reset bit, software must wait for 4 peripheral clocks (for clock synchronization and reset propagation).<br>0b - Timer channels and registers are not reset<br>1b - Reset timer channels and registers |
| 0<br>M_CEN | Module Clock Enable<br>Enables the peripheral clock to the module timers.<br>• The Module Clock Enable bit must be asserted when writing to these registers:<br>  • Module Status Register (MSR)<br>  • Set Timer Enable Register (SETTEN)<br>  • Clear Timer Enable Register (CLRTEN)<br>  • Timer Control Registers (TCTRL*n*)<br>  • Timer Value Registers (TVAL*n*)<br>• To allow for clock synchronization and updating of register bits in the LPIT module, both the bus clock and peripheral clock must be enabled.<br>• Writing to the MSR, SETTEN , CLRTEN, TCTRL, and TVAL registers while the Module Clock Enable bit (M_CEN) = 0, will assert a transfer error for that bus cycle.<br>• Writing to Current Timer Value (CVAL*n*) registers and Reserved registers will always generate a transfer error.<br>**NOTE:** There may be additional clock gating bits in your device that gate the peripheral clock to the LPIT module. Ensure that those additional clock gating bits are configured appropriately, to enable the peripheral clock to the LPIT module.<br>0b - Disable peripheral clock to timers<br>1b - Enable peripheral clock to timers |

## 50.3.1.5 Module Status Register (MSR)

### 50.3.1.5.1 Offset

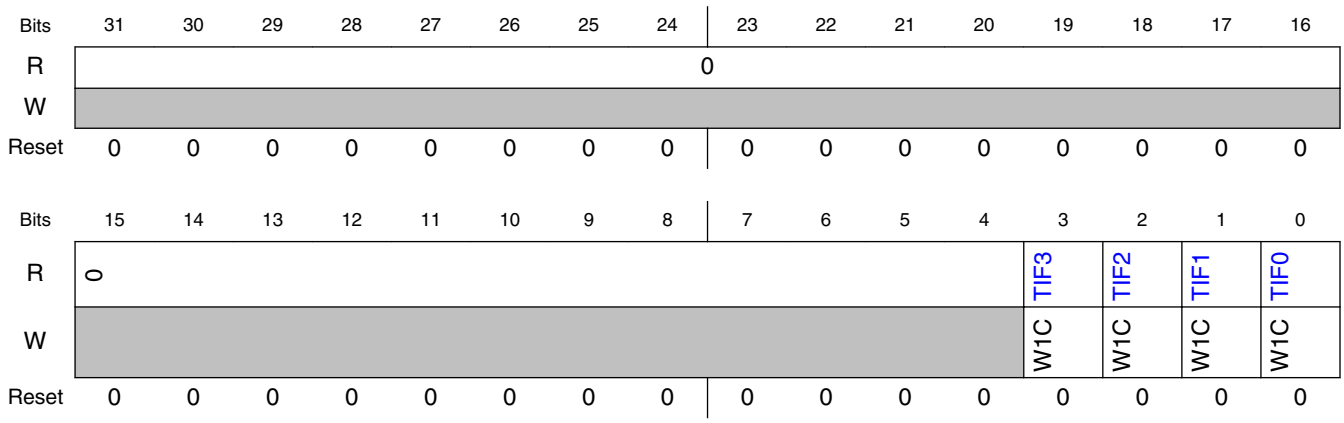
| Register | Offset |
|---|---|
| MSR | Ch |

### 50.3.1.5.2 Function

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Module Status register will generate a transfer error.

### 50.3.1.5.3 Diagram




### 50.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>TIF3 | Channel 3 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br><br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 2<br>TIF2 | Channel 2 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br><br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 1<br>TIF1 | Channel 1 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1 |

*Table continues on the next page...*

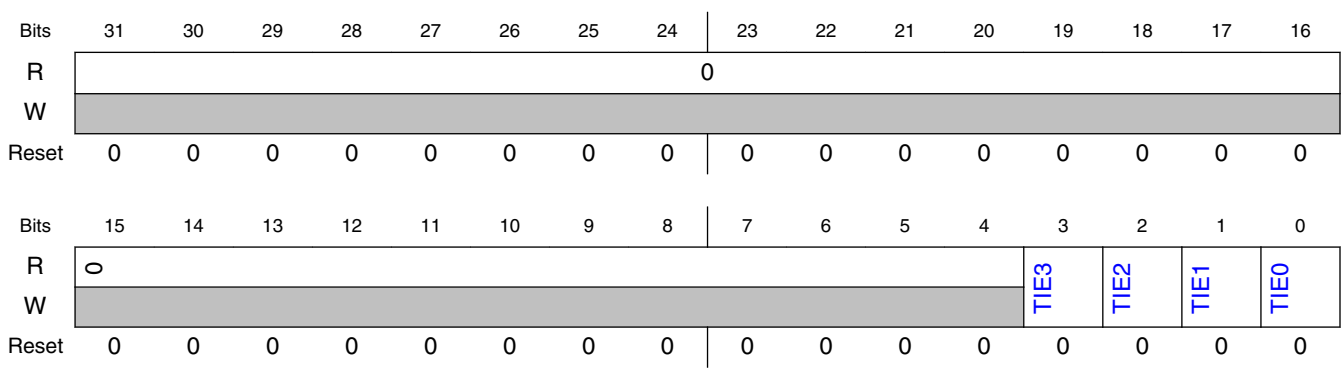
| Field | Function |
|---|---|
| | • To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br><br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 0<br>TIF0 | Channel 0 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br><br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |

## 50.3.1.6 Module Interrupt Enable Register (MIER)

### 50.3.1.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 10h |

### 50.3.1.6.2 Diagram




### 50.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3<br>TIE3 | Channel 3 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 3 Timer Interrupt Enable bit (TIE3) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF3]) is asserted (=1, timeout has occurred)<br><br>0b - Disabled<br>1b - Enabled |
| 2<br>TIE2 | Channel 2 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 2 Timer Interrupt Enable bit (TIE2) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF2]) is asserted (=1, timeout has occurred)<br><br>0b - Disabled<br>1b - Enabled |
| 1<br>TIE1 | Channel 1 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 1 Timer Interrupt Enable bit (TIE1) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF1]) is asserted (=1, timeout has occurred)<br><br>0b - Disabled<br>1b - Enabled |
| 0<br>TIE0 | Channel 0 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 0 Timer Interrupt Enable bit (TIE0) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF0]) is asserted (=1, timeout has occurred)<br><br>0b - Disabled<br>1b - Enabled |

## 50.3.1.7 Set Timer Enable Register (SETTEN)

### 50.3.1.7.1 Offset

| Register | Offset |
|---|---|
| SETTEN | 14h |

### 50.3.1.7.2 Function

The Set Timer Enable register allows the simultaneous enabling of timer channels.

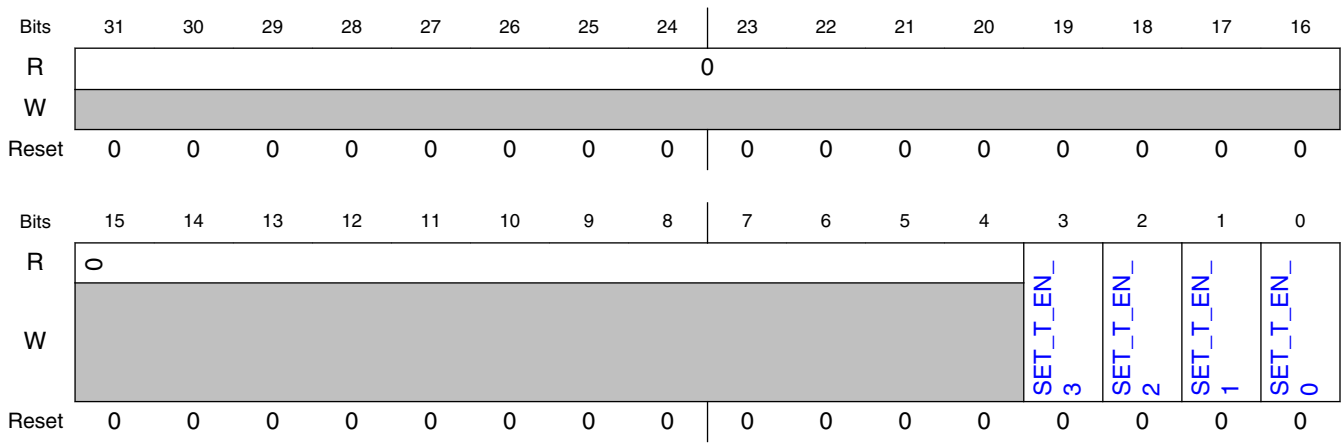
- Timer channels can be enabled by
  - either by writing '1' to Timer Enable bit (T_EN) in the respective TCTRLn register,
  - or by setting the corresponding Set Timer Enable bit (SET_T_EN_*n*) in the Set Timer Enable register (SETTEN).

- To disable timer channels simultaneously, use the Clear Timer Enable register (CLRTEN).
- Writing a '0' to the Set Timer Enable register has no effect.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled)), reading or writing the Set Timer Enable register will generate a transfer error.

### 50.3.1.7.3 Diagram




### 50.3.1.7.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>SET_T_EN_3 | Set Timer 3 Enable<br>To enable timer channel 3, write '1' to Set Timer 3 Enable bit.<br>• The Set Timer 3 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL3 register.<br>• Writing a 0 to Set Timer 3 Enable bit will not disable the counter.<br>• The Set Timer 3 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL3[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 3 Enable bit (CLRTEN[CLR_T_EN_3]).<br><br>0b - No effect<br>1b - Enables Timer Channel 3 |
| 2<br>SET_T_EN_2 | Set Timer 2 Enable<br>To enable timer channel 2, write '1' to Set Timer 2 Enable bit.<br>• The Set Timer 2 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL2 register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • Writing a 0 to Set Timer 2 Enable bit will not disable the counter.<br>• The Set Timer 2 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL2[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 2 Enable bit (CLRTEN[CLR_T_EN_2]).<br><br>0b - No Effect<br>1b - Enables Timer Channel 2 |
| 1<br><br>SET_T_EN_1 | Set Timer 1 Enable<br><br>To enable timer channel 1, write '1' to Set Timer 1 Enable bit.<br>• The Set Timer 1 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL1 register.<br>• Writing a 0 to Set Timer 1 Enable bit will not disable the counter.<br>• The Set Timer 1 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL1[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 1 Enable bit (CLRTEN[CLR_T_EN_1]).<br><br>0b - No Effect<br>1b - Enables Timer Channel 1 |
| 0<br><br>SET_T_EN_0 | Set Timer 0 Enable<br><br>To enable timer channel 0, write '1' to Set Timer 0 Enable bit.<br>• The Set Timer 0 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL0 register.<br>• Writing a 0 to Set Timer 0 Enable bit will not disable the counter.<br>• The Set Timer 0 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL0[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 0 Enable bit (CLRTEN[CLR_T_EN_0]).<br><br>0b - No effect<br>1b - Enables Timer Channel 0 |

## 50.3.1.8 Clear Timer Enable Register (CLRTEN)

### 50.3.1.8.1 Offset

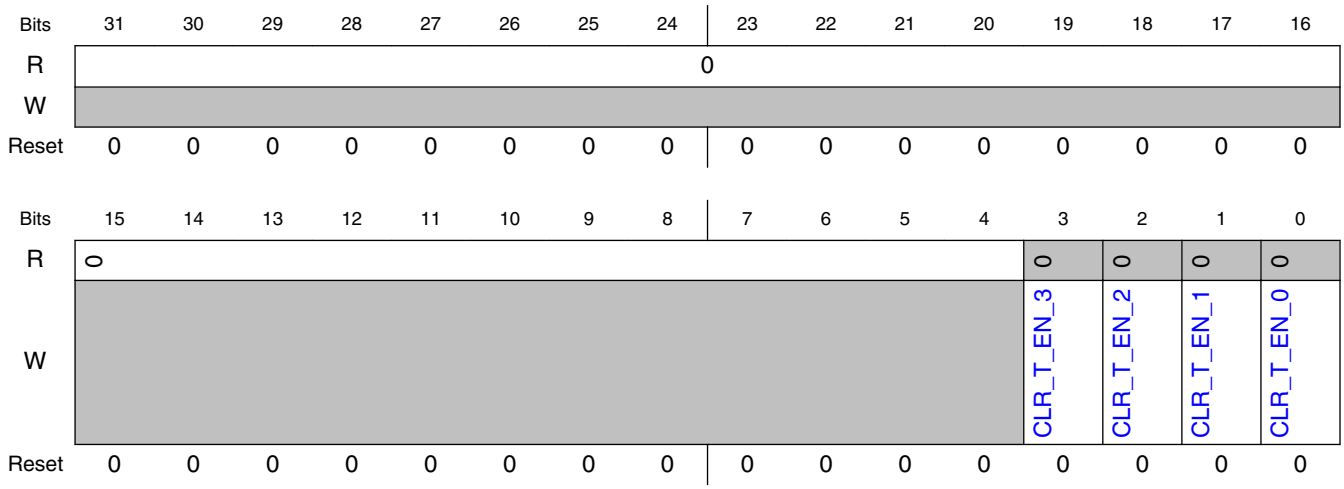
| Register | Offset |
|---|---|
| CLRTEN | 18h |

### 50.3.1.8.2 Function

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Clear Timer Enable register will generate a transfer error.

### 50.3.1.8.3 Diagram




### 50.3.1.8.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>CLR_T_EN_3 | Clear Timer 3 Enable<br>• To disable timer channel 3, write '1' to Clear Timer 3 Enable bit.<br>• The Clear Timer 3 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL3 register.<br>• Writing a 1 to Clear Timer 3 Enable bit will not enable the counter.<br>• The Clear Timer 3 Enable bit is self-clearing, which means that the Clear Timer 3 Enable bit will always read 0.<br><br>0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL3[T_EN]) for Timer Channel 3 |
| 2<br>CLR_T_EN_2 | Clear Timer 2 Enable<br>• To disable timer channel 2, write '1' to Clear Timer 2 Enable bit.<br>• The Clear Timer 2 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL2 register.<br>• Writing a 1 to Clear Timer 2 Enable bit will not enable the counter.<br>• The Clear Timer 2 Enable bit is self-clearing, which means that the Clear Timer 2 Enable bit will always read 0.<br><br>0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL2[T_EN]) for Timer Channel 2 |
| 1<br>CLR_T_EN_1 | Clear Timer 1 Enable<br>• To disable timer channel 1, write '1' to Clear Timer 1 Enable bit.<br>• The Clear Timer 1 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL1 register.<br>• Writing a 1 to Clear Timer 1 Enable bit will not enable the counter.<br>• The Clear Timer 1 Enable bit is self-clearing, which means that the Clear Timer 1 Enable bit will always read 0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL1[T_EN]) for Timer Channel 1 |
| 0<br>CLR_T_EN_0 | Clear Timer 0 Enable<br>• To disable timer channel 0, write '1' to Clear Timer 0 Enable bit.<br>• The Clear Timer 0 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL0 register.<br>• Writing a 1 to Clear Timer 0 Enable bit will not enable the counter.<br>• The Clear Timer 0 Enable bit is self-clearing, which means that the Clear Timer 0 Enable bit will always read 0.<br><br>0b - No action<br>1b - Clear the Timer Enable bit (TCTRL0[T_EN]) for Timer Channel 0 |

## 50.3.1.9 Timer Value Register (TVAL0 - TVAL3)

### 50.3.1.9.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TVALn | 20h + (n × 10h) |

### 50.3.1.9.2 Function

- In compare modes: the Timer Value Registers (TVAL*n*) select the timeout period for the timer channels.
- In capture modes: the Timer Value Registers (TVAL*n*) are loaded with the value of the counter when the trigger asserts.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), then reading or writing the TVALn register will generate a transfer error.

### 50.3.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 50.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>TMR_VAL | Timer Value<br>• In compare mode: Timer Value (TMR_VAL) is the timer channel start value.<br>  • The timer will count down until the timer reaches 0, then the timer will generate an interrupt and load the Timer Value register (TVAL*n*) value again.<br>  • Writing a new value to the Timer Value register (TVAL*n*) will not restart the timer channel; instead the new value will be loaded *after the timer expires*.<br>  • To abort the current timer cycle and start a timer period with a new value, the timer channel must be disabled and enabled *again*.<br>• In capture mode: whenever the trigger asserts, the Timer Value register stores the inverse of the counter value.<br><br>00000000000000000000000000000000b - Invalid load value in compare mode<br>00000000000000000000000000000001b - Invalid load value in compare mode<br>00000000000000000000000000000010-11111111111111111111111111111111b - In compare mode: the value to be loaded; in capture mode, the value of the timer |

## 50.3.1.10 Current Timer Value (CVAL0 - CVAL3)

### 50.3.1.10.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| CVALn | 24h + (n × 10h) |

### 50.3.1.10.2 Function

The Current Timer Value registers indicate the current timer counter value.

#### 50.3.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 50.3.1.10.4 Fields

| Field | Function |
|---|---|
| 31-0<br>TMR_CUR_VAL | Current Timer Value<br>Represents the current timer value, if the timer is enabled |

### 50.3.1.11 Timer Control Register (TCTRL0 - TCTRL3)

#### 50.3.1.11.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TCTRLn | 28h + (n × 10h) |

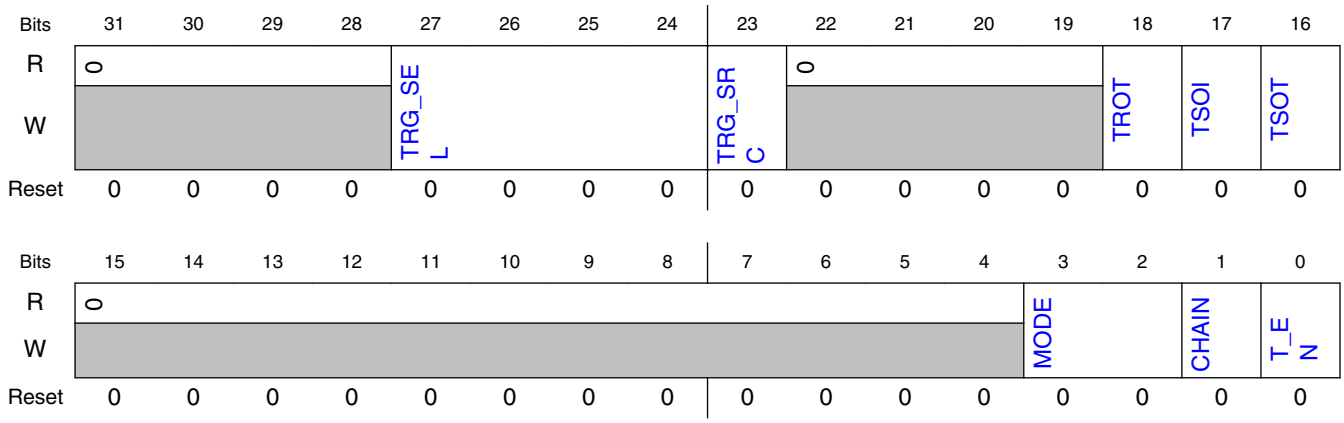
#### 50.3.1.11.2 Function

The Timer Control registers contain the control bits for each timer channel.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Timer Control register will generate a transfer error.

## 50.3.1.11.3 Diagram




## 50.3.1.11.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | This read-only field is reserved and always has the value 0 |
| 27-24<br>TRG_SEL | Trigger Select<br>Selects the trigger to use for starting and/or reloading the LPIT timer.<br>• The TRG_SEL field selects one trigger from the set of internal or external triggers that are selected by the Trigger Source bit (TRG_SRC)<br>• Recall that the TRG_SRC bit selects between internal and external trigger signals for each channel<br>**NOTE:** The Trigger Select field should only be changed when the LPIT timer channel is disabled.<br>0000-0011b - Timer channel 0 - 3 trigger source is selected<br>0100-1111b - Reserved |
| 23<br>TRG_SRC | Trigger Source<br>Selects between internal or external trigger sources. The trigger to be used is selected using the TRG_SRC and TRG_SEL bits.<br>Refer to the chip configuration section for available external trigger options. If a channel does not have an associated external trigger, then set the Trigger Source bit (TRG_SRC) = 1.<br>0b - Selects external triggers<br>1b - Selects internal triggers |
| 22-19<br>— | This read-only field is reserved and always has the value 0 |
| 18<br>TROT | Timer Reload On Trigger<br>When set, the LPIT timer will reload when a rising edge is detected on the selected trigger input. The trigger input is ignored if the LPIT is disabled during debug mode (DBGEN = 0) or DOZE mode (DOZE_EN = 0)<br>0b - Timer will not reload on the selected trigger<br>1b - Timer will reload on the selected trigger |
| 17<br>TSOI | Timer Stop On Interrupt<br>Controls whether the channel timer will stop after it (the channel timer) times out or not.<br>0b - The channel timer does not stop after timeout |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - The channel timer will stop after a timeout, and the channel timer will restart based on Timer Start On Trigger bit (TSOT). When TSOT = 0, the channel timer will restart after a rising edge on the Timer Enable bit (T_EN) is detected (which means that the timer channel is disabled and then enabled). When TSOT = 1, the channel timer will restart after a rising edge on the selected trigger is detected. |
| 16<br>TSOT | Timer Start On Trigger<br>Controls when the timer starts decrementing.<br>0b - Timer starts to decrement immediately based on the restart condition (controlled by the Timer Stop On Interrupt bit (TSOI))<br>1b - Timer starts to decrement when a rising edge on a selected trigger is detected |
| 15-4<br>— | This read-only field is reserved and always has the value 0 |
| 3-2<br>MODE | Timer Operation Mode<br>Configures the channel timer's mode of operation. The MODE bits control how the timer decrements.<br>00b - 32-bit Periodic Counter<br>01b - Dual 16-bit Periodic Counter<br>10b - 32-bit Trigger Accumulator<br>11b - 32-bit Trigger Input Capture |
| 1<br>CHAIN | Chain Channel<br>When enabled, the timer channel will decrement when timer channel N-1 trigger asserts. Timer channel 0 cannot be chained.<br>0b - Channel Chaining is disabled. The channel timer runs independently.<br>1b - Channel Chaining is enabled. The timer decrements on the previous channel's timeout. |
| 0<br>T_EN | Timer Enable<br>Enables or disables the Timer Channel<br>0b - Timer Channel is disabled<br>1b - Timer Channel is enabled |

## 50.3.2 LPIT1 register descriptions

The memory map comprises of 32-bit aligned registers, which can be accessed via 8-bit, 16-bit, or 32-bit accesses.

- Write access to reserved locations will generate a transfer error.
- Read access to reserved locations will also generate a transfer error, and the read data bus will show all 0s.

**NOTE**

- The Memory Map and complete module is in Big Endian format.
- The LPIT module will not check if programmed values in the registers are correct; software must ensure that correct programmed values are being written.

### 50.3.2.1 LPIT1 Memory map

LPIT1 base address: 4102_A000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0100_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0404h |
| 8h | Module Control Register (MCR) | 32 | RW | 0000_0000h |
| Ch | Module Status Register (MSR) | 32 | W1C | 0000_0000h |
| 10h | Module Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |
| 14h | Set Timer Enable Register (SETTEN) | 32 | RW | 0000_0000h |
| 18h | Clear Timer Enable Register (CLRTEN) | 32 | WORZ | 0000_0000h |
| 20h | Timer Value Register (TVAL0) | 32 | RW | 0000_0000h |
| 24h | Current Timer Value (CVAL0) | 32 | RO | FFFF_FFFFh |
| 28h | Timer Control Register (TCTRL0) | 32 | RW | 0000_0000h |
| 30h | Timer Value Register (TVAL1) | 32 | RW | 0000_0000h |
| 34h | Current Timer Value (CVAL1) | 32 | RO | FFFF_FFFFh |
| 38h | Timer Control Register (TCTRL1) | 32 | RW | 0000_0000h |
| 40h | Timer Value Register (TVAL2) | 32 | RW | 0000_0000h |
| 44h | Current Timer Value (CVAL2) | 32 | RO | FFFF_FFFFh |
| 48h | Timer Control Register (TCTRL2) | 32 | RW | 0000_0000h |
| 50h | Timer Value Register (TVAL3) | 32 | RW | 0000_0000h |
| 54h | Current Timer Value (CVAL3) | 32 | RO | FFFF_FFFFh |
| 58h | Timer Control Register (TCTRL3) | 32 | RW | 0000_0000h |

### 50.3.2.2 Version ID Register (VERID)

#### 50.3.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 50.3.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 50.3.2.2.3 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | Returns the major version number for the module design specification. Read-only field. |
| 23-16 | Minor Version Number |
| MINOR | Returns the minor version number for the module design specification. Read-only field. |
| 15-0 | Feature Number |
| FEATURE | Returns the feature set number. Read-only field. |

### 50.3.2.3 Parameter Register (PARAM)

#### 50.3.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 50.3.2.3.2 Function

Provides details on the parameter settings that were used while incorporating this module into the device.

### 50.3.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EXT_TRIG | | | | | | | | CHANNEL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### 50.3.2.3.4 Fields

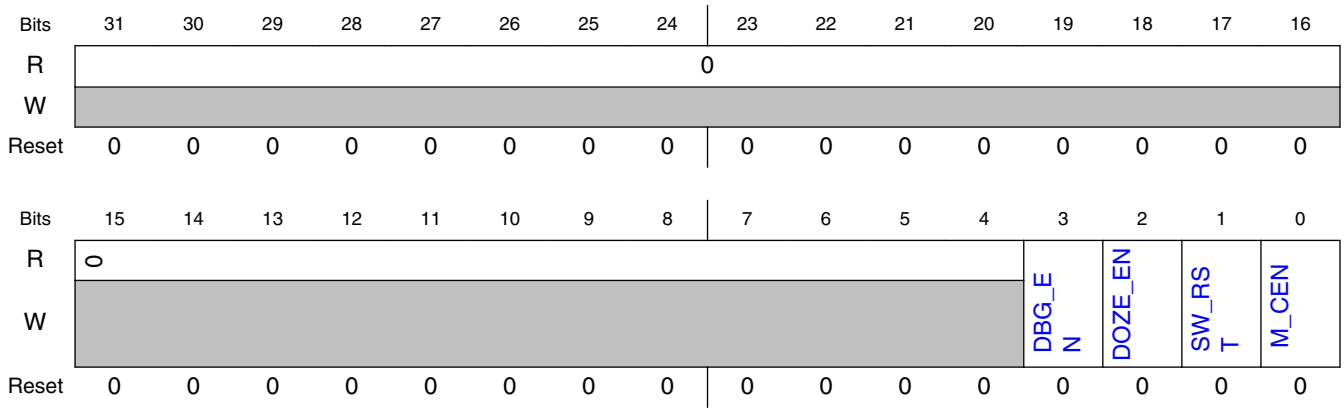
| Field | Function |
|---|---|
| 31-16<br>— | This read-only field is reserved and always has the value 0 |
| 15-8<br>EXT_TRIG | Number of External Trigger Inputs<br>Number of external triggers implemented in this device |
| 7-0<br>CHANNEL | Number of Timer Channels<br>Number of timer channels implemented in this device |

## 50.3.2.4 Module Control Register (MCR)

### 50.3.2.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 8h |

### 50.3.2.4.2 Diagram




### 50.3.2.4.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>DBG_EN | Debug Enable Bit<br>Stops the timer channels when the device enters Debug mode<br>0b - Stop timer channels in Debug mode<br>1b - Allow timer channels to continue to run in Debug mode |
| 2<br>DOZE_EN | DOZE Mode Enable Bit<br>Stops the timer channels when the device enters DOZE mode<br>0b - Stop timer channels in DOZE mode<br>1b - Allow timer channels to continue to run in DOZE mode |
| 1<br>SW_RST | Software Reset Bit<br>Resets all channels and registers, except the Module Control Register. The Software Reset bit remains set until cleared by software.<br>**NOTE:** Before clearing the Software Reset bit, software must wait for 4 peripheral clocks (for clock synchronization and reset propagation).<br>0b - Timer channels and registers are not reset<br>1b - Reset timer channels and registers |
| 0<br>M_CEN | Module Clock Enable<br>Enables the peripheral clock to the module timers.<br>• The Module Clock Enable bit must be asserted when writing to these registers:<br>  • Module Status Register (MSR)<br>  • Set Timer Enable Register (SETTEN)<br>  • Clear Timer Enable Register (CLRTEN)<br>  • Timer Control Registers (TCTRL*n*)<br>  • Timer Value Registers (TVAL*n*)<br>• To allow for clock synchronization and updating of register bits in the LPIT module, both the bus clock and peripheral clock must be enabled. |

| Field | Function |
|---|---|
| | • Writing to the MSR, SETTEN , CLRTEN, TCTRL, and TVAL registers while the Module Clock Enable bit (M_CEN) = 0, will assert a transfer error for that bus cycle.<br>• Writing to Current Timer Value (CVAL*n*) registers and Reserved registers will always generate a transfer error.<br><br>**NOTE:** There may be additional clock gating bits in your device that gate the peripheral clock to the LPIT module. Ensure that those additional clock gating bits are configured appropriately, to enable the peripheral clock to the LPIT module.<br><br>0b - Disable peripheral clock to timers<br>1b - Enable peripheral clock to timers |

## 50.3.2.5 Module Status Register (MSR)

### 50.3.2.5.1 Offset

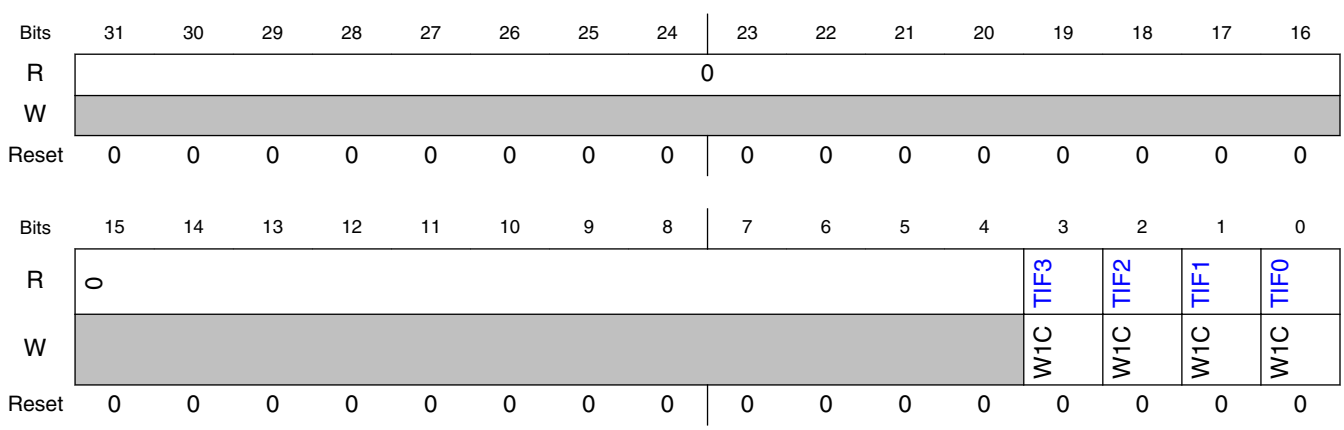
| Register | Offset |
|---|---|
| MSR | Ch |

### 50.3.2.5.2 Function

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Module Status register will generate a transfer error.

### 50.3.2.5.3 Diagram



#### 50.3.2.5.4 Fields

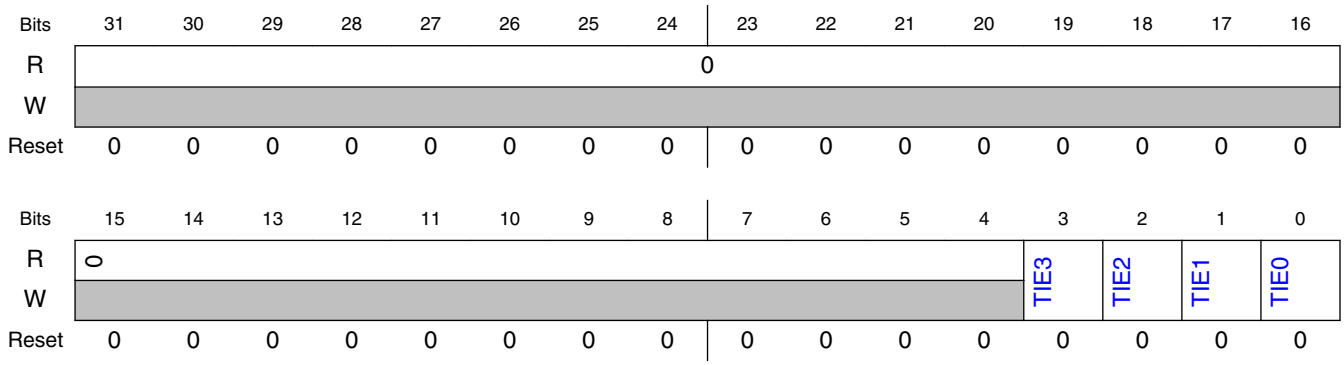
| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>TIF3 | Channel 3 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 2<br>TIF2 | Channel 2 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 1<br>TIF1 | Channel 1 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |
| 0<br>TIF0 | Channel 0 Timer Interrupt Flag<br>• In compare modes: at the end of the timer period, the channel timer interrupt flag is set to 1<br>• In capture modes: when the trigger asserts, the channel timer interrupt flag is set to 1<br>• To clear a channel timer interrupt flag, write logic 1 to it<br>• Writing 0 to a channel timer interrupt flag has no effect<br>0b - Timer has not timed out<br>1b - Timeout has occurred (timer has timed out) |

### 50.3.2.6 Module Interrupt Enable Register (MIER)

#### 50.3.2.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 10h |

## 50.3.2.6.2 Diagram




## 50.3.2.6.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>TIE3 | Channel 3 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 3 Timer Interrupt Enable bit (TIE3) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF3]) is asserted (=1, timeout has occurred)<br>0b - Disabled<br>1b - Enabled |
| 2<br>TIE2 | Channel 2 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 2 Timer Interrupt Enable bit (TIE2) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF2]) is asserted (=1, timeout has occurred)<br>0b - Disabled<br>1b - Enabled |
| 1<br>TIE1 | Channel 1 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 1 Timer Interrupt Enable bit (TIE1) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF1]) is asserted (=1, timeout has occurred)<br>0b - Disabled<br>1b - Enabled |
| 0<br>TIE0 | Channel 0 Timer Interrupt Enable<br>Enables interrupt generation when:<br>• the Channel 0 Timer Interrupt Enable bit (TIE0) is set to 1<br>• and if the corresponding Timer Interrupt Flag (MSR[TIF0]) is asserted (=1, timeout has occurred)<br>0b - Disabled<br>1b - Enabled |

## 50.3.2.7 Set Timer Enable Register (SETTEN)

### 50.3.2.7.1 Offset

| Register | Offset |
|---|---|
| SETTEN | 14h |

### 50.3.2.7.2 Function

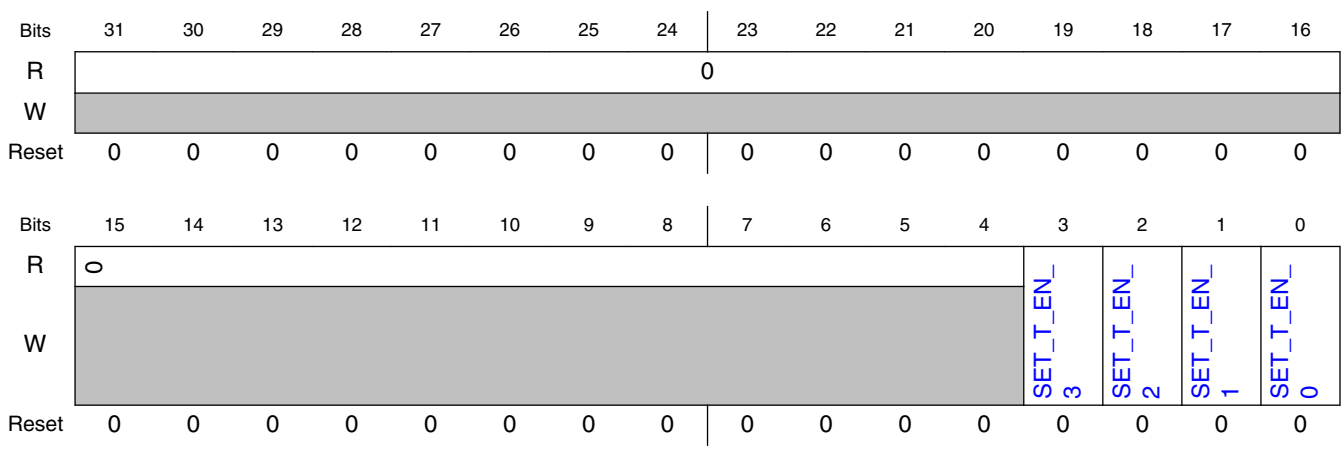
The Set Timer Enable register allows the simultaneous enabling of timer channels.

- Timer channels can be enabled by
  - either by writing '1' to Timer Enable bit (T_EN) in the respective TCTRLn register,
  - or by setting the corresponding Set Timer Enable bit (SET_T_EN_*n*) in the Set Timer Enable register (SETTEN).
- To disable timer channels simultaneously, use the Clear Timer Enable register (CLRTEN).
- Writing a '0' to the Set Timer Enable register has no effect.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled)), reading or writing the Set Timer Enable register will generate a transfer error.

### 50.3.2.7.3 Diagram



## 50.3.2.7.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>SET_T_EN_3 | Set Timer 3 Enable<br>To enable timer channel 3, write '1' to Set Timer 3 Enable bit.<br>• The Set Timer 3 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL3 register.<br>• Writing a 0 to Set Timer 3 Enable bit will not disable the counter.<br>• The Set Timer 3 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL3[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 3 Enable bit (CLRTEN[CLR_T_EN_3]).<br><br>0b - No effect<br>1b - Enables Timer Channel 3 |
| 2<br>SET_T_EN_2 | Set Timer 2 Enable<br>To enable timer channel 2, write '1' to Set Timer 2 Enable bit.<br>• The Set Timer 2 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL2 register.<br>• Writing a 0 to Set Timer 2 Enable bit will not disable the counter.<br>• The Set Timer 2 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL2[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 2 Enable bit (CLRTEN[CLR_T_EN_2]).<br><br>0b - No Effect<br>1b - Enables Timer Channel 2 |
| 1<br>SET_T_EN_1 | Set Timer 1 Enable<br>To enable timer channel 1, write '1' to Set Timer 1 Enable bit.<br>• The Set Timer 1 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL1 register.<br>• Writing a 0 to Set Timer 1 Enable bit will not disable the counter.<br>• The Set Timer 1 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL1[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 1 Enable bit (CLRTEN[CLR_T_EN_1]).<br><br>0b - No Effect<br>1b - Enables Timer Channel 1 |
| 0<br>SET_T_EN_0 | Set Timer 0 Enable<br>To enable timer channel 0, write '1' to Set Timer 0 Enable bit.<br>• The Set Timer 0 Enable bit can be used in addition to the Timer Enable bit (T_EN) in TCTRL0 register.<br>• Writing a 0 to Set Timer 0 Enable bit will not disable the counter.<br>• The Set Timer 0 Enable bit will be cleared<br>  • when Timer Enable bit (TCTRL0[T_EN]) is set to 0<br>  • or when '1' is written to the Clear Timer 0 Enable bit (CLRTEN[CLR_T_EN_0]).<br><br>0b - No effect<br>1b - Enables Timer Channel 0 |

# 50.3.2.8 Clear Timer Enable Register (CLRTEN)

## 50.3.2.8.1 Offset

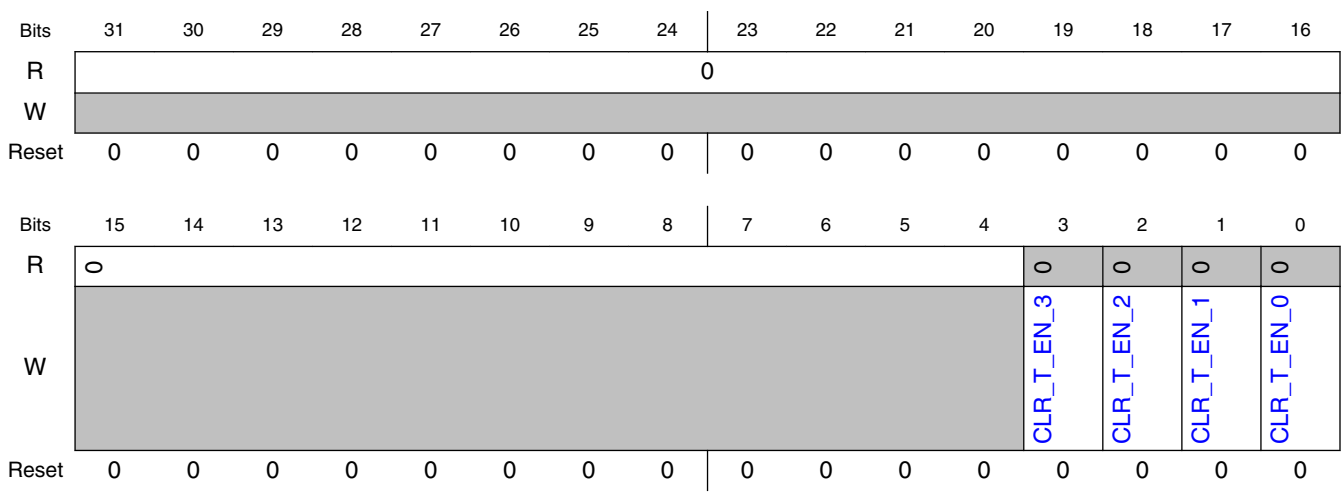
| Register | Offset |
|---|---|
| CLRTEN | 18h |

## 50.3.2.8.2 Function

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Clear Timer Enable register will generate a transfer error.

## 50.3.2.8.3 Diagram




## 50.3.2.8.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | This read-only field is reserved and always has the value 0 |
| 3<br>CLR_T_EN_3 | Clear Timer 3 Enable<br>• To disable timer channel 3, write '1' to Clear Timer 3 Enable bit.<br>• The Clear Timer 3 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL3 register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • Writing a 1 to Clear Timer 3 Enable bit will not enable the counter.<br>• The Clear Timer 3 Enable bit is self-clearing, which means that the Clear Timer 3 Enable bit will always read 0.<br><br>0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL3[T_EN]) for Timer Channel 3 |
| 2<br><br>CLR_T_EN_2 | Clear Timer 2 Enable<br>• To disable timer channel 2, write '1' to Clear Timer 2 Enable bit.<br>• The Clear Timer 2 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL2 register.<br>• Writing a 1 to Clear Timer 2 Enable bit will not enable the counter.<br>• The Clear Timer 2 Enable bit is self-clearing, which means that the Clear Timer 2 Enable bit will always read 0.<br><br>0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL2[T_EN]) for Timer Channel 2 |
| 1<br><br>CLR_T_EN_1 | Clear Timer 1 Enable<br>• To disable timer channel 1, write '1' to Clear Timer 1 Enable bit.<br>• The Clear Timer 1 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL1 register.<br>• Writing a 1 to Clear Timer 1 Enable bit will not enable the counter.<br>• The Clear Timer 1 Enable bit is self-clearing, which means that the Clear Timer 1 Enable bit will always read 0.<br><br>0b - No Action<br>1b - Clear the Timer Enable bit (TCTRL1[T_EN]) for Timer Channel 1 |
| 0<br><br>CLR_T_EN_0 | Clear Timer 0 Enable<br>• To disable timer channel 0, write '1' to Clear Timer 0 Enable bit.<br>• The Clear Timer 0 Enable bit can be used in addition to Timer Enable bit (T_EN) in TCTRL0 register.<br>• Writing a 1 to Clear Timer 0 Enable bit will not enable the counter.<br>• The Clear Timer 0 Enable bit is self-clearing, which means that the Clear Timer 0 Enable bit will always read 0.<br><br>0b - No action<br>1b - Clear the Timer Enable bit (TCTRL0[T_EN]) for Timer Channel 0 |

## 50.3.2.9 Timer Value Register (TVAL0 - TVAL3)

### 50.3.2.9.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TVALn | 20h + (n × 10h) |

### 50.3.2.9.2 Function

- In compare modes: the Timer Value Registers (TVAL*n*) select the timeout period for the timer channels.
- In capture modes: the Timer Value Registers (TVAL*n*) are loaded with the value of the counter when the trigger asserts.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), then reading or writing the TVALn register will generate a transfer error.

### 50.3.2.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 50.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>TMR_VAL | Timer Value<br>• In compare mode: Timer Value (TMR_VAL) is the timer channel start value.<br>  • The timer will count down until the timer reaches 0, then the timer will generate an interrupt and load the Timer Value register (TVAL*n*) value again.<br>  • Writing a new value to the Timer Value register (TVAL*n*) will not restart the timer channel; instead the new value will be loaded *after the timer expires*.<br>  • To abort the current timer cycle and start a timer period with a new value, the timer channel must be disabled and enabled *again*.<br>• In capture mode: whenever the trigger asserts, the Timer Value register stores the inverse of the counter value.<br><br>00000000000000000000000000000000b - Invalid load value in compare mode<br>00000000000000000000000000000001b - Invalid load value in compare mode<br>00000000000000000000000000000010-11111111111111111111111111111111b - In compare mode: the value to be loaded; in capture mode, the value of the timer |

## 50.3.2.10 Current Timer Value (CVAL0 - CVAL3)

### 50.3.2.10.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| CVALn | 24h + (n × 10h) |

### 50.3.2.10.2 Function

The Current Timer Value registers indicate the current timer counter value.

### 50.3.2.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 50.3.2.10.4 Fields

| Field | Function |
|---|---|
| 31-0 | Current Timer Value |
| TMR_CUR_VAL | Represents the current timer value, if the timer is enabled |

## 50.3.2.11 Timer Control Register (TCTRL0 - TCTRL3)

### 50.3.2.11.1 Offset

For n = 0 to 3:

| Register | Offset |
|---|---|
| TCTRLn | 28h + (n × 10h) |

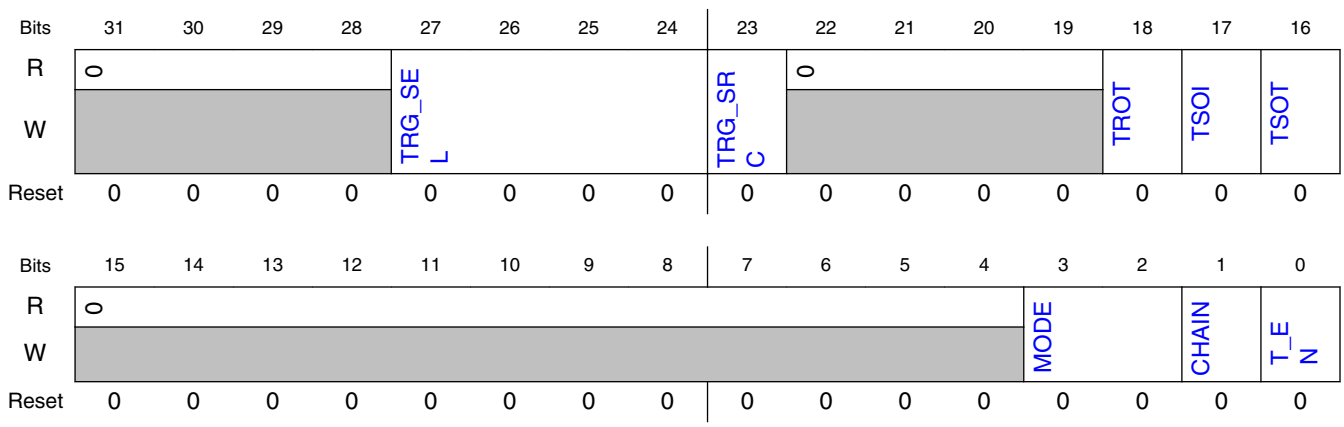
### 50.3.2.11.2 Function

The Timer Control registers contain the control bits for each timer channel.

**NOTE**

Unless the Module Clock Enable bit (MCR[M_CEN]) is set (=1, peripheral clock to timers is enabled), reading or writing the Timer Control register will generate a transfer error.

### 50.3.2.11.3 Diagram




### 50.3.2.11.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | This read-only field is reserved and always has the value 0 |
| 27-24<br>TRG_SEL | Trigger Select<br>Selects the trigger to use for starting and/or reloading the LPIT timer.<br>• The TRG_SEL field selects one trigger from the set of internal or external triggers that are selected by the Trigger Source bit (TRG_SRC)<br>• Recall that the TRG_SRC bit selects between internal and external trigger signals for each channel<br><br>**NOTE:** The Trigger Select field should only be changed when the LPIT timer channel is disabled.<br>0000-0011b - Timer channel 0 - 3 trigger source is selected<br>0100-1111b - Reserved |
| 23<br>TRG_SRC | Trigger Source<br>Selects between internal or external trigger sources. The trigger to be used is selected using the TRG_SRC and TRG_SEL bits. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Refer to the chip configuration section for available external trigger options. If a channel does not have an associated external trigger, then set the Trigger Source bit (TRG_SRC) = 1.<br>0b - Selects external triggers<br>1b - Selects internal triggers |
| 22-19<br>— | This read-only field is reserved and always has the value 0 |
| 18<br>TROT | Timer Reload On Trigger<br>When set, the LPIT timer will reload when a rising edge is detected on the selected trigger input. The trigger input is ignored if the LPIT is disabled during debug mode (DBGEN = 0) or DOZE mode (DOZE_EN = 0)<br>0b - Timer will not reload on the selected trigger<br>1b - Timer will reload on the selected trigger |
| 17<br>TSOI | Timer Stop On Interrupt<br>Controls whether the channel timer will stop after it (the channel timer) times out or not.<br>0b - The channel timer does not stop after timeout<br>1b - The channel timer will stop after a timeout, and the channel timer will restart based on Timer Start On Trigger bit (TSOT). When TSOT = 0, the channel timer will restart after a rising edge on the Timer Enable bit (T_EN) is detected (which means that the timer channel is disabled and then enabled). When TSOT = 1, the channel timer will restart after a rising edge on the selected trigger is detected. |
| 16<br>TSOT | Timer Start On Trigger<br>Controls when the timer starts decrementing.<br>0b - Timer starts to decrement immediately based on the restart condition (controlled by the Timer Stop On Interrupt bit (TSOI))<br>1b - Timer starts to decrement when a rising edge on a selected trigger is detected |
| 15-4<br>— | This read-only field is reserved and always has the value 0 |
| 3-2<br>MODE | Timer Operation Mode<br>Configures the channel timer's mode of operation. The MODE bits control how the timer decrements.<br>00b - 32-bit Periodic Counter<br>01b - Dual 16-bit Periodic Counter<br>10b - 32-bit Trigger Accumulator<br>11b - 32-bit Trigger Input Capture |
| 1<br>CHAIN | Chain Channel<br>When enabled, the timer channel will decrement when timer channel N-1 trigger asserts. Timer channel 0 cannot be chained.<br>0b - Channel Chaining is disabled. The channel timer runs independently.<br>1b - Channel Chaining is enabled. The timer decrements on the previous channel's timeout. |
| 0<br>T_EN | Timer Enable<br>Enables or disables the Timer Channel<br>0b - Timer Channel is disabled<br>1b - Timer Channel is enabled |

# 50.4 Functional description

## 50.4.1 LPIT programming model

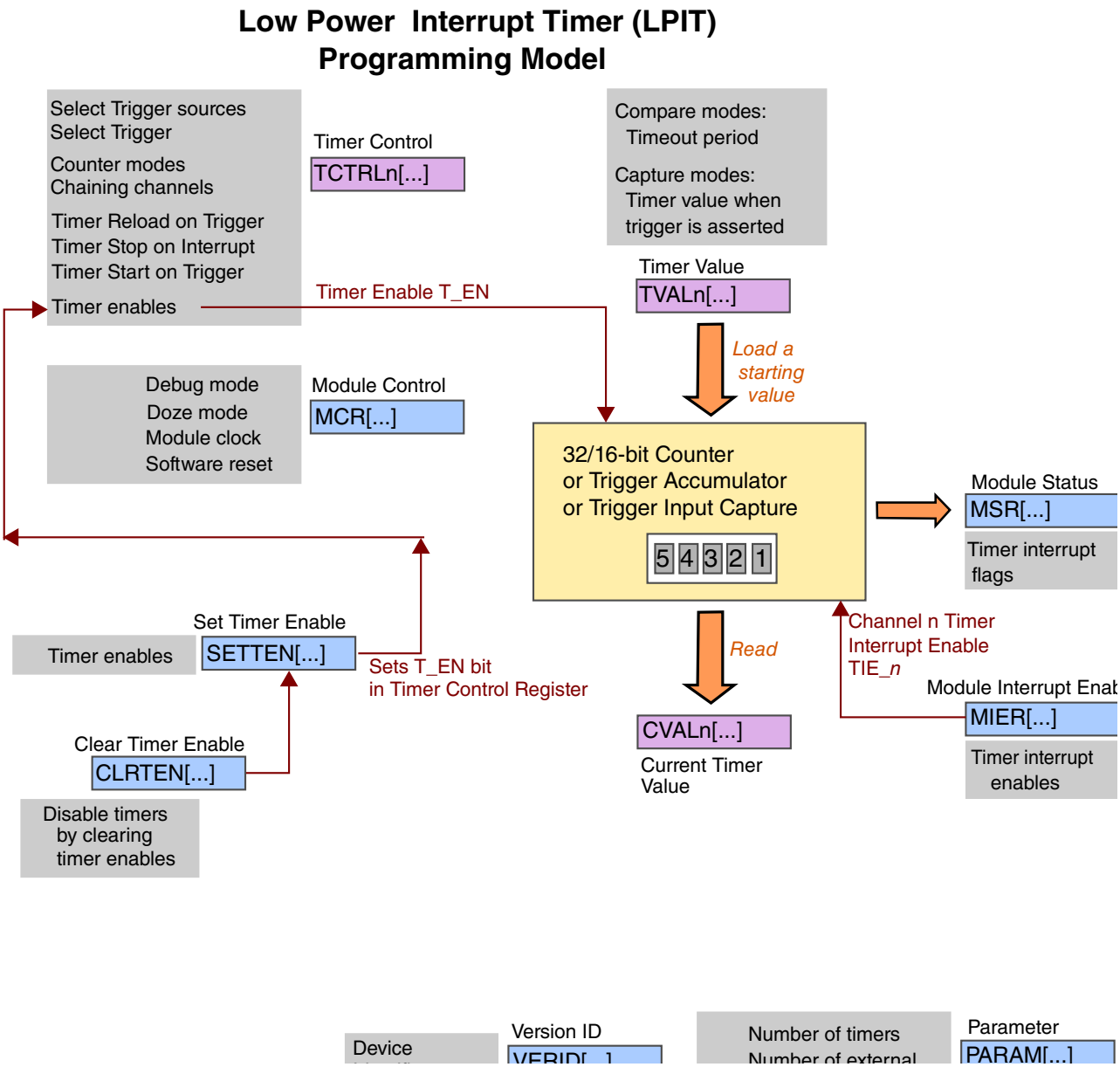



**Figure 50-3. Programming Model**

Global registers need only be set once:

- Version ID Register (VERID)
- Parameter Register (PARAM)
- Module Control Register (MCR)
- Module Status Register (MSR)
- Module Interrupt Enable Register (MIER)

- Set Timer Enable Register (SETTEN)
- Clear Timer Enable Register (CLRTEN)

Channel registers must be set for each channel:

- Timer Value Register (TVAL0 - TVAL3)
- Current Timer Value (CVAL0 - CVAL3)
- Timer Control Register (TCTRL0 - TCTRL3)

## 50.4.2 Initialization

**Table 50-2. Initializing the LPIT module**

| Step | | How? Why? |
|---|---|---|
| 1 | Enable the peripheral clock | by setting the Module Clock Enable bit (M_CEN) in the Module Control Register (MCR) register.<br>**NOTE:**<br>• Writing to certain registers (MSR, SETTEN, CLRTEN, TVAL, and TCTRL) while M_CEN = 0 will lead to assertion of a transfer error for that bus access.<br>  • Writing to CVAL and Reserved registers will generate a transfer error, regardless of the value of the Module Clock Enable bit (M_CEN) bit.<br>  • Reads to CVAL and Reserved registers can happen regardless of the M_CEN bit value.<br>• There might be additional clock gating bits in the device that gate the peripheral clock to this module. When enabling the clock to this module, ensure that software is configuring those additional clock gating bits (in addition to M_CEN bit). |
| 2 | Wait for 4 peripheral clock cycles | to allow time for clock synchronization and reset de-assertion. |
| 3 | Configure timer control bits | for each timer channel that is to be enabled:<br>• timer mode of operation bits TCTRLn[MODE]<br>• trigger source selection bits TCTRLn[TRG_SEL, TRG_SRC]<br>• trigger control bits TCTRLn[TROT, TSOT, TSOI] |
| 4 | Configure the channels that are to be chained | by setting the CHAIN bit in the corresponding channel's TCTRLn register. |
| 5 | Set the timer timeout value | for channels configured in Compare Mode, by programming the appropriate value in TVAL register for those channels. |
| 6 | Configure TIEn bits in MIER register | for those channels that are required to generate interrupts on timer timeouts. |
| 7 | Configure the low power modes of the module | by setting the DBG_EN and DOZE_EN bits in the MCR register. This is common to all timer channels. |
| 8 | Enable the channel timers | by setting the corresponding T_EN bit in the corresponding channel's TCRTLn register. |

*Also:*

- For channels configured in Capture Mode, the timer value can be read from TVALn register when a channel timeout occurs.

- At any time, the current value of the timer for any channel can be read by reading the corresponding channel's CVALn register.
- The timer interrupt flag bits (TIFn) in MSR register get asserted on timer timeout. To clear these timer interrupt flag bits, write '1' to them.

## 50.4.3 Timer Modes

The timer mode is configured by setting an appropriate value in the MODE bits in TCTRLn register.

**Table 50-3. Timer modes that are supported**

| Timer Mode | Operation |
|---|---|
| 32-bit Periodic Counter | • The counter will load, decrement down to 0<br>• which will set the timer interrupt flag and assert the output pre-trigger. |
| Dual 16-bit Periodic Counter | • The counter will load, then the lower 16-bits will decrement down to 0<br>• which will assert the output pre-trigger.<br>• The upper 16-bits will then decrement down to 0<br>• which will set the timer interrupt flag and then negate the output pre-trigger. |
| 32-bit Trigger Accumulator | • The counter will load on the 1st trigger rising edge and then decrement down to 0 on each trigger rising edge<br>• which will set the timer interrupt flag and assert the output pre-trigger. |
| 32-bit Trigger Input Capture | • The counter will load with 0xFFFF_FFFF and then decrement down to 0.<br>• If a trigger rising edge is detected,<br>  • then it will store the inverse of the current counter value in the timer value register<br>  • which will set the timer interrupt flag and assert the output pre-trigger. |

The timer operation is controlled by Trigger Control bits (TSOT, TSOI, TROT) which control the timer load, reload, start and restart of the timers.

**NOTE**

- The trigger output is asserted one Peripheral Timer Clock cycle later than pre-trigger output. The trigger output and the pre-trigger output de-assert at the same time.
- The pre-trigger output is asserted for 2 clock cycles and the trigger output is asserted for 1 clock cycle (except in 16-bit Periodic Counter mode, where both pre-trigger and trigger are asserted for many cycles depending on TMR_VAL[31:16]).
- Timer changes *that are based on external triggers* take effect after 4 peripheral clocks after the actual external trigger assertion (due to clock synchronization, rise edge detection and timer update).

### 50.4.4 Trigger Control for Timers

Various programmable bits control how the trigger inputs and the timer operate. TRG_SEL and TRG_SRC select the timer triggers:

- Trigger Select (TRG_SEL) selects the input trigger for the channel from all of the other channel's trigger outputs.
- Trigger Source (TRG_SRC) selects between the internal trigger and the external trigger inputs to the channel.

The selected trigger affects how the timer operates, using the configuration of the TROT, TSOI and TSOT bits.

**Table 50-4. How bits control timer operations**

| If | = | Then |
|---|---|---|
| TSOI<br>*Timer Stop On Interrupt* | 1 | then the counter stops on a Timer Interrupt flag (MSR[TIFn]) assertion. To reload and decrement, it requires:<br>• a trigger (if TSOT = 1)<br>• a T_EN rising edge (if TSOT = 0) |
| | 0 | then the counter does not stop after timeout |
| TROT<br>*Timer Reload On Trigger* | 1 | then the counter is loaded on each trigger |
| | 0 | then the counter is loaded on every T_EN rising edge or timeout rising edge (timeout not used in Capture modes) |
| TSOT<br>*Timer Start On Trigger* | 1 | then the counter will start to decrement on a trigger. Subsequent triggers are ignored until the counter times out. |
| | 0 | then the counter decrements immediately on the next clock edge. When channel is Chained or in Capture mode, TSOT has no effect. |

In different timer modes, these programmable bits affect the timer operation differently:

**Table 50-5. Time modes and programmable bits**

| Mode | Which programmable bits affect timer operations |
|---|---|
| 32-bit Periodic Counter | All bits (TSOT, TSOI, TROT) affect the timer operation as described previously. |
| Dual 16-bit Periodic Counter | |
| 32-bit Trigger Accumulator | • Only the TSOI bit controls the timer function.<br>• TROT and TSOT bits have no effect on timer operations. |
| 32-bit Input Trigger Capture | • Only TSOI and TROT bits control the timer function.<br>• TSOT bit has no effect on timer operations. |

## 50.4.5 Channel Chaining

Individual timer channels can be chained together to achieve a larger value of timeout. Chaining the timer channel causes them to work in a *'nested loop'* manner thereby leading to an effective timeout value of $TVAL_{CHn} \times (TVAL_{CHn-1} + 1)$.

The channels are chained by setting the CHAIN bit in corresponding channel's $TCTRL_{CHn}$ register. When a channel is chained, that channel's timer decrements on previous channel's timeout pulse, regardless of the timer mode (MODE bits). The TSOT bit does not have any effect if the channel timer (Channel 'n') is chained to previous channel's timer (Channel 'n-1').

## 50.4.6 Detailed timing

**NOTE**

The timing diagrams in these sections are not *cycle-accurate* (i.e., some cycles may not be shown), but the timing diagrams do show the timer channel behavior across several clock cycles.

**Table 50-6. Timing diagrams list**

| Mode | | Timing Diagram | TSOT | TROT | TSOI | CHAIN |
|---|---|---|---|---|---|---|
| 32-bit periodic counter (compare mode) | 00 | Figure 50-4 | 0 | 0 | 0 | 0 |
| | | Figure 50-5 | 0 | 0 | 1 | 0 |
| | | Figure 50-6 | 0 | 1 | 0 | 0 |
| | | Figure 50-7 | 0 | 1 | 1 | 0 |
| | | Figure 50-8 | 1 | 0 | 0 | 0 |
| | | Figure 50-9 | 1 | 0 | 1 | 0 |
| | | Figure 50-10 | 1 | 1 | 0 | 0 |
| | | Figure 50-11 | 1 | 1 | 1 | 0 |
| 16-bit dual periodic counter (compare mode) | 01 | Figure 50-12 | 0 | 0 | 0 | 0 |
| 32-bit trigger accumulator mode | 10 | Figure 50-13 | X | X | 0 | 0 |
| | | Figure 50-14 | X | X | 1 | 0 |
| 32-bit trigger capture mode | 11 | Figure 50-15 | X | 0 | 0 | 0 |
| | | Figure 50-16 | X | 1 | 0 | 0 |
| | | Figure 50-17 | X | 0 | 1 | 0 |
| Timer chaining: effects on timing operations | - | Figure 50-18 | - | - | - | - |

### 50.4.6.1 Mode=00: 32-bit periodic counter (compare mode)

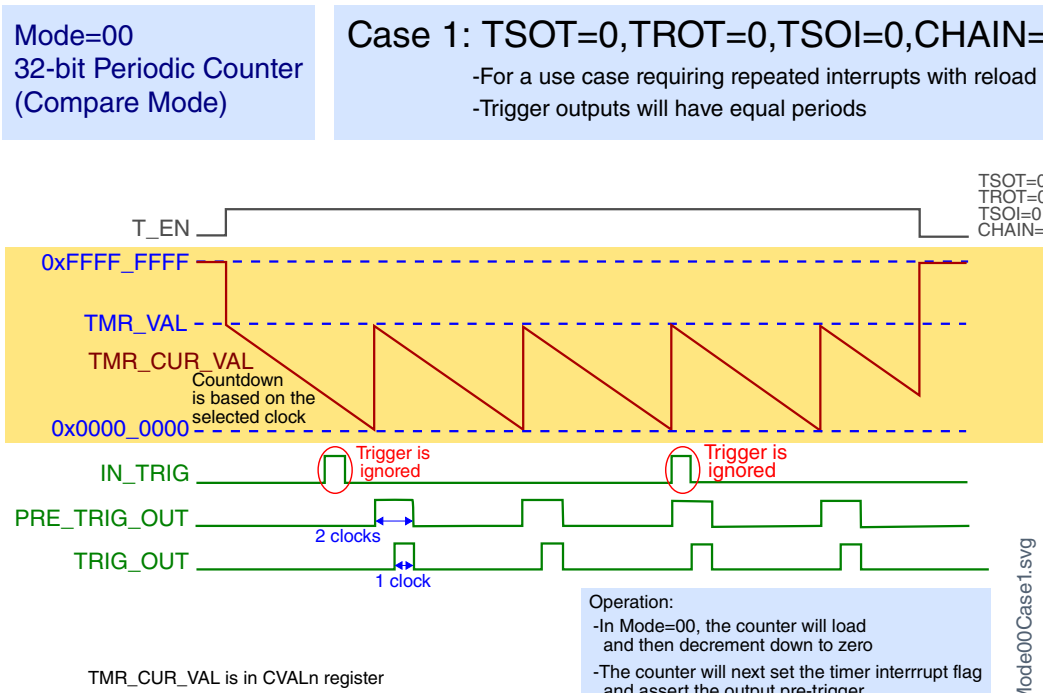



**Figure 50-4. Case 1: TSOT = 0, TROT = 0, TSOI = 0, CHAIN = 0**

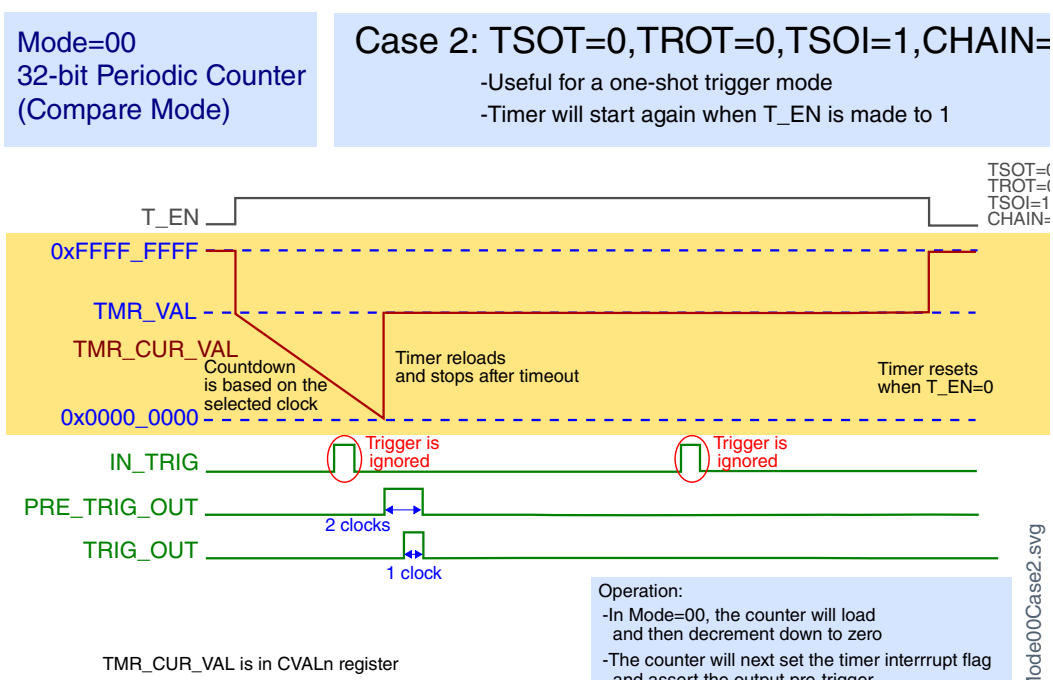



**Figure 50-5. Case 2: TSOT = 0, TROT = 0, TSOI = 1, CHAIN = 0**

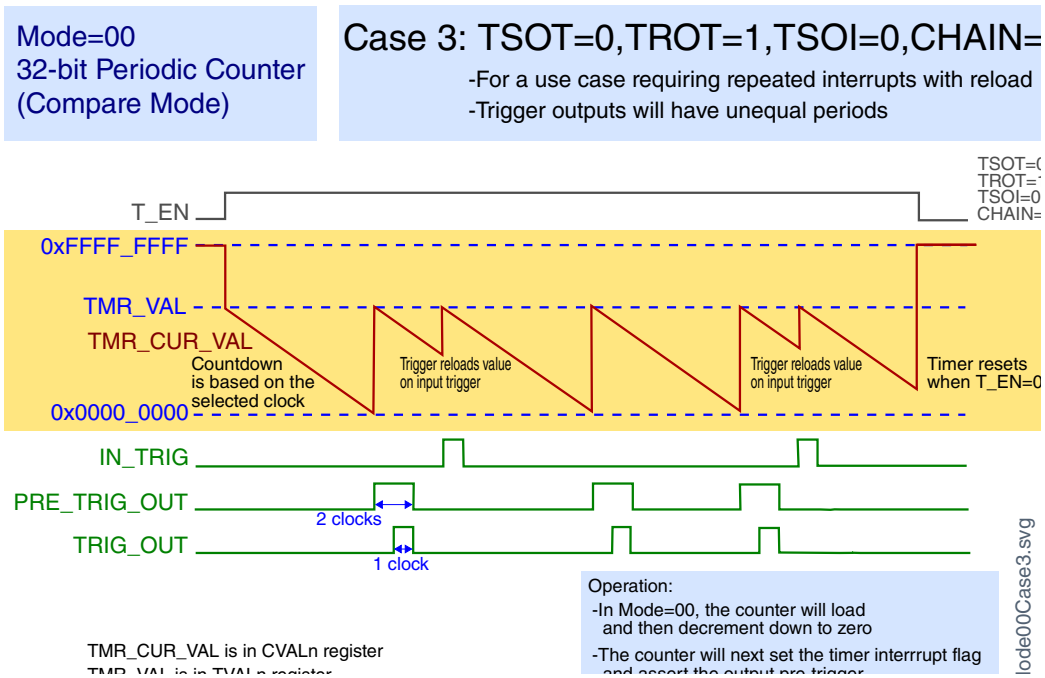



**Figure 50-6. Case 3: TSOT = 0, TROT = 1, TSOI = 0, CHAIN = 0**

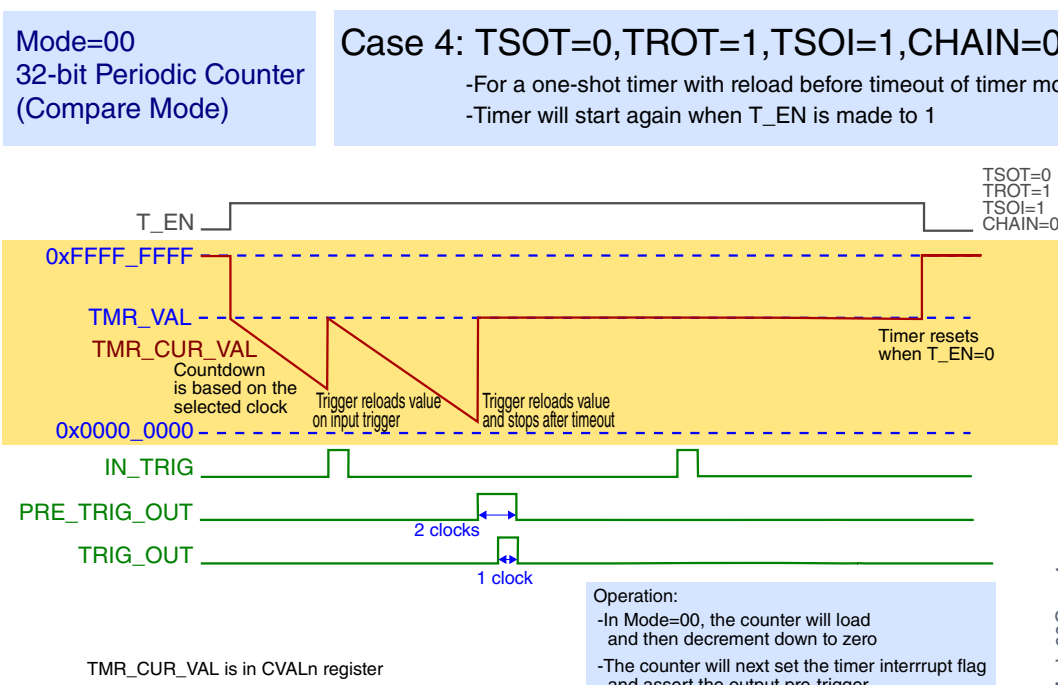



**Figure 50-7. Case 4: TSOT = 0, TROT = 1, TSOI = 1, CHAIN = 0**

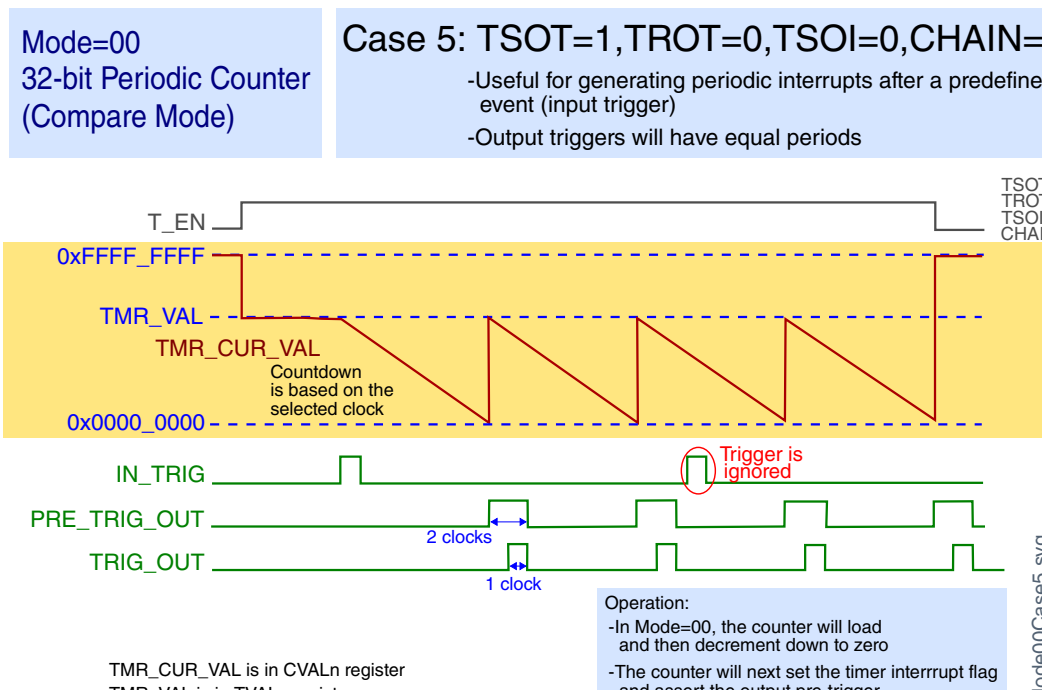



**Figure 50-8. Case 5: TSOT = 1, TROT = 0, TSOI = 0, CHAIN = 0**

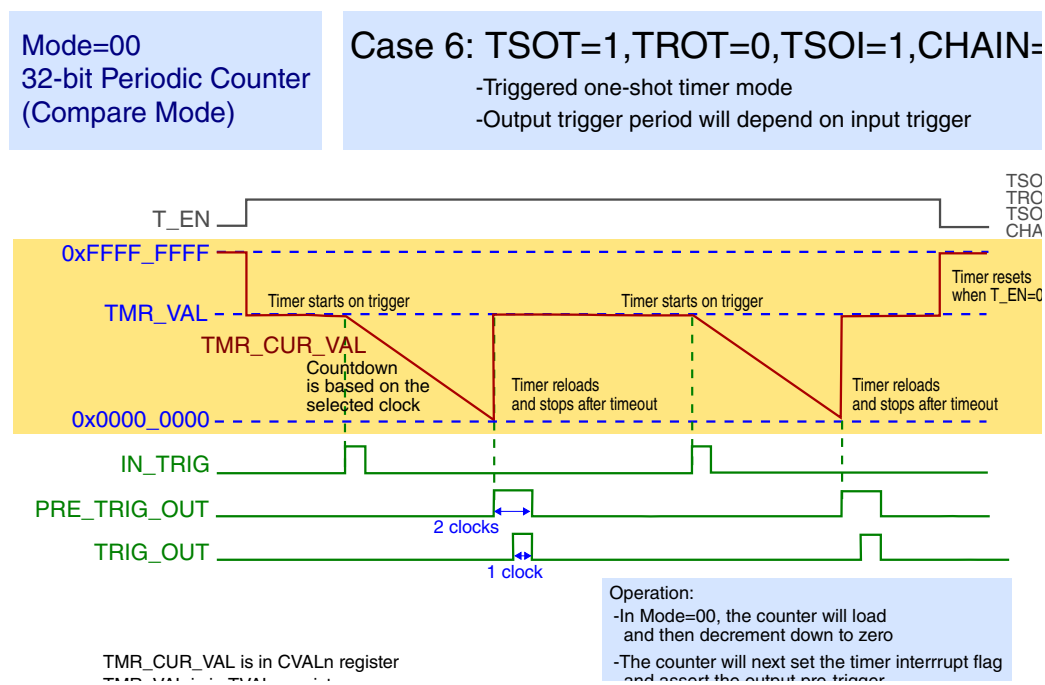



**Figure 50-9. Case 6: TSOT = 1, TROT = 0, TSOI = 1, CHAIN = 0**

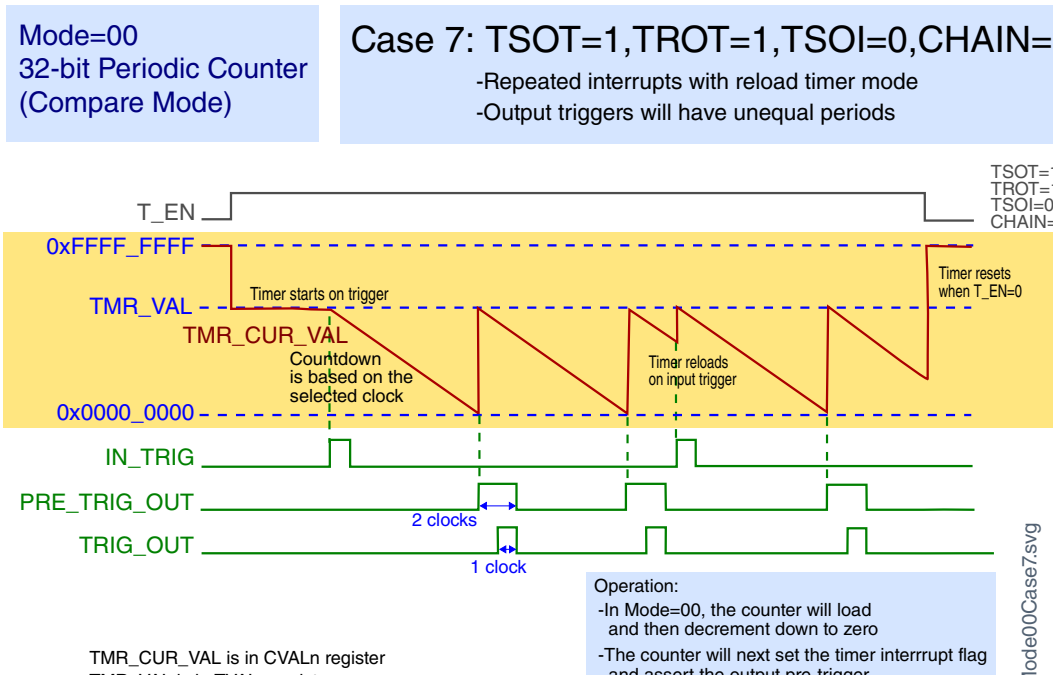



**Figure 50-10. Case 7: TSOT = 1, TROT = 1, TSOI = 0, CHAIN = 0**

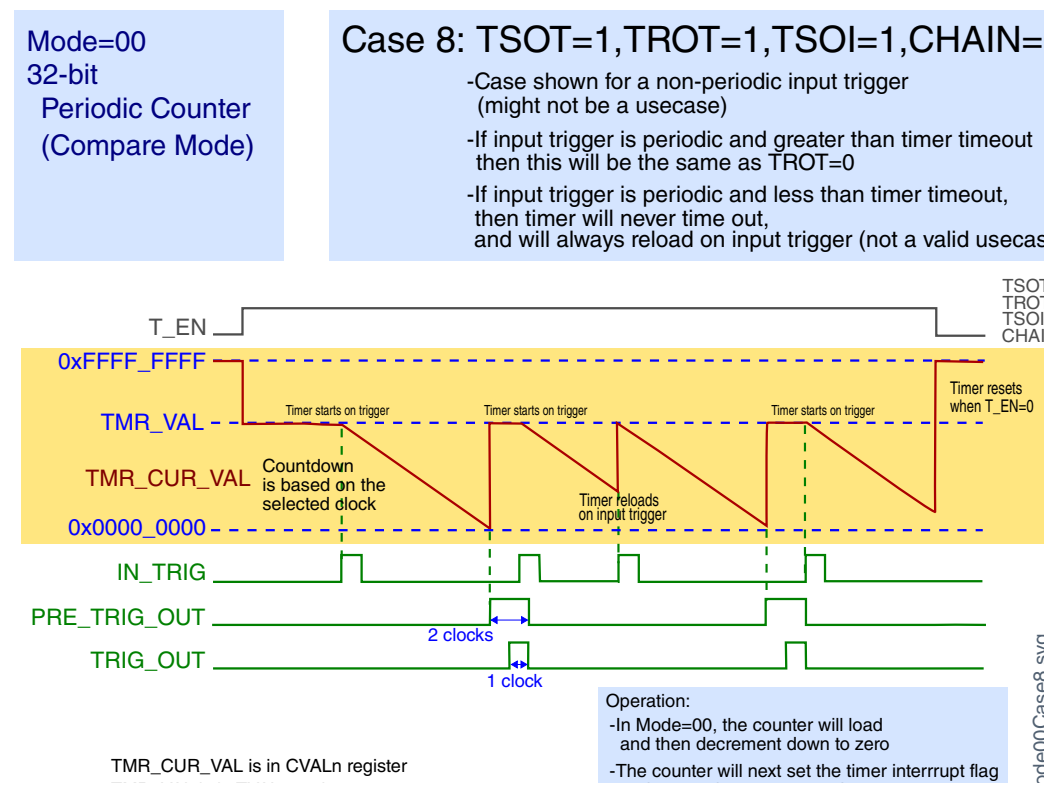



**Figure 50-11. Case 8: TSOT = 1, TROT = 1, TSOI = 1, CHAIN = 0**

### 50.4.6.2 Mode=01: 16-bit dual periodic counter (compare mode)

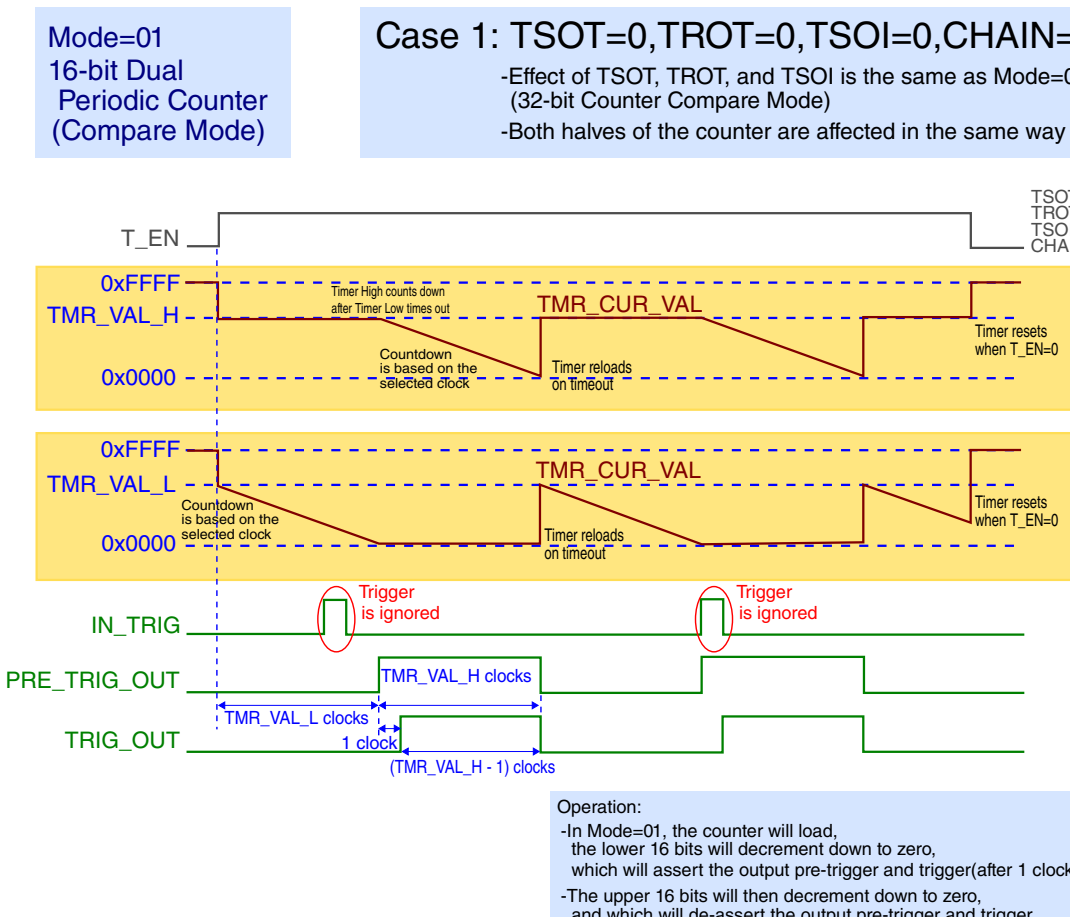



**Figure 50-12. Case 1: TSOT = 0, TROT = 0, TSOI = 0, CHAIN = 0**

## Effect of Timer Control Bits in Mode=01:

- Effect of timer control bits are the same as for Mode=00. Refer to the individual figures of Mode=00.
- The Timer Interrupt (timeout) asserts when {TMR_H,TMR_L} = 0x0000_0000.
- Behavior for Mode = 01 is explained in the next table.

**Table 50-7. Mode=01 timer control bits**

| TSOT | TROT | TSOI | Function | Effect on Timer |
|---|---|---|---|---|
| 0 | 0 | 0 | • For repeated interrupts with reload<br>• Trigger outputs will have equal periods | Similar to Figure 50-12. |
| 0 | 0 | 1 | One-shot mode | • Both timers stop after first count down and then time out<br>• Timers will not count again until T_EN is made 1 again<br>• Similar to Figure 50-5. |

*Table continues on the next page...*

## Table 50-7. Mode=01 timer control bits (continued)

| TSOT | TROT | TSOI | Function | Effect on Timer |
|---|---|---|---|---|
| 0 | 1 | 0 | • For repeated interrupts with reload<br>• Trigger outputs will have unequal periods | • Both timers will reload TMR_VAL on trigger rise edge<br>• Output triggers will clear on reload, if asserted<br>• Similar to Figure 50-6. |
| 0 | 1 | 1 | Reloadable one-shot mode | • If a trigger occurs before timeout, then both timers reload and count down (as shown); the timers stop after timeout<br>• A trigger assertion after timeout simply reloads TMR_VAL into the timers<br>• The timers will not count again until T_EN is set to 1 again<br>• Similar to Figure 50-7. |
| 1 | 0 | 0 | • For generating periodic interrupts after a predefined event (input trigger)<br>• Output triggers will have equal periods | After T_EN rises, the timers do not start until the first trigger's rising edge<br>• Subsequent triggers will have no effect<br>• Similar to Figure 50-8. |
| 1 | 0 | 1 | • Triggered one-shot timer mode<br>• Output trigger period will depend on input trigger | After T_EN rises, the timers do not start until the first trigger's rising edge.<br>• The timer stops counting after a timeout assertion (a timeout is asserted)<br>• The timer does not start counting again until a new trigger's rising edge is detected<br>• Similar to Figure 50-9. |
| 1 | 1 | 0 | • For repeated interrupts with reload timer mode<br>• Output triggers will have unequal periods | After T_EN rises, the timers do not start until the first trigger's rising edge<br>• Subsequent triggers will cause the timer to reload TMR_VAL into both counters<br>• The output triggers will clear after a reload, if asserted<br>• Similar to Figure 50-10. |
| 1 | 1 | 1 | • For a non-periodic input trigger<br>• If input trigger is periodic and greater than timer timeout, then this will be the same as TROT=0 (which is triggered one-shot timer mode; output trigger period will depend on input trigger) | After T_EN rises, the timers do not start until the first trigger's rising edge<br>• The timers stops counting after a timeout assertion (a timeout is asserted)<br>• A trigger's rising edge will cause the timers to reload and then count down<br>• Similar to Figure 50-11. |

### 50.4.6.3 Mode=10: 32-bit trigger accumulator mode

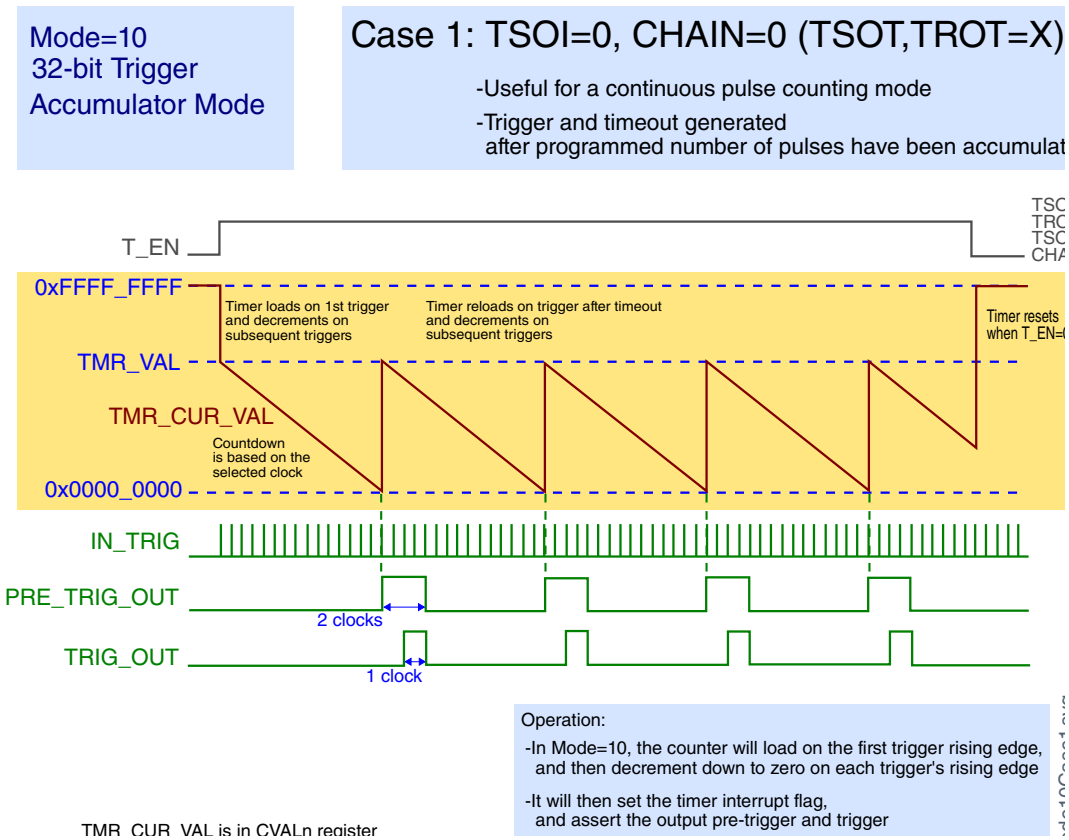



**Figure 50-13. Case 1: TSOT = X, TROT = X, TSOI = 0, CHAIN = 0**

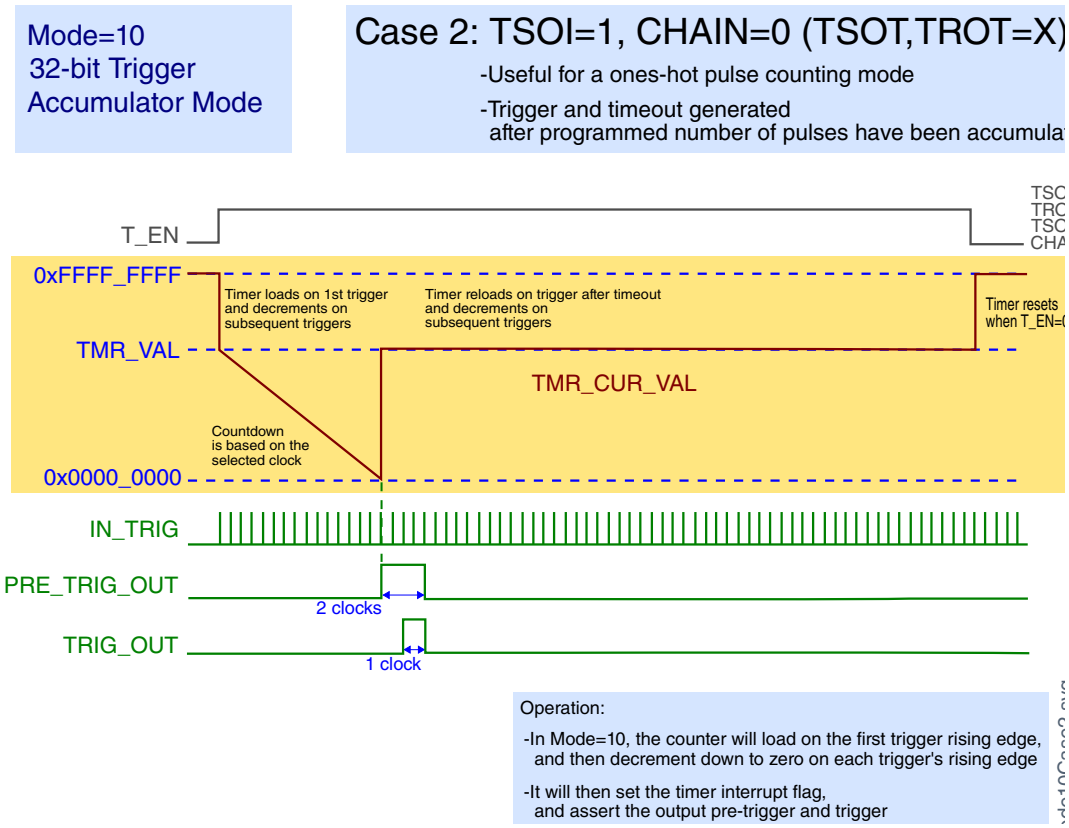



**Figure 50-14. Case 2: TSOT = X, TROT = X, TSOI = 1, CHAIN = 0**

### 50.4.6.4 Mode=11: 32-bit trigger capture mode

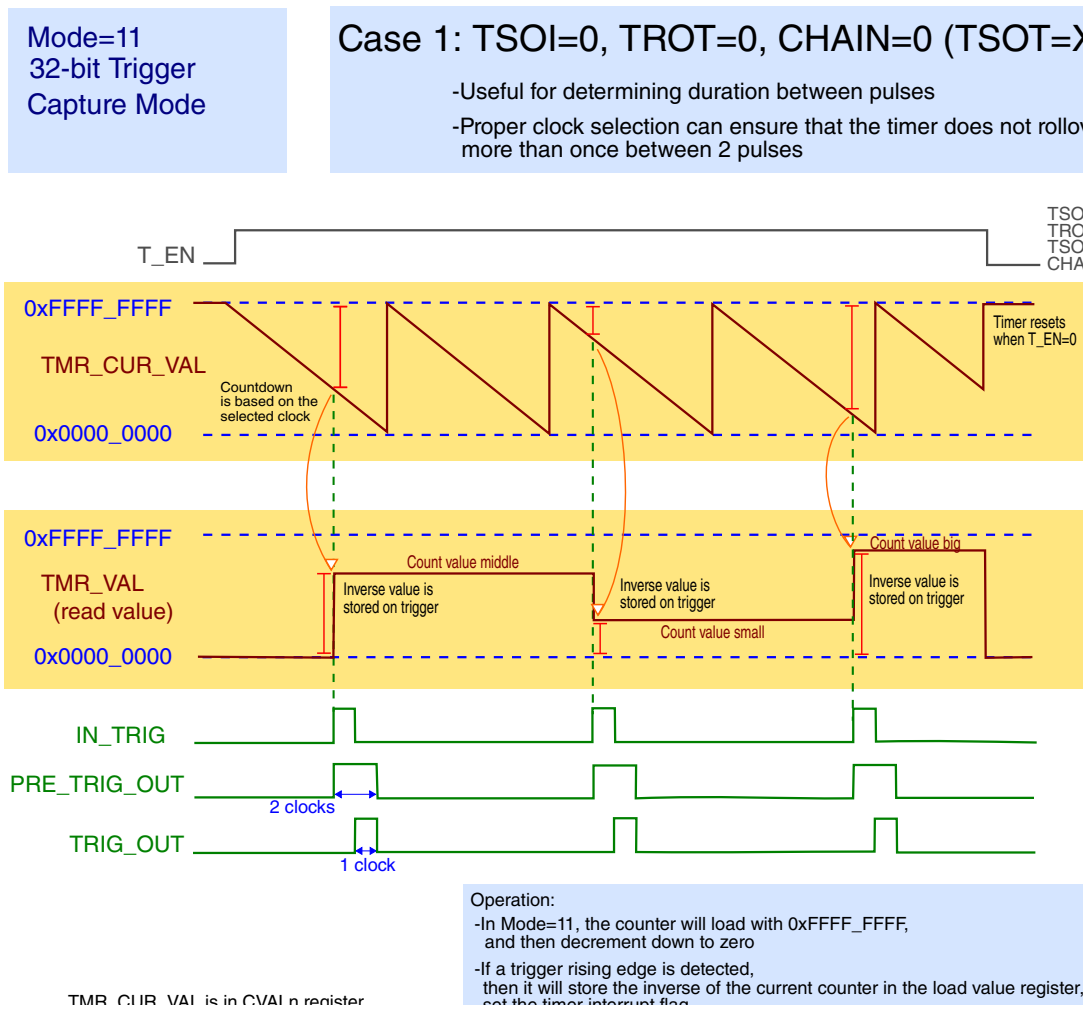



**Figure 50-15. Case 1: TSOT = X, TROT = 0, TSOI = 0, CHAIN = 0**

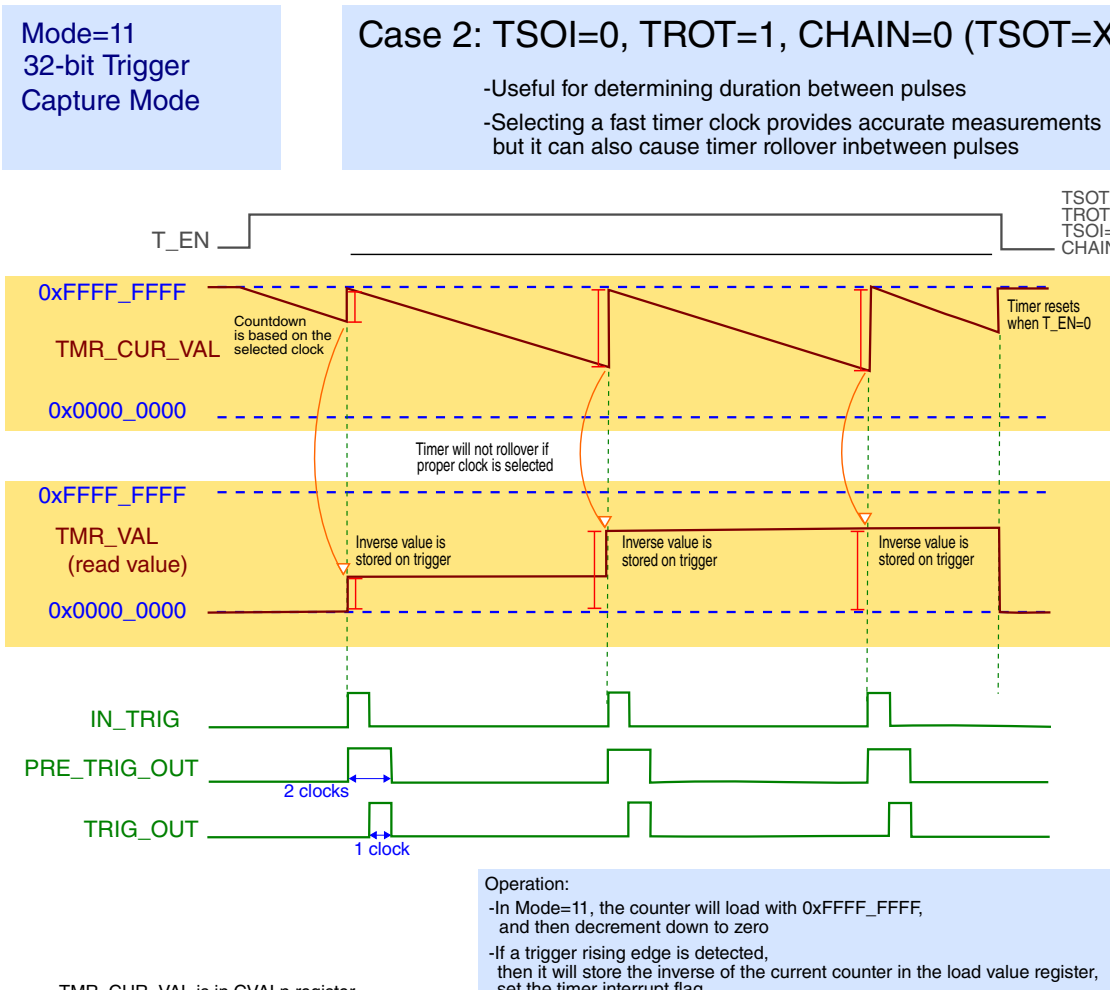



**Figure 50-16. Case 2: TSOT = X, TROT = 1, TSOI = 0, CHAIN = 0**

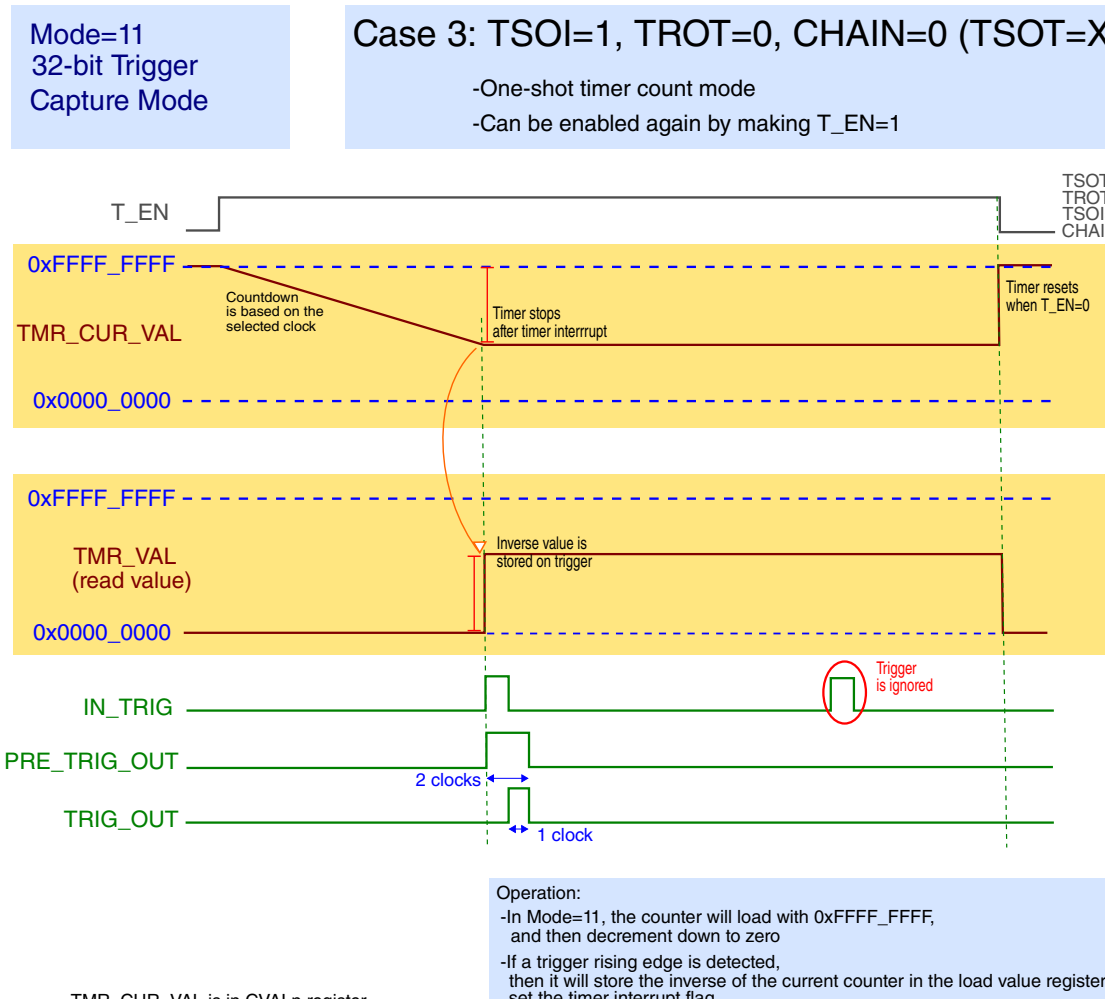



**Figure 50-17. Case 3: TSOT = X, TROT = 0, TSOI = 1, CHAIN = 0**

## Case 4: TSOT = X, TROT = 1, TSOI = 1, CHAIN = 0

Same as previous case (Case 3), except that the timer reloads to 0xFFFF_FFFF and then stops. The timer does not start until T_EN is made 1 again. Case 4 is not a very useful case, because Case 3 covers this.

### 50.4.6.5 Timer chaining

| Effect of Chaining | -Chaining causes Timer “n” to decrement on every timeout pulse (trigger output pulse) from Timer “n – 1”, regardless of what mode is configured in Timer “n”<br>-Timers “n” and “n – 1” effectively form a larger width timer (64-bits)<br>-More than 2 timer channels (or all timer channels) can be chained<br>-It is preferred to have the same trigger source and timer controls configured for chained channels |
|---|---|

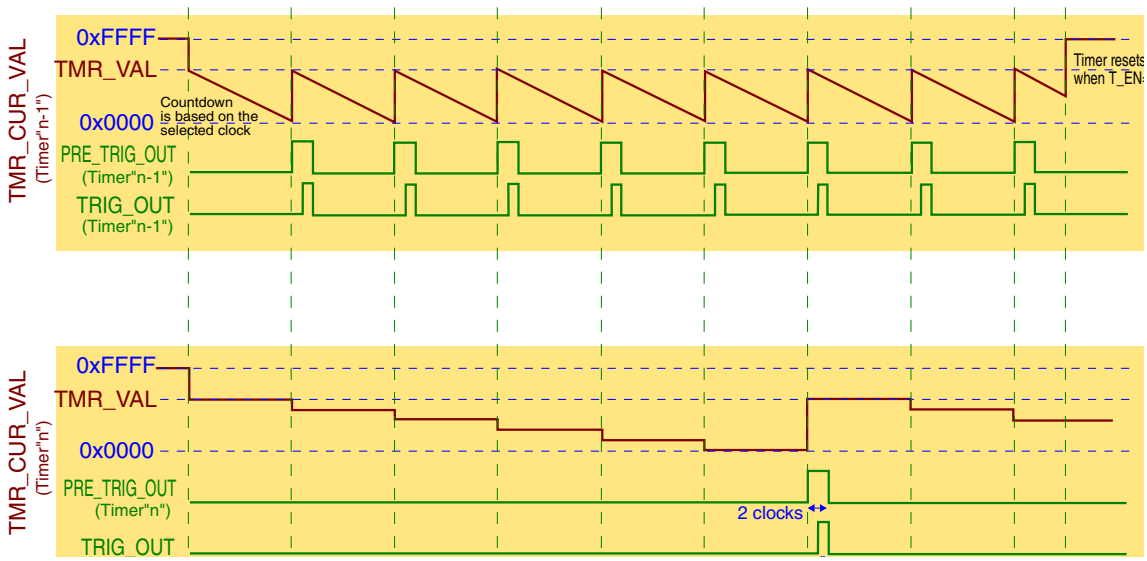



**Figure 50-18. Chaining Effects**

# Chapter 51
# Timer/PWM Module (TPM)

## 51.1 Chip-specific TPM information

### 51.1.1 TPM Instantiation Information

This device contains 4 TPM modules. All TPM modules in the device are configured with basic TPM functionality with no quadrature decoder function and all can be functional in Stop/VLPS mode. The clock source is either external or internal in Stop/VLPS mode.

The following table shows how these modules are configured.

**Table 51-1. TPM configuration**

| TPM instance | Number of channels | Features/usage |
|---|---|---|
| TPM0 | 6 | Basic TPM,functional in Stop/VLPS mode |
| TPM1 | 2 | Basic TPM,functional in Stop/VLPS mode |
| TPM2 | 6 | Basic TPM,functional in Stop/VLPS mode |
| TPM3 | 2 | Basic TPM,functional in Stop/VLPS mode |

### 51.1.2 Global Timebase

Each TPM has a global timebase feature controlled by the TPMx_CONF[GTBEEN] bit. TPM1 is configured as the global time when this option is enabled.

### 51.1.3 TPM Interrupts

The TPM has multiple sources of interrupt. However, these sources are OR'd together to generate a single interrupt request to the interrupt controller. When a TPM interrupt occurs, read the TPM status registers to determine the exact interrupt source.

## 51.2 Introduction

The TPM (Timer/PWM Module) is a 2- to 8-channel timer which supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications.

The counter, compare and capture registers are clocked by an asynchronous clock that can remain enabled in low power modes. An example of using the TPM with the asynchronous DMA is described in AN4631:Using the Asynchronous DMA features of the Kinetis L Series .

### 51.2.1 TPM Philosophy

The TPM is built upon a very simple timer (HCS08 Timer PWM Module – TPM) used for many years on 8-bit microcontrollers. The TPM extends the functionality to support operation in low power modes by clocking the counter, compare and capture registers from an asynchronous clock that can remain functional in low power modes.

### 51.2.2 Features

The TPM features include:

- TPM clock mode is selectable
  - Can increment on every edge of the asynchronous counter clock
  - Can increment on rising edge of an external clock input synchronized to the asynchronous counter clock
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- TPM includes a 16-bit counter
  - It can be a free-running counter or modulo counter
  - The counting can be up or up-down

- Includes 6 channels that can be configured for input capture, output compare, edge-aligned PWM mode, or center-aligned PWM mode
  - In input capture mode the capture can occur on rising edges, falling edges or both edges
  - In output compare mode the output signal can be set, cleared, pulsed, or toggled on match
  - All channels can be configured for edge-aligned PWM mode or center-aligned PWM mode
- Support the generation of an interrupt and/or DMA request per channel
- Support the generation of an interrupt and/or DMA request when the counter overflows
- Support selectable trigger input to optionally reset or cause the counter to start incrementing.
  - The counter can also optionally stop incrementing on counter overflow
- Support the generation of hardware triggers when the counter overflows and per channel

## 51.2.3 Modes of operation

During debug mode, the TPM can can be configured to temporarily pause all counting until the core returns to normal user operating mode or to operate normally. When the counter is paused, trigger inputs and input capture events are ignored, and PWM outputs are forced to their initial state.

During doze mode, the TPM can be configured to operate normally or to pause all counting for the duration of doze mode. When the counter is paused, trigger inputs and input capture events are ignored, and PWM outputs are forced to their initial state.

During stop mode, the TPM counter clock can remain functional and the TPM can generate an asynchronous interrupt to exit the MCU from stop mode.

## 51.2.4 Block diagram

The TPM uses one input/output (I/O) pin per channel, CHn (TPM channel (n)) where n is the channel number.

The following figure shows the TPM structure. The central component of the TPM is the 16-bit counter with programmable final value and its counting can be up or up-down.

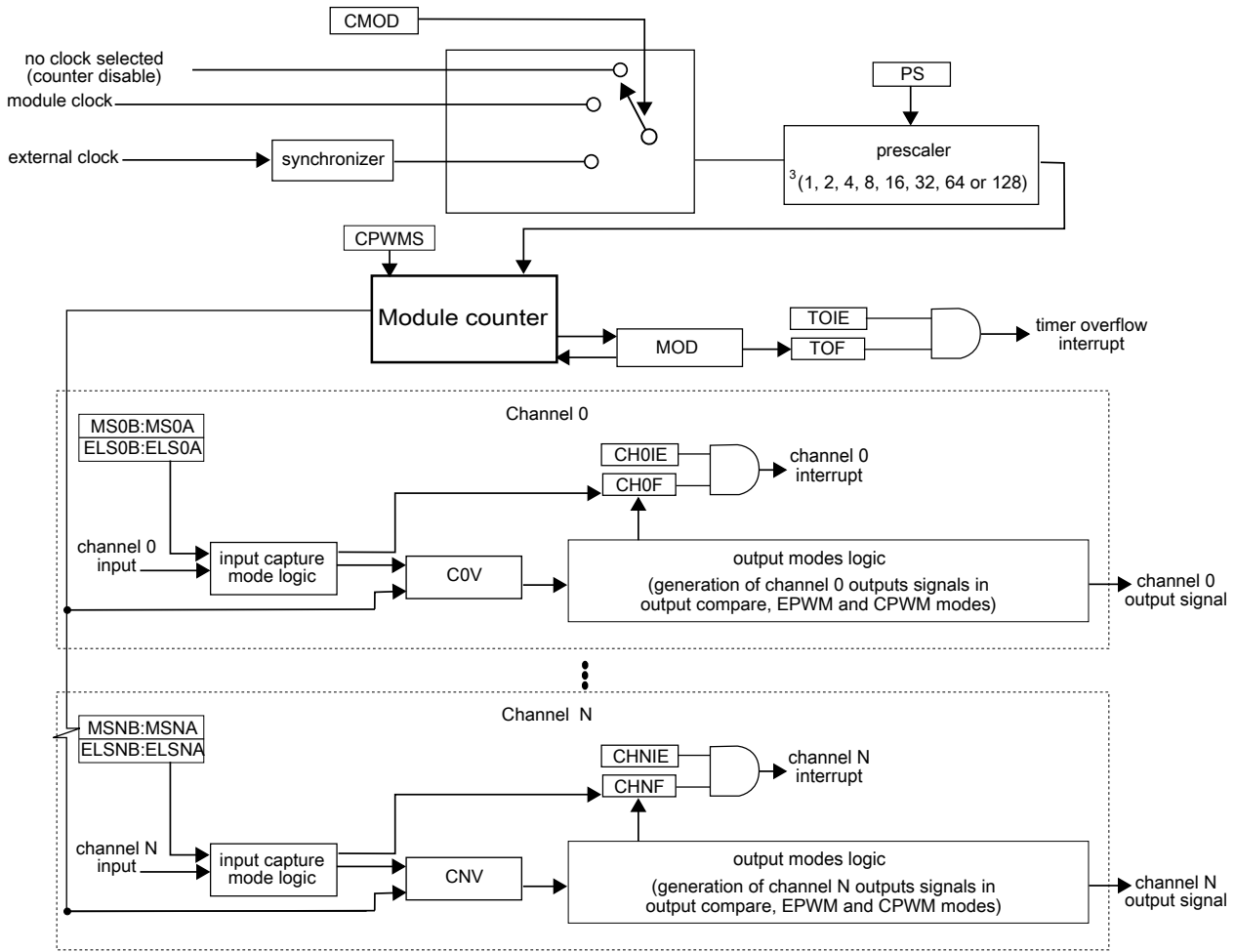



**Figure 51-1. TPM block diagram**

## 51.3 TPM Signal Descriptions

Table 51-2 shows the user-accessible signals for the TPM.

**Table 51-2. TPM signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| TPM_EXTCLK | External clock. TPM external clock can be selected to increment the counter on every rising edge synchronized to the counter clock. | I |
| TPM_CHn | TPM channel (n = 5 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input. | I/O |

### 51.3.1 TPM_EXTCLK — TPM External Clock

The rising edge of the external input signal is used to increment the TPM counter if selected by CMOD[1:0] bits in the SC register. This input signal must be less than half of the TPM counter clock frequency. The TPM counter prescaler selection and settings are also used when an external input is selected.

### 51.3.2 TPM_CHn — TPM Channel (n) I/O Pin

Each TPM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.

## 51.4 Memory Map and Register Definition

This section provides a detailed description of all TPM registers.

Attempting to access a reserved register location in the TPM memory map will generate a bus error.

### 51.4.1 TPM0 register descriptions

#### 51.4.1.1 TPM0 Memory map

TPM0 base address: 4003_5000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0600_0007h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0010_0406h |
| 8h | TPM Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| 10h | Status and Control (SC) | 32 | RW | 0000_0000h |
| 14h | Counter (CNT) | 32 | RW | 0000_0000h |
| 18h | Modulo (MOD) | 32 | RW | 0000_FFFFh |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1Ch | Capture and Compare Status (STATUS) | 32 | W1C | 0000_0000h |
| 20h | Channel (n) Status and Control (C0SC) | 32 | RW | 0000_0000h |
| 24h | Channel (n) Value (C0V) | 32 | RW | 0000_0000h |
| 28h | Channel (n) Status and Control (C1SC) | 32 | RW | 0000_0000h |
| 2Ch | Channel (n) Value (C1V) | 32 | RW | 0000_0000h |
| 30h | Channel (n) Status and Control (C2SC) | 32 | RW | 0000_0000h |
| 34h | Channel (n) Value (C2V) | 32 | RW | 0000_0000h |
| 38h | Channel (n) Status and Control (C3SC) | 32 | RW | 0000_0000h |
| 3Ch | Channel (n) Value (C3V) | 32 | RW | 0000_0000h |
| 40h | Channel (n) Status and Control (C4SC) | 32 | RW | 0000_0000h |
| 44h | Channel (n) Value (C4V) | 32 | RW | 0000_0000h |
| 48h | Channel (n) Status and Control (C5SC) | 32 | RW | 0000_0000h |
| 4Ch | Channel (n) Value (C5V) | 32 | RW | 0000_0000h |
| 64h | Combine Channel Register (COMBINE) | 32 | RW | 0000_0000h |
| 6Ch | Channel Trigger (TRIG) | 32 | RW | 0000_0000h |
| 70h | Channel Polarity (POL) | 32 | RW | 0000_0000h |
| 78h | Filter Control (FILTER) | 32 | RW | 0000_0000h |
| 80h | Quadrature Decoder Control and Status (QDCTRL) | 32 | RW | 0000_0000h |
| 84h | Configuration (CONF) | 32 | RW | 0000_0000h |

## 51.4.1.2 Version ID Register (VERID)

### 51.4.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 51.4.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

#### 51.4.1.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with Filter and Combine registers implemented.<br>0000000000000111b - Standard feature set with Filter, Combine and Quadrature registers implemented. |

### 51.4.1.3 Parameter Register (PARAM)

#### 51.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 51.4.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WIDTH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRIG | | | | | | | | CHAN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

### 51.4.1.3.3 Fields

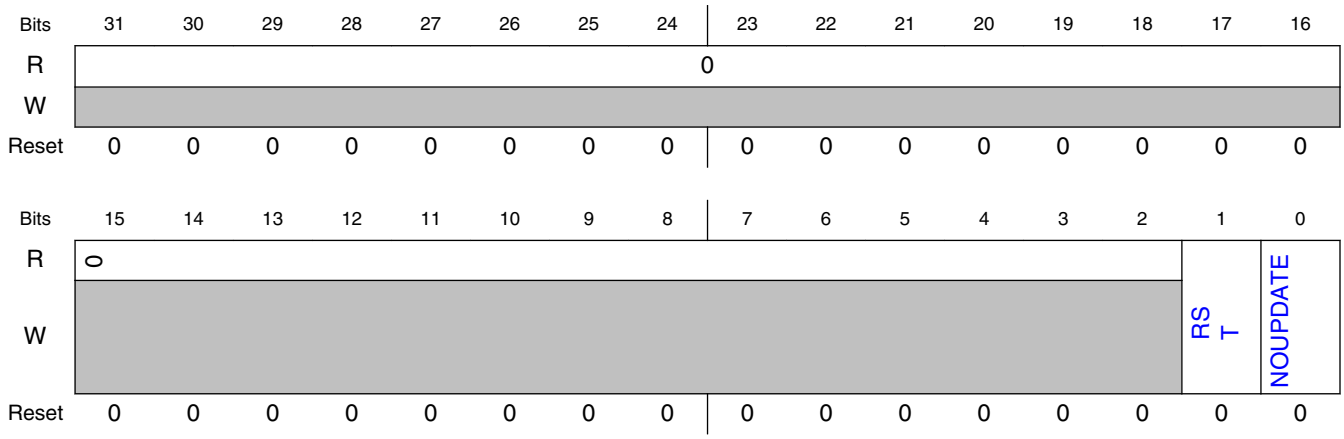
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>WIDTH | Counter Width<br>Width of the counter and timer channels. |
| 15-8<br>TRIG | Trigger Count<br>Number of trigger inputs implemented. |
| 7-0<br>CHAN | Channel Count<br>Number of timer channels implemented. |

## 51.4.1.4 TPM Global Register (GLOBAL)

### 51.4.1.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

#### 51.4.1.4.2 Diagram




#### 51.4.1.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>NOUPDATE | No Update<br>Blocks updates to all internal double buffered registers while it remains set. Software can still update the software visible registers, but changes cannot take effect until this bit is cleared at which point they update as normal.<br>0b - Internal double buffered registers update as normal.<br>1b - Internal double buffered registers do not update. |

### 51.4.1.5 Status and Control (SC)

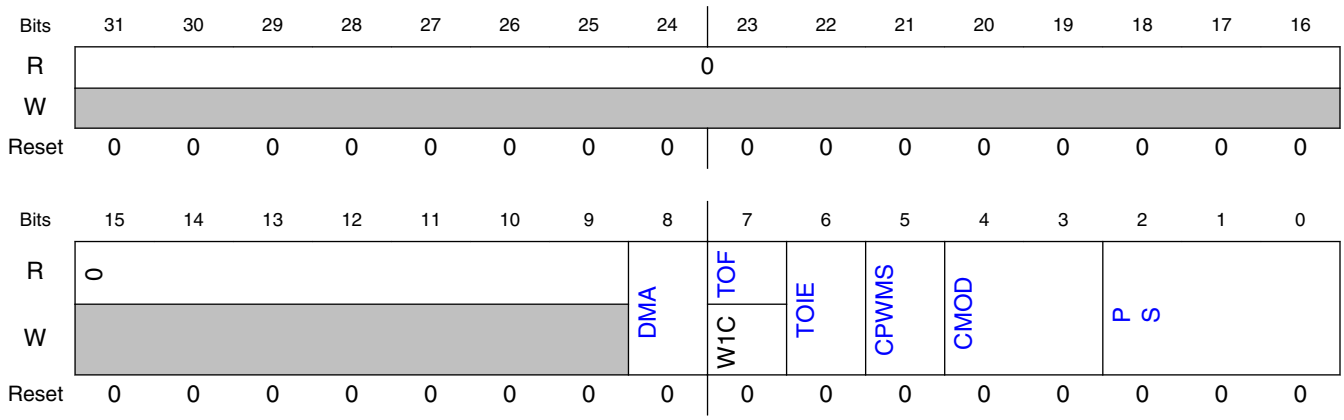
#### 51.4.1.5.1 Offset

| Register | Offset |
|---|---|
| SC | 10h |

### 51.4.1.5.2 Function

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

### 51.4.1.5.3 Diagram




### 51.4.1.5.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>DMA | DMA Enable<br>Enables DMA transfers for the overflow flag.<br>0b - Disables DMA transfers.<br>1b - Enables DMA transfers. |
| 7<br>TOF | Timer Overflow Flag<br>Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect.<br>If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 6<br>TOIE | Timer Overflow Interrupt Enable<br>Enables TPM overflow interrupts.<br>0b - Disable TOF interrupts. Use software polling or DMA request.<br>1b - Enable TOF interrupts. An interrupt is generated when TOF equals one. |
| 5<br>CPWMS | Center-Aligned PWM Select<br>Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode.<br>This field is write protected. It can be written only when the counter is disabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - TPM counter operates in up counting mode.<br>1b - TPM counter operates in up-down counting mode. |
| 4-3<br>CMOD | Clock Mode Selection<br>Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the TPM clock domain.<br>00b - TPM counter is disabled<br>01b - TPM counter increments on every TPM counter clock<br>10b - TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock<br>11b - TPM counter increments on rising edge of the selected external input trigger. |
| 2-0<br>PS | Prescale Factor Selection<br>Selects one of 8 division factors for the clock mode selected by CMOD.<br>This field is write protected. It can be written only when the counter is disabled.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 51.4.1.6 Counter (CNT)

### 51.4.1.6.1 Offset

| Register | Offset |
|---|---|
| CNT | 14h |

### 51.4.1.6.2 Function

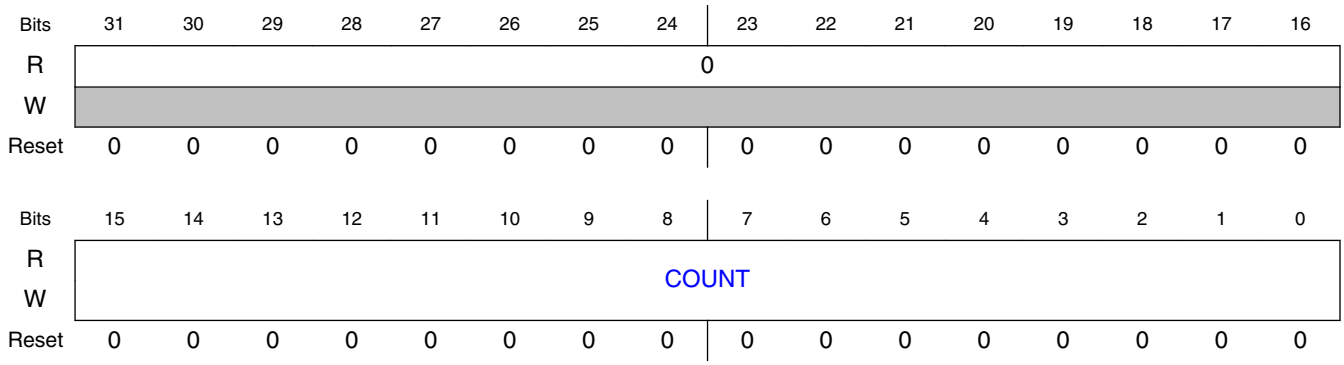
The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT will trigger a reload of the counter, this will force PWM outputs to their reload state..

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

#### 51.4.1.6.3 Diagram




#### 51.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>COUNT | Counter value |

### 51.4.1.7 Modulo (MOD)

#### 51.4.1.7.1 Offset

| Register | Offset |
|---|---|
| MOD | 18h |

#### 51.4.1.7.2 Function

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM counter depends on the selected counting method (see Counter ).

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to MOD Register Update . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write to CNT) before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

#### 51.4.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MOD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 51.4.1.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>MOD | Modulo value<br>This field must be written with single 16-bit or 32-bit access. |

### 51.4.1.8 Capture and Compare Status (STATUS)

#### 51.4.1.8.1 Offset

| Register | Offset |
|---|---|
| STATUS | 1Ch |

#### 51.4.1.8.2 Function

The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

### 51.4.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | TOF | 0 | | CH5F | CH4F | CH3F | CH2F | CH1F | CH0F |
| W | | | | | | | | W1C | | | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 51.4.1.8.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TOF | Timer Overflow Flag<br>See register description<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 7-6<br>— | Reserved |
| 5<br>CH5F | Channel 5 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 4 | Channel 4 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CH4F | See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 3<br>CH3F | Channel 3 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 2<br>CH2F | Channel 2 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 1<br>CH1F | Channel 1 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 0<br>CH0F | Channel 0 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |

## 51.4.1.9 Channel (n) Status and Control (C0SC - C5SC)

### 51.4.1.9.1 Offset

For a = 0 to 5:

| Register | Offset |
|---|---|
| CaSC | 20h + (a × 8h) |

### 51.4.1.9.2 Function

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 51-3. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 00 | None | Channel disabled |
| 0 | 01 | 00 | Software compare | Pin not used for TPM |
| 0 | 00 | 01 | Input capture | Capture on Rising Edge Only |

*Table continues on the next page...*

## Table 51-3. Mode, Edge, and Level Selection (continued)

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | 10 | | Capture on Falling Edge Only |
| | | 11 | | Capture on Rising or Falling Edge |
| | 01 | 01 | Output compare | Toggle Output on match |
| | | 10 | | Clear Output on match |
| | | 11 | | Set Output on match |
| | 10 | 10 | Edge-aligned PWM | High-true pulses (clear Output on counter match, set Output on counter reload, set Output when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match, set Output on counter reload, clear Output when counter first enabled or paused) |
| | | 01 | | Low-true pulses (set Output on counter match, clear Output on counter reload, clear Output when counter first enabled or paused) |
| | | 11 | | Low-true pulses (set Output on counter match, clear Output on counter reload, set Output when counter first enabled or paused) |
| | 11 | 10 | Output compare | Pulse Output low on match |
| | | 01 | | Pulse Output high on match |
| 1 | 10 | 10 | Center-aligned PWM | High-true pulses (clear Output on counter match-up, set Output on counter match-down, set Output on counter reload or when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match-up, set Output on counter match-down, clear Output on |

*Table continues on the next page...*

**Table 51-3. Mode, Edge, and Level Selection (continued)**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | | | counter reload or when counter first enabled or paused)) |
| | | 01 | | Low-true pulses (set Output on match-up, clear Output on match-down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 11 | | Low-true pulses (set Output on match-up, clear Output on match-down, set Output on counter reload or when counter first enabled or paused)) |

## 51.4.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHF | CHIE | MSB | MSA | ELSB | ELSA | 0 | DMA |
| W | | | | | | | | | W1C | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 51.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>CHF | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect.<br>If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 6<br>CHIE | Channel Interrupt Enable<br>Enables channel interrupts.<br>0b - Disable channel interrupts.<br>1b - Enable channel interrupts. |
| 5<br>MSB | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 4<br>MSA | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 3<br>ELSB | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. ph> |
| 2<br>ELSA | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. |
| 1<br>— | Reserved |
| 0<br>DMA | DMA Enable<br>Enables DMA transfers for the channel.<br>0b - Disable DMA transfers.<br>1b - Enable DMA transfers. |

## 51.4.1.10 Channel (n) Value (C0V - C5V)

### 51.4.1.10.1 Offset

For a = 0 to 5:

| Register | Offset |
|---|---|
| CaV | 24h + (a × 8h) |

### 51.4.1.10.2 Function

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to CnV Register Update . Additional writes to the CnV write buffer are ignored until the register has been updated.

### 51.4.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 51.4.1.10.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>VAL | Channel Value<br>Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access. |

## 51.4.1.11 Combine Channel Register (COMBINE)

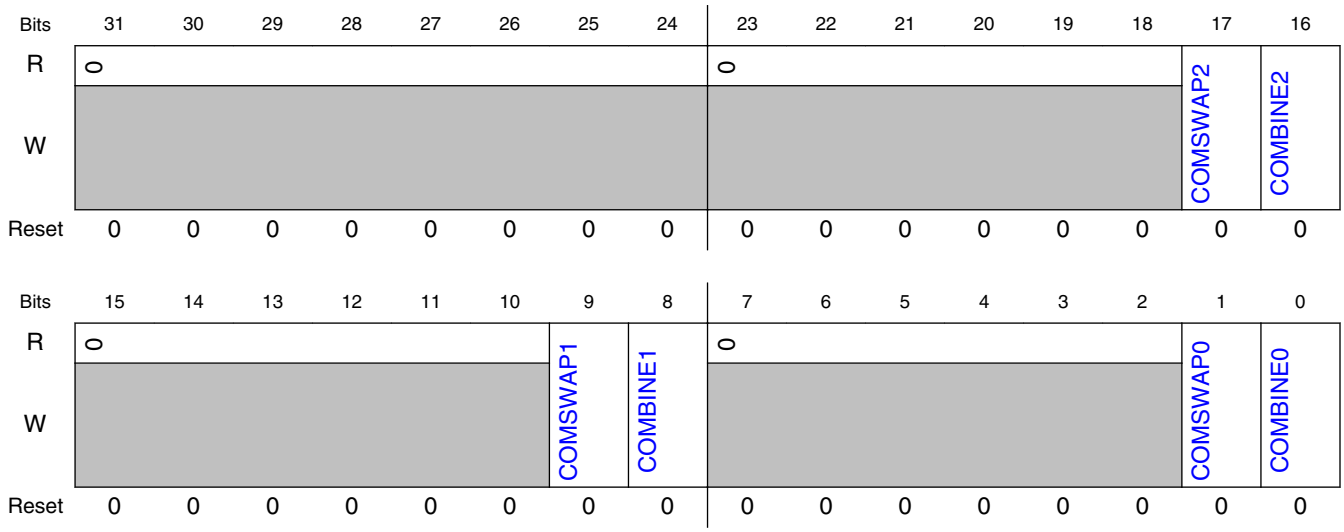
### 51.4.1.11.1 Offset

| Register | Offset |
|---|---|
| COMBINE | 64h |

### 51.4.1.11.2 Function

This register contains the control bits used to configure the combine channel modes for each pair of channels (n) and (n+1), where n is all the even numbered channels.

## 51.4.1.11.3 Diagram




## 51.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>COMSWAP2 | Combine Channels 4 and 5 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 16<br>COMBINE2 | Combine Channels 4 and 5<br>Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 4 and 5 are independent.<br>1b - Channels 4 and 5 are combined. |
| 15-10<br>— | Reserved |
| 9<br>COMSWAP1 | Combine Channels 2 and 3 Swap<br>When set in combine mode, the odd channel is used for the input capture and 1st compare, the even channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 8<br>COMBINE1 | Combine Channels 2 and 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 2 and 3 are independent.<br>1b - Channels 2 and 3 are combined. |
| 7-2<br>— | Reserved |
| 1<br>COMSWAP0 | Combine Channel 0 and 1 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 0<br>COMBINE0 | Combine Channels 0 and 1<br>Enables the combine feature for channels 0 and 1. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 0 and 1 are independent.<br>1b - Channels 0 and 1 are combined. |

## 51.4.1.12 Channel Trigger (TRIG)

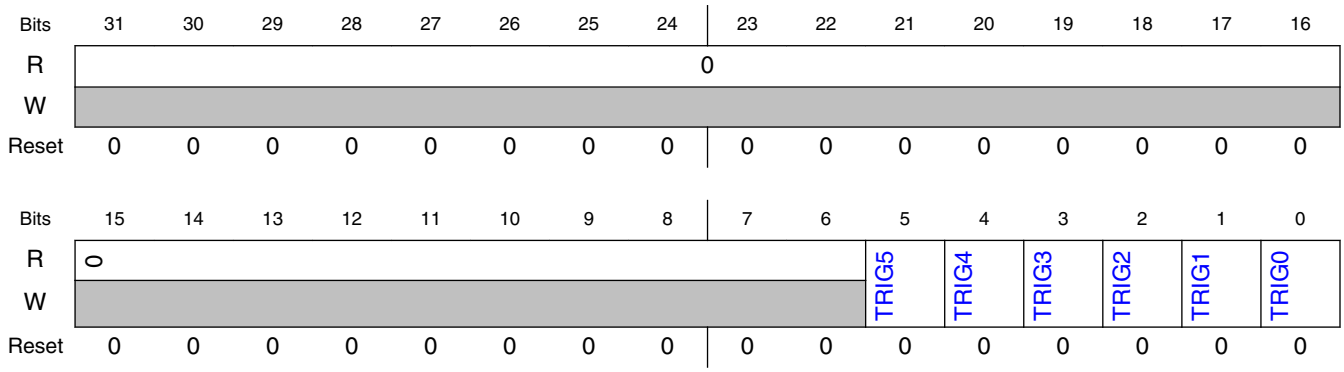
### 51.4.1.12.1 Offset

| Register | Offset |
|---|---|
| TRIG | 6Ch |

### 51.4.1.12.2 Function

In input capture mode, configures the trigger input that is used by the channel to capture the counter value. In output compare or PWM mode, configures the trigger input used to modulate the channel output. When modulating the output, the output is forced to the channel initial value whenever the trigger is not asserted. Note that the even numbered channels share the first input trigger source and the odd numbered channels share the second input trigger source.

### 51.4.1.12.3 Diagram




### 51.4.1.12.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>TRIG5 | Channel 5 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 5. |
| 4<br>TRIG4 | Channel 4 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 4. |
| 3<br>TRIG3 | Channel 3 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 3. |
| 2<br>TRIG2 | Channel 2 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 2. |
| 1<br>TRIG1 | Channel 1 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 1. |
| 0<br>TRIG0 | Channel 0 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 0. |

## 51.4.1.13 Channel Polarity (POL)

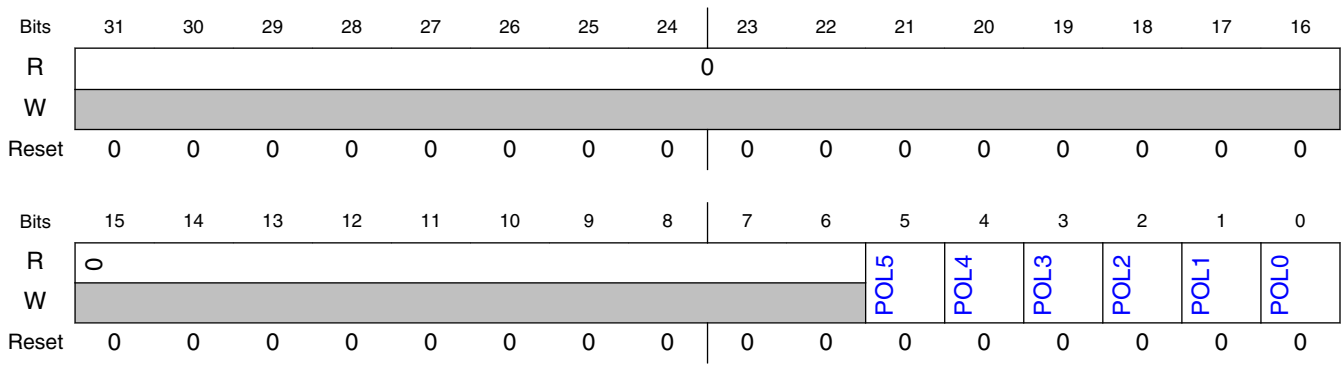
### 51.4.1.13.1 Offset

| Register | Offset |
|---|---|
| POL | 70h |

### 51.4.1.13.2 Function

This register defines the input and output polarity of each of the channels.

### 51.4.1.13.3 Diagram




### 51.4.1.13.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>POL5 | Channel 5 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 4<br>POL4 | Channel 4 Polarity<br>0b - The channel polarity is active high<br>1b - The channel polarity is active low. |
| 3<br>POL3 | Channel 3 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 2<br>POL2 | Channel 2 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 1<br>POL1 | Channel 1 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 0<br>POL0 | Channel 0 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |

# 51.4.1.14 Filter Control (FILTER)

## 51.4.1.14.1 Offset

| Register | Offset |
|---|---|
| FILTER | 78h |

## 51.4.1.14.2 Function

This register selects the filter value of the channel inputs, and an additional output delay value for the channel outputs. In PWM combine modes, the filter can be used to implement deadtime insertion.

## 51.4.1.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CH5FVAL | | | | CH4FVAL | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH3FVAL | | | | CH2FVAL | | | | CH1FVAL | | | | CH0FVAL | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 51.4.1.14.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20<br>CH5FVAL | Channel 5 Filter Value<br>Sets the filter value for the channel 5 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH5FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 5 in PWM modes. Deadtime insertion is disabled when CH5FVAL is zero, otherwise deadtime insertion for channel 5 is configured as (CH5FVAL * 4) clock cycles. |
| 19-16<br>CH4FVAL | Channel 4 Filter Value<br>Sets the filter value for the channel 4 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH4FVAL * 4) clock cycles. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Sets the deadtime insertion time for channel 4 in PWM modes. Deadtime insertion is disabled when CH4FVAL is zero, otherwise deadtime insertion for channel 4 is configured as (CH4FVAL * 4) clock cycles. |
| 15-12<br>CH3FVAL | Channel 3 Filter Value<br>Sets the filter value for the channel 3 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH3FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 3 in PWM modes. Deadtime insertion is disabled when CH3FVAL is zero, otherwise deadtime insertion for channel 3 is configured as (CH3FVAL * 4) clock cycles. |
| 11-8<br>CH2FVAL | Channel 2 Filter Value<br>Sets the filter value for the channel 2 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH2FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 2 in PWM modes. Deadtime insertion is disabled when CH2FVAL is zero, otherwise deadtime insertion for channel 2 is configured as (CH2FVAL * 4) clock cycles. |
| 7-4<br>CH1FVAL | Channel 1 Filter Value<br>Sets the filter value for the channel 1 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH1FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 1 in PWM modes. Deadtime insertion is disabled when CH1FVAL is zero, otherwise deadtime insertion for channel 1 is configured as (CH1FVAL * 4) clock cycles. |
| 3-0<br>CH0FVAL | Channel 0 Filter Value<br>Sets the filter value for the channel 0 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH0FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 0 in PWM modes. Deadtime insertion is disabled when CH0FVAL is zero, otherwise deadtime insertion for channel 0 is configured as (CH0FVAL * 4) clock cycles. |

## 51.4.1.15 Quadrature Decoder Control and Status (QDCTRL)

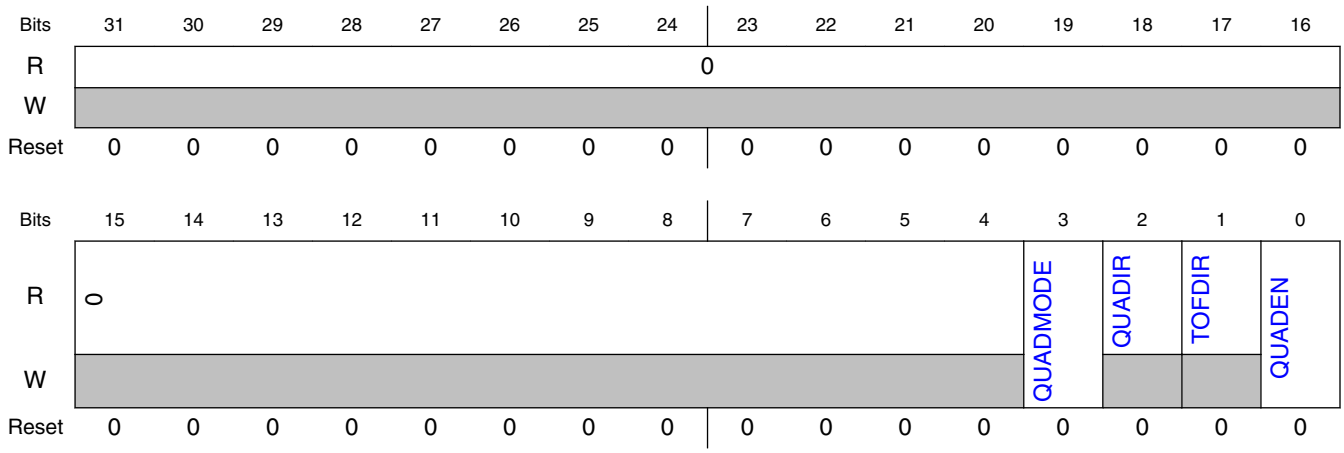
### 51.4.1.15.1 Offset

| Register | Offset |
|---|---|
| QDCTRL | 80h |

### 51.4.1.15.2 Function

This register has the control and status bits for the quadrature decoder mode.

## 51.4.1.15.3 Diagram




## 51.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>QUADMODE | Quadrature Decoder Mode<br>Selects the encoding mode used in the quadrature decoder mode.<br>0b - Phase encoding mode.<br>1b - Count and direction encoding mode. |
| 2<br>QUADIR | Counter Direction in Quadrature Decode Mode<br>Indicates the counting direction.<br>0b - Counter direction is decreasing (counter decrement).<br>1b - Counter direction is increasing (counter increment). |
| 1<br>TOFDIR | TOFDIR<br>Indicates if the TOF bit was set on the top or the bottom of counting.<br>0b - TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (zero) to its maximum value (MOD register).<br>1b - TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (zero). |
| 0<br>QUADEN | QUADEN<br>Enables the quadrature decoder mode. In this mode, the channel 0 and channel 1 inputs control the TPM counter direction and can only be used for software compare. The quadrature decoder mode has precedence over the other modes.<br>0b - Quadrature decoder mode is disabled.<br>1b - Quadrature decoder mode is enabled. |

## 51.4.1.16 Configuration (CONF)

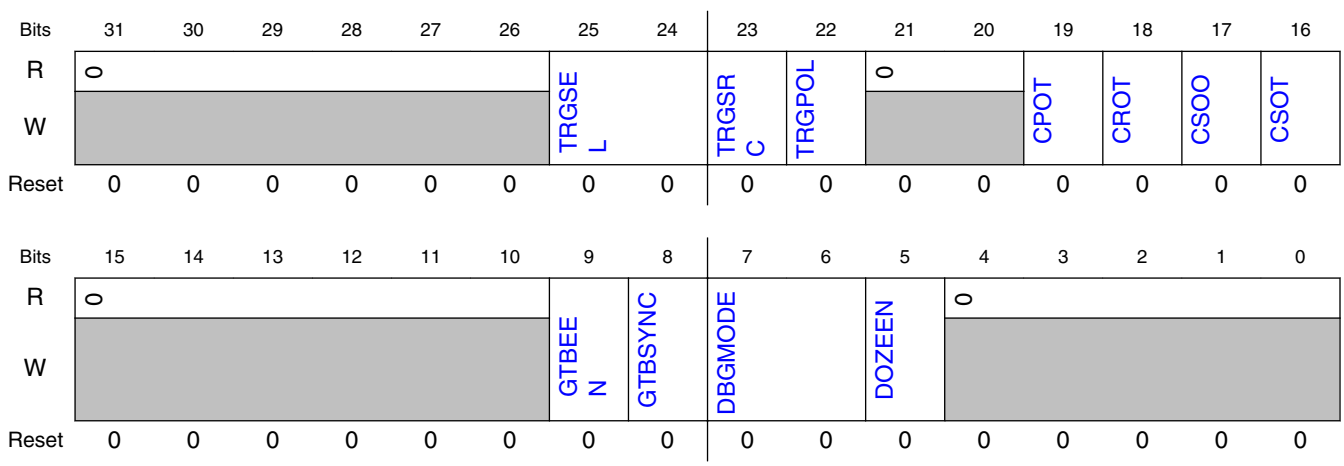
### 51.4.1.16.1 Offset

| Register | Offset |
|---|---|
| CONF | 84h |

### 51.4.1.16.2 Function

This register selects the behavior in debug and wait modes and the use of an external global time base.

### 51.4.1.16.3 Diagram




### 51.4.1.16.4 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>TRGSEL | Trigger Select<br>Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled.<br>Refer to chip configuration section for available external trigger options.<br>The available internal trigger sources are listed below.<br>01b - Channel 0 pin input capture<br>10b - Channel 1 pin input capture |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 11b - Channel 0 or Channel 1 pin input capture |
| 23<br>TRGSRC | Trigger Source<br>Selects between internal (channel pin input capture) or external trigger sources.<br>When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins.<br>This field should only be changed when the TPM counter is disabled.<br>0b - Trigger source selected by TRGSEL is external.<br>1b - Trigger source selected by TRGSEL is internal (channel pin input capture). |
| 22<br>TRGPOL | Trigger Polarity<br>Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled.<br>0b - Trigger is active high.<br>1b - Trigger is active low. |
| 21-20<br>— | Reserved |
| 19<br>CPOT | Counter Pause On Trigger<br>When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. |
| 18<br>CROT | Counter Reload On Trigger<br>When set, the TPM counter will reload with zero (and set PWM outputs to their reload state) when a rising edge is detected on the selected trigger input.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - Counter is not reloaded due to a rising edge on the selected input trigger<br>1b - Counter is reloaded when a rising edge is detected on the selected input trigger |
| 17<br>CSOO | Counter Stop On Overflow<br>When set, the TPM counter will pause incrementing once the counter equals the MOD value and increments (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT is set. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter continues incrementing or decrementing after overflow<br>1b - TPM counter stops incrementing or decrementing after overflow. |
| 16<br>CSOT | Counter Start on Trigger<br>When set, the TPM counter will pause incrementing when it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is paused due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter starts to increment immediately, once it is enabled.<br>1b - TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow. |
| 15-10<br>— | Reserved |
| 9<br>GTBEEN | Global time base enable<br>Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag.<br>0b - All channels use the internally generated TPM counter as their timebase<br>1b - All channels use an externally generated global timebase as their timebase |
| 8<br>GTBSYNC | Global Time Base Synchronization<br>When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger, paused state and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled.<br>0b - Global timebase synchronization disabled.<br>1b - Global timebase synchronization enabled. |
| 7-6<br>DBGMODE | Debug Mode<br>Configures the TPM behavior in debug mode. All other configurations are reserved.<br>00b - TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state.<br>11b - TPM counter continues in debug mode. |
| 5<br>DOZEEN | Doze Enable<br>Configures the TPM behavior in wait mode.<br>0b - Internal TPM counter continues in Doze mode.<br>1b - Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state. |
| 4-0<br>— | Reserved |

## 51.4.2 TPM1 register descriptions

### 51.4.2.1 TPM1 Memory map

TPM1 base address: 4003_6000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0600_0007h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0010_0402h |
| 8h | TPM Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| 10h | Status and Control (SC) | 32 | RW | 0000_0000h |
| 14h | Counter (CNT) | 32 | RW | 0000_0000h |
| 18h | Modulo (MOD) | 32 | RW | 0000_FFFFh |
| 1Ch | Capture and Compare Status (STATUS) | 32 | W1C | 0000_0000h |
| 20h | Channel (n) Status and Control (C0SC) | 32 | RW | 0000_0000h |
| 24h | Channel (n) Value (C0V) | 32 | RW | 0000_0000h |
| 28h | Channel (n) Status and Control (C1SC) | 32 | RW | 0000_0000h |
| 2Ch | Channel (n) Value (C1V) | 32 | RW | 0000_0000h |
| 64h | Combine Channel Register (COMBINE) | 32 | RW | 0000_0000h |
| 6Ch | Channel Trigger (TRIG) | 32 | RW | 0000_0000h |
| 70h | Channel Polarity (POL) | 32 | RW | 0000_0000h |
| 78h | Filter Control (FILTER) | 32 | RW | 0000_0000h |
| 80h | Quadrature Decoder Control and Status (QDCTRL) | 32 | RW | 0000_0000h |
| 84h | Configuration (CONF) | 32 | RW | 0000_0000h |

## 51.4.2.2 Version ID Register (VERID)

### 51.4.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 51.4.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

#### 51.4.2.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with Filter and Combine registers implemented.<br>0000000000000111b - Standard feature set with Filter, Combine and Quadrature registers implemented. |

### 51.4.2.3 Parameter Register (PARAM)

#### 51.4.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 51.4.2.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WIDTH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRIG | | | | | | | | CHAN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 51.4.2.3.3 Fields

| Field | Function |
|---|---|
| 31-24 | Reserved |

*Table continues on the next page...*

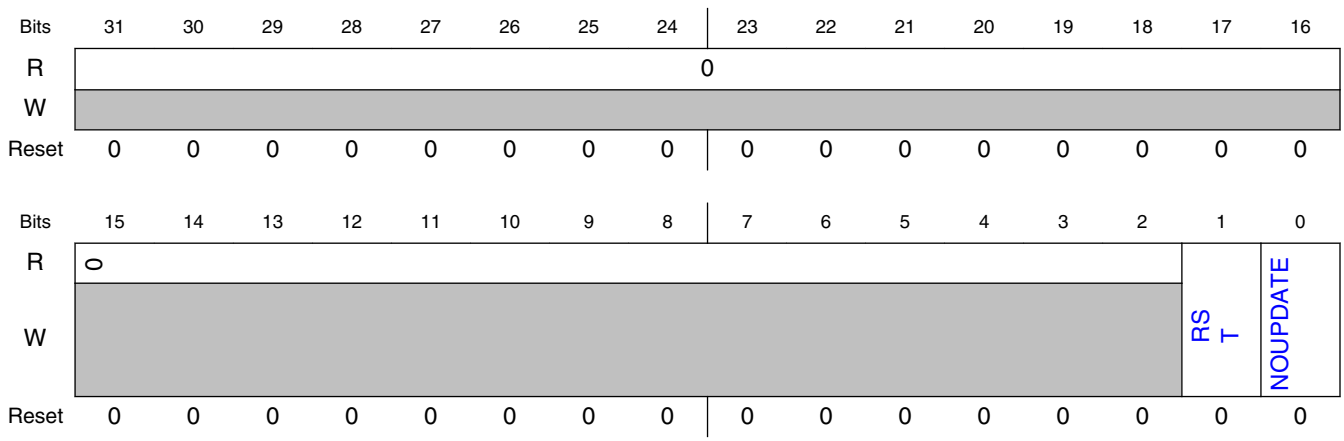
| Field | Function |
|---|---|
| — | |
| 23-16<br>WIDTH | Counter Width<br>Width of the counter and timer channels. |
| 15-8<br>TRIG | Trigger Count<br>Number of trigger inputs implemented. |
| 7-0<br>CHAN | Channel Count<br>Number of timer channels implemented. |

## 51.4.2.4 TPM Global Register (GLOBAL)

### 51.4.2.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 51.4.2.4.2 Diagram




### 51.4.2.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>NOUPDATE | No Update<br>Blocks updates to all internal double buffered registers while it remains set. Software can still update the software visible registers, but changes cannot take effect until this bit is cleared at which point they update as normal.<br>0b - Internal double buffered registers update as normal.<br>1b - Internal double buffered registers do not update. |

## 51.4.2.5 Status and Control (SC)

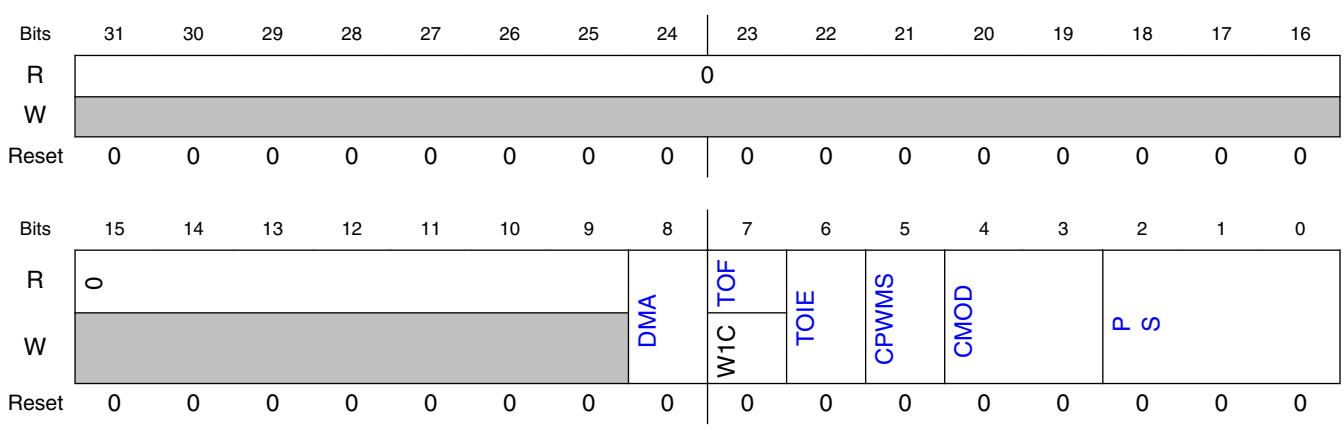
### 51.4.2.5.1 Offset

| Register | Offset |
|---|---|
| SC | 10h |

### 51.4.2.5.2 Function

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

### 51.4.2.5.3 Diagram



## 51.4.2.5.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>DMA | DMA Enable<br>Enables DMA transfers for the overflow flag.<br>0b - Disables DMA transfers.<br>1b - Enables DMA transfers. |
| 7<br>TOF | Timer Overflow Flag<br>Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect.<br>If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 6<br>TOIE | Timer Overflow Interrupt Enable<br>Enables TPM overflow interrupts.<br>0b - Disable TOF interrupts. Use software polling or DMA request.<br>1b - Enable TOF interrupts. An interrupt is generated when TOF equals one. |
| 5<br>CPWMS | Center-Aligned PWM Select<br>Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode.<br>This field is write protected. It can be written only when the counter is disabled.<br>0b - TPM counter operates in up counting mode.<br>1b - TPM counter operates in up-down counting mode. |
| 4-3<br>CMOD | Clock Mode Selection<br>Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the TPM clock domain.<br>00b - TPM counter is disabled<br>01b - TPM counter increments on every TPM counter clock<br>10b - TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock<br>11b - TPM counter increments on rising edge of the selected external input trigger. |
| 2-0<br>PS | Prescale Factor Selection<br>Selects one of 8 division factors for the clock mode selected by CMOD.<br>This field is write protected. It can be written only when the counter is disabled.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 51.4.2.6 Counter (CNT)

### 51.4.2.6.1 Offset

| Register | Offset |
|---|---|
| CNT | 14h |

### 51.4.2.6.2 Function

The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT will trigger a reload of the counter, this will force PWM outputs to their reload state..

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

### 51.4.2.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COUNT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 51.4.2.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>COUNT | Counter value |

## 51.4.2.7 Modulo (MOD)

### 51.4.2.7.1 Offset

| Register | Offset |
|---|---|
| MOD | 18h |

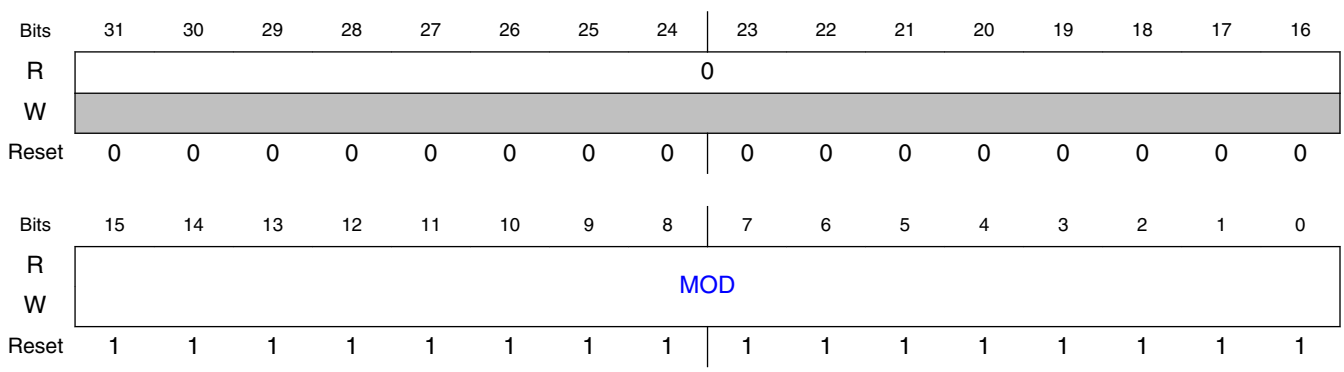
### 51.4.2.7.2 Function

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM counter depends on the selected counting method (see Counter ).

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to MOD Register Update . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write to CNT) before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

### 51.4.2.7.3 Diagram




### 51.4.2.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0 | Modulo value |

| Field | Function |
|---|---|
| MOD | This field must be written with single 16-bit or 32-bit access. |

## 51.4.2.8 Capture and Compare Status (STATUS)

### 51.4.2.8.1 Offset

| Register | Offset |
|---|---|
| STATUS | 1Ch |

### 51.4.2.8.2 Function

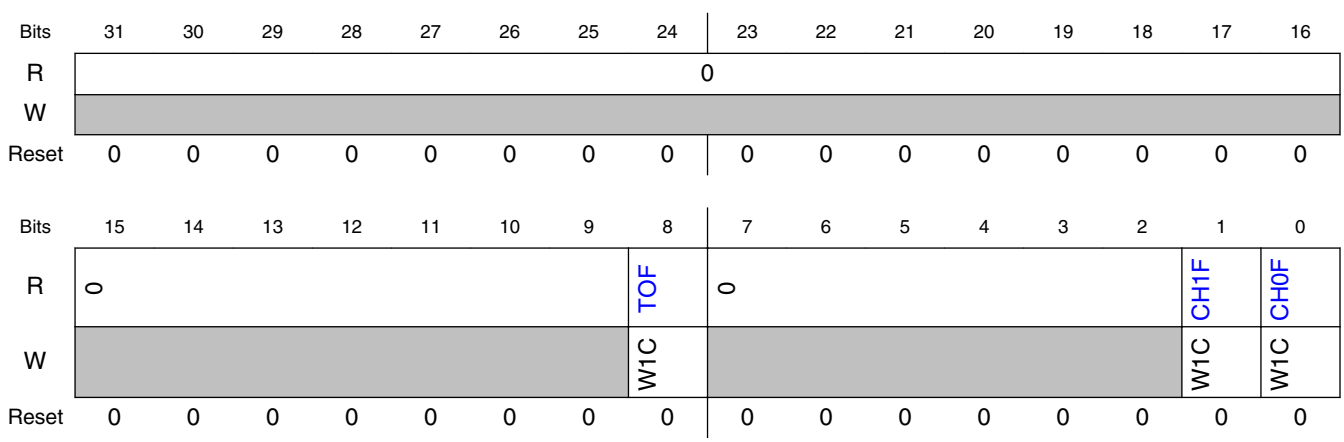
The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

### 51.4.2.8.3 Diagram



#### 51.4.2.8.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TOF | Timer Overflow Flag<br>See register description<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 7-2<br>— | Reserved |
| 1<br>CH1F | Channel 1 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 0<br>CH0F | Channel 0 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |

### 51.4.2.9 Channel (n) Status and Control (C0SC - C1SC)

#### 51.4.2.9.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| CaSC | 20h + (a × 8h) |

#### 51.4.2.9.2 Function

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 51-4. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 00 | None | Channel disabled |
| 0 | 01 | 00 | Software compare | Pin not used for TPM |

*Table continues on the next page...*

## Table 51-4. Mode, Edge, and Level Selection (continued)

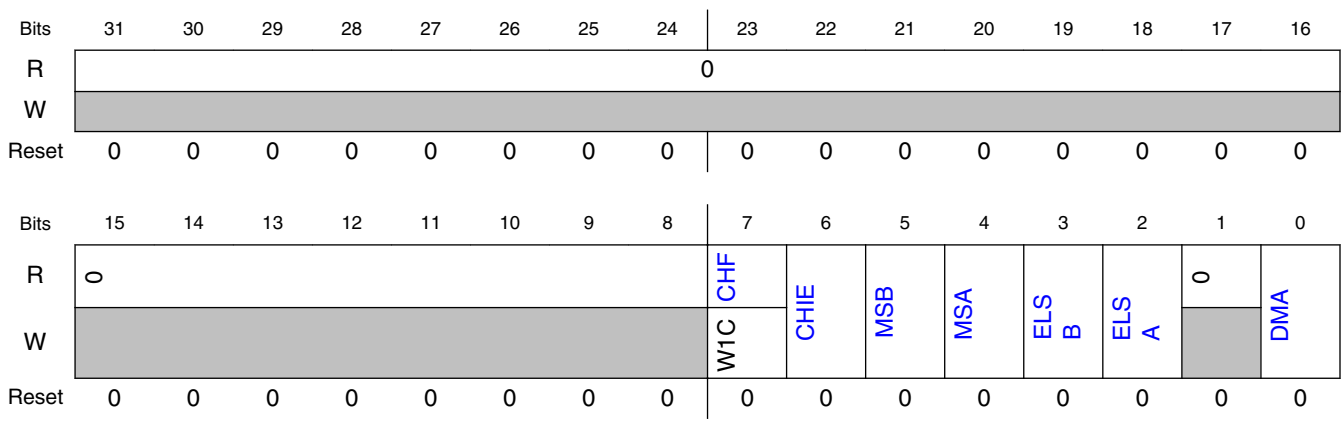
| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 01 | Input capture | Capture on Rising Edge Only |
| | | 10 | | Capture on Falling Edge Only |
| | | 11 | | Capture on Rising or Falling Edge |
| | 01 | 01 | Output compare | Toggle Output on match |
| | | 10 | | Clear Output on match |
| | | 11 | | Set Output on match |
| | 10 | 10 | Edge-aligned PWM | High-true pulses (clear Output on counter match, set Output on counter reload, set Output when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match, set Output on counter reload, clear Output when counter first enabled or paused) |
| | | 01 | | Low-true pulses (set Output on counter match, clear Output on counter reload, clear Output when counter first enabled or paused) |
| | | 11 | | Low-true pulses (set Output on counter match, clear Output on counter reload, set Output when counter first enabled or paused) |
| | 11 | 10 | Output compare | Pulse Output low on match |
| | | 01 | | Pulse Output high on match |
| 1 | 10 | 10 | Center-aligned PWM | High-true pulses (clear Output on counter match-up, set Output on counter match-down, set Output on counter reload or when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match-up, set Output on counter match- |

*Table continues on the next page...*

**Table 51-4. Mode, Edge, and Level Selection (continued)**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | | | down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 01 | | Low-true pulses (set Output on match-up, clear Output on match-down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 11 | | Low-true pulses (set Output on match-up, clear Output on match-down, set Output on counter reload or when counter first enabled or paused)) |

## 51.4.2.9.3 Diagram




## 51.4.2.9.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>CHF | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 6<br>CHIE | Channel Interrupt Enable<br>Enables channel interrupts.<br>0b - Disable channel interrupts.<br>1b - Enable channel interrupts. |
| 5<br>MSB | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 4<br>MSA | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 3<br>ELSB | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. ph> |
| 2<br>ELSA | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. |
| 1<br>— | Reserved |
| 0<br>DMA | DMA Enable<br>Enables DMA transfers for the channel.<br>0b - Disable DMA transfers.<br>1b - Enable DMA transfers. |

## 51.4.2.10 Channel (n) Value (C0V - C1V)

### 51.4.2.10.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| CaV | 24h + (a × 8h) |

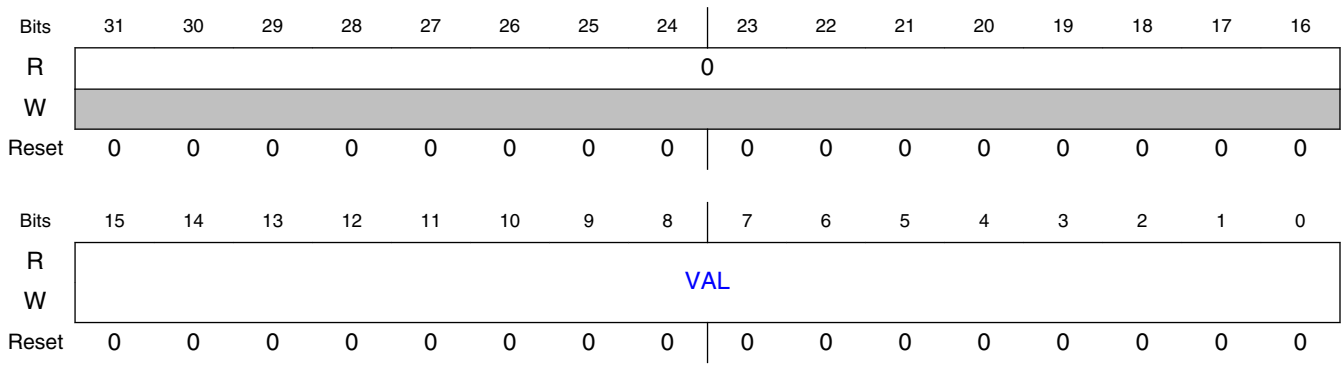
### 51.4.2.10.2 Function

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to CnV Register Update . Additional writes to the CnV write buffer are ignored until the register has been updated.

### 51.4.2.10.3 Diagram




### 51.4.2.10.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>VAL | Channel Value<br>Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access. |

## 51.4.2.11 Combine Channel Register (COMBINE)

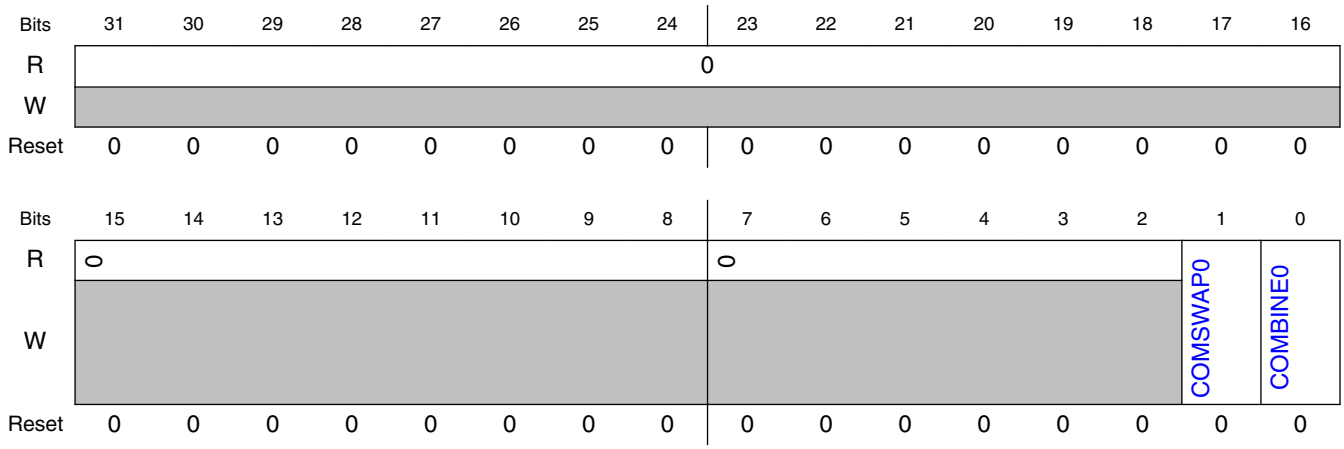
### 51.4.2.11.1 Offset

| Register | Offset |
|---|---|
| COMBINE | 64h |

### 51.4.2.11.2 Function

This register contains the control bits used to configure the combine channel modes for each pair of channels (n) and (n+1), where n is all the even numbered channels.

### 51.4.2.11.3 Diagram




### 51.4.2.11.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1<br>COMSWAP0 | Combine Channel 0 and 1 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 0<br>COMBINE0 | Combine Channels 0 and 1<br>Enables the combine feature for channels 0 and 1. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 0 and 1 are independent.<br>1b - Channels 0 and 1 are combined. |

## 51.4.2.12 Channel Trigger (TRIG)

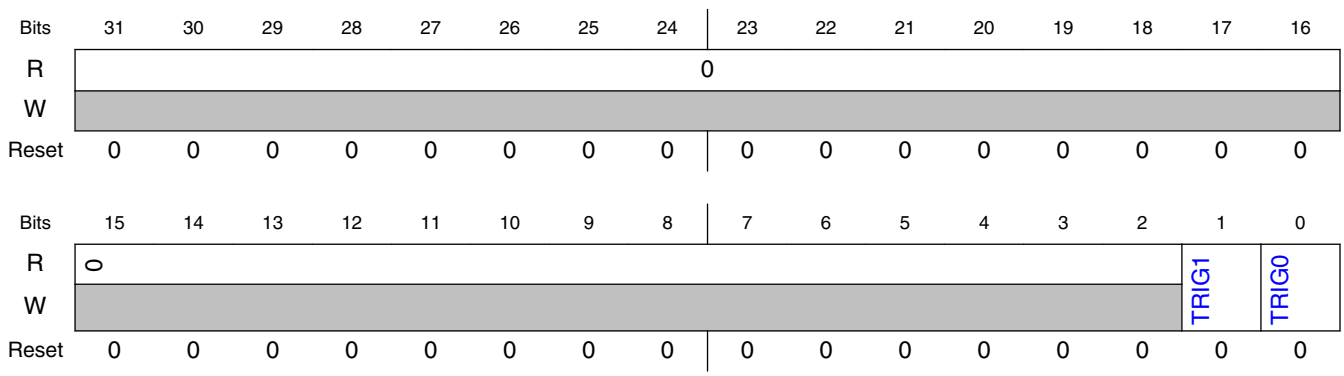
### 51.4.2.12.1 Offset

| Register | Offset |
|---|---|
| TRIG | 6Ch |

### 51.4.2.12.2 Function

In input capture mode, configures the trigger input that is used by the channel to capture the counter value. In output compare or PWM mode, configures the trigger input used to modulate the channel output. When modulating the output, the output is forced to the channel initial value whenever the trigger is not asserted. Note that the even numbered channels share the first input trigger source and the odd numbered channels share the second input trigger source.

### 51.4.2.12.3 Diagram




### 51.4.2.12.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>TRIG1 | Channel 1 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 1. |
| 0<br>TRIG0 | Channel 0 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 0. |

## 51.4.2.13 Channel Polarity (POL)

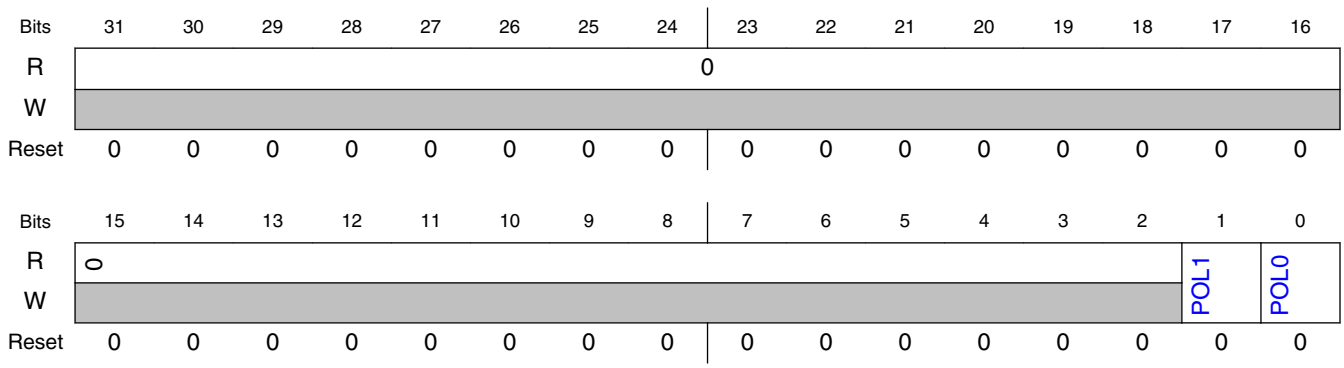
### 51.4.2.13.1 Offset

| Register | Offset |
|---|---|
| POL | 70h |

### 51.4.2.13.2 Function

This register defines the input and output polarity of each of the channels.

### 51.4.2.13.3 Diagram




### 51.4.2.13.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>POL1 | Channel 1 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 0<br>POL0 | Channel 0 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |

## 51.4.2.14 Filter Control (FILTER)

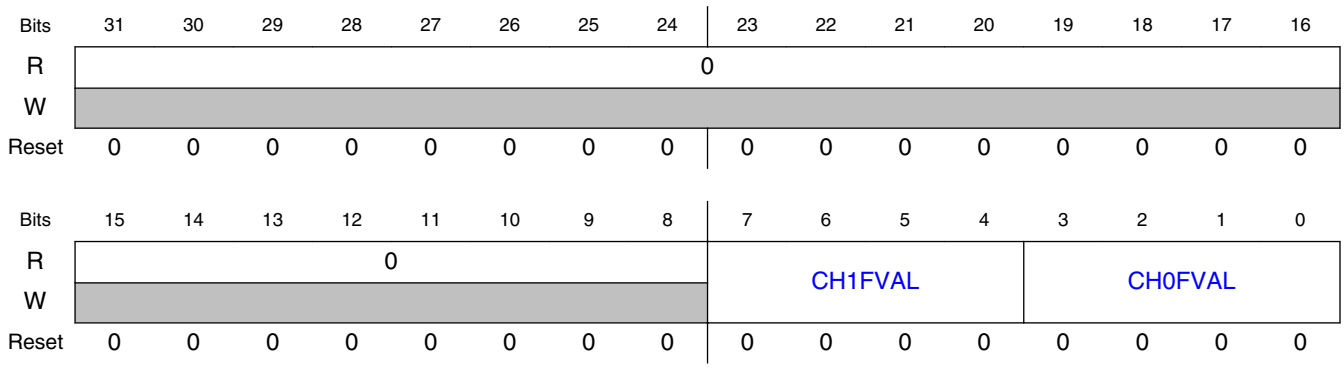
### 51.4.2.14.1 Offset

| Register | Offset |
|---|---|
| FILTER | 78h |

### 51.4.2.14.2 Function

This register selects the filter value of the channel inputs, and an additional output delay value for the channel outputs. In PWM combine modes, the filter can be used to implement deadtime insertion.

### 51.4.2.14.3 Diagram




### 51.4.2.14.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-4<br>CH1FVAL | Channel 1 Filter Value<br>Sets the filter value for the channel 1 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH1FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 1 in PWM modes. Deadtime insertion is disabled when CH1FVAL is zero, otherwise deadtime insertion for channel 1 is configured as (CH1FVAL * 4) clock cycles. |
| 3-0<br>CH0FVAL | Channel 0 Filter Value<br>Sets the filter value for the channel 0 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH0FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 0 in PWM modes. Deadtime insertion is disabled when CH0FVAL is zero, otherwise deadtime insertion for channel 0 is configured as (CH0FVAL * 4) clock cycles. |

## 51.4.2.15 Quadrature Decoder Control and Status (QDCTRL)

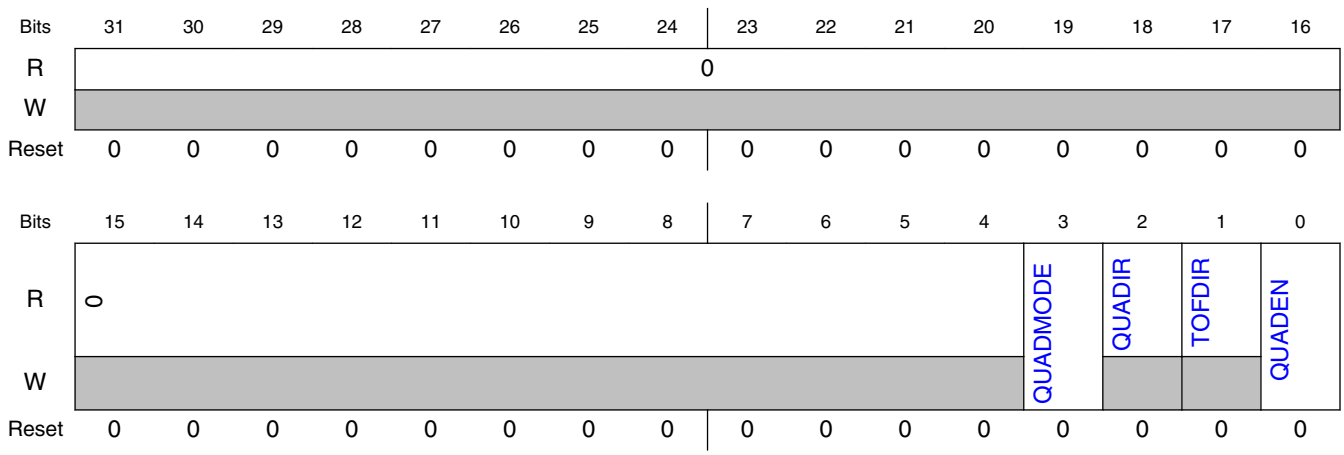
### 51.4.2.15.1 Offset

| Register | Offset |
|---|---|
| QDCTRL | 80h |

### 51.4.2.15.2 Function

This register has the control and status bits for the quadrature decoder mode.

### 51.4.2.15.3 Diagram




### 51.4.2.15.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>QUADMODE | Quadrature Decoder Mode<br>Selects the encoding mode used in the quadrature decoder mode.<br>0b - Phase encoding mode.<br>1b - Count and direction encoding mode. |
| 2<br>QUADIR | Counter Direction in Quadrature Decode Mode<br>Indicates the counting direction.<br>0b - Counter direction is decreasing (counter decrement).<br>1b - Counter direction is increasing (counter increment). |
| 1<br>TOFDIR | TOFDIR<br>Indicates if the TOF bit was set on the top or the bottom of counting.<br>0b - TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (zero) to its maximum value (MOD register).<br>1b - TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (zero). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>QUADEN | QUADEN<br>Enables the quadrature decoder mode. In this mode, the channel 0 and channel 1 inputs control the TPM counter direction and can only be used for software compare. The quadrature decoder mode has precedence over the other modes.<br>0b - Quadrature decoder mode is disabled.<br>1b - Quadrature decoder mode is enabled. |

## 51.4.2.16 Configuration (CONF)

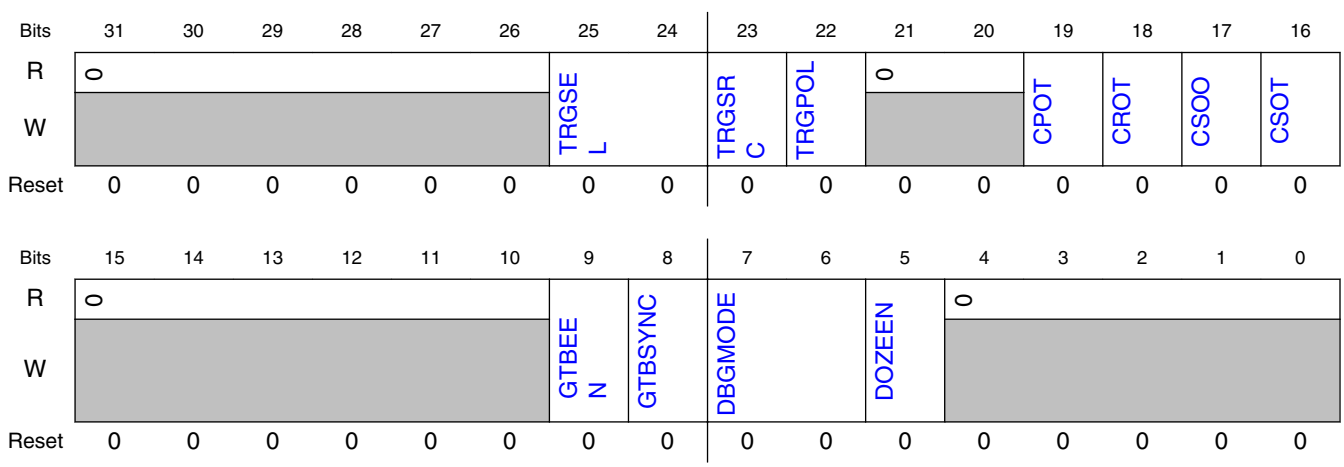
### 51.4.2.16.1 Offset

| Register | Offset |
|---|---|
| CONF | 84h |

### 51.4.2.16.2 Function

This register selects the behavior in debug and wait modes and the use of an external global time base.

### 51.4.2.16.3 Diagram




### 51.4.2.16.4 Fields

| Field | Function |
|---|---|
| 31-26 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 25-24<br>TRGSEL | Trigger Select<br>Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled.<br>Refer to chip configuration section for available external trigger options.<br>The available internal trigger sources are listed below.<br>01b - Channel 0 pin input capture<br>10b - Channel 1 pin input capture<br>11b - Channel 0 or Channel 1 pin input capture |
| 23<br>TRGSRC | Trigger Source<br>Selects between internal (channel pin input capture) or external trigger sources.<br>When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins.<br>This field should only be changed when the TPM counter is disabled.<br>0b - Trigger source selected by TRGSEL is external.<br>1b - Trigger source selected by TRGSEL is internal (channel pin input capture). |
| 22<br>TRGPOL | Trigger Polarity<br>Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled.<br>0b - Trigger is active high.<br>1b - Trigger is active low. |
| 21-20<br>— | Reserved |
| 19<br>CPOT | Counter Pause On Trigger<br>When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. |
| 18<br>CROT | Counter Reload On Trigger<br>When set, the TPM counter will reload with zero (and set PWM outputs to their reload state) when a rising edge is detected on the selected trigger input.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - Counter is not reloaded due to a rising edge on the selected input trigger<br>1b - Counter is reloaded when a rising edge is detected on the selected input trigger |
| 17<br>CSOO | Counter Stop On Overflow<br>When set, the TPM counter will pause incrementing once the counter equals the MOD value and increments (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT is set. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter continues incrementing or decrementing after overflow<br>1b - TPM counter stops incrementing or decrementing after overflow. |
| 16<br>CSOT | Counter Start on Trigger<br>When set, the TPM counter will pause incrementing when it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is paused due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter starts to increment immediately, once it is enabled.<br>1b - TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow. |
| 15-10<br>— | Reserved |
| 9<br>GTBEEN | Global time base enable<br>Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag.<br>0b - All channels use the internally generated TPM counter as their timebase<br>1b - All channels use an externally generated global timebase as their timebase |
| 8<br>GTBSYNC | Global Time Base Synchronization<br>When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger, paused state and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled.<br>0b - Global timebase synchronization disabled.<br>1b - Global timebase synchronization enabled. |
| 7-6<br>DBGMODE | Debug Mode<br>Configures the TPM behavior in debug mode. All other configurations are reserved.<br>00b - TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state.<br>11b - TPM counter continues in debug mode. |
| 5<br>DOZEEN | Doze Enable<br>Configures the TPM behavior in wait mode.<br>0b - Internal TPM counter continues in Doze mode.<br>1b - Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state. |
| 4-0<br>— | Reserved |

## 51.4.3 TPM2 register descriptions

### 51.4.3.1 TPM2 Memory map

TPM2 base address: 4003_7000h

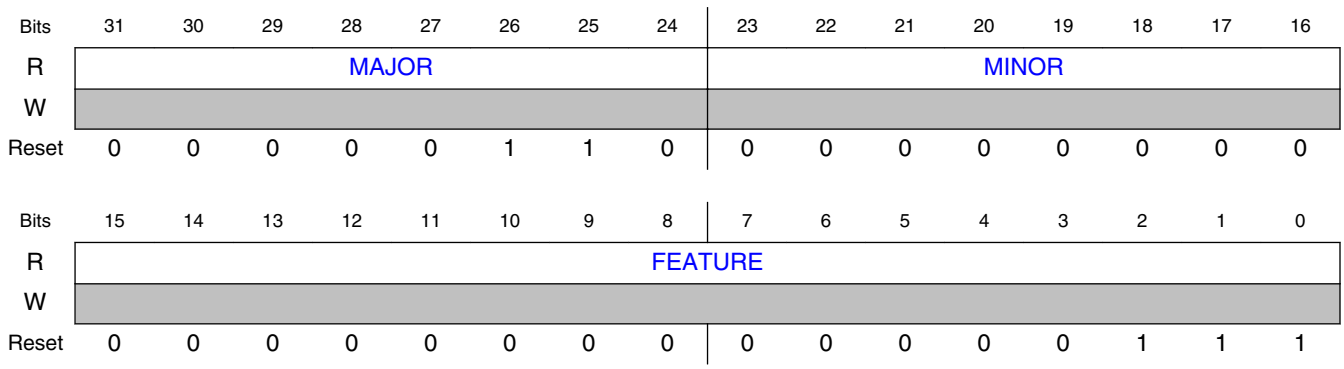
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0600_0007h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0010_0406h |
| 8h | TPM Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| 10h | Status and Control (SC) | 32 | RW | 0000_0000h |
| 14h | Counter (CNT) | 32 | RW | 0000_0000h |
| 18h | Modulo (MOD) | 32 | RW | 0000_FFFFh |
| 1Ch | Capture and Compare Status (STATUS) | 32 | W1C | 0000_0000h |
| 20h | Channel (n) Status and Control (C0SC) | 32 | RW | 0000_0000h |
| 24h | Channel (n) Value (C0V) | 32 | RW | 0000_0000h |
| 28h | Channel (n) Status and Control (C1SC) | 32 | RW | 0000_0000h |
| 2Ch | Channel (n) Value (C1V) | 32 | RW | 0000_0000h |
| 30h | Channel (n) Status and Control (C2SC) | 32 | RW | 0000_0000h |
| 34h | Channel (n) Value (C2V) | 32 | RW | 0000_0000h |
| 38h | Channel (n) Status and Control (C3SC) | 32 | RW | 0000_0000h |
| 3Ch | Channel (n) Value (C3V) | 32 | RW | 0000_0000h |
| 40h | Channel (n) Status and Control (C4SC) | 32 | RW | 0000_0000h |
| 44h | Channel (n) Value (C4V) | 32 | RW | 0000_0000h |
| 48h | Channel (n) Status and Control (C5SC) | 32 | RW | 0000_0000h |
| 4Ch | Channel (n) Value (C5V) | 32 | RW | 0000_0000h |
| 64h | Combine Channel Register (COMBINE) | 32 | RW | 0000_0000h |
| 6Ch | Channel Trigger (TRIG) | 32 | RW | 0000_0000h |
| 70h | Channel Polarity (POL) | 32 | RW | 0000_0000h |
| 78h | Filter Control (FILTER) | 32 | RW | 0000_0000h |
| 80h | Quadrature Decoder Control and Status (QDCTRL) | 32 | RW | 0000_0000h |
| 84h | Configuration (CONF) | 32 | RW | 0000_0000h |

### 51.4.3.2 Version ID Register (VERID)

#### 51.4.3.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 51.4.3.2.2 Diagram




#### 51.4.3.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with Filter and Combine registers implemented.<br>0000000000000111b - Standard feature set with Filter, Combine and Quadrature registers implemented. |

### 51.4.3.3 Parameter Register (PARAM)

#### 51.4.3.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 51.4.3.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WIDTH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRIG | | | | | | | | CHAN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

#### 51.4.3.3.3 Fields

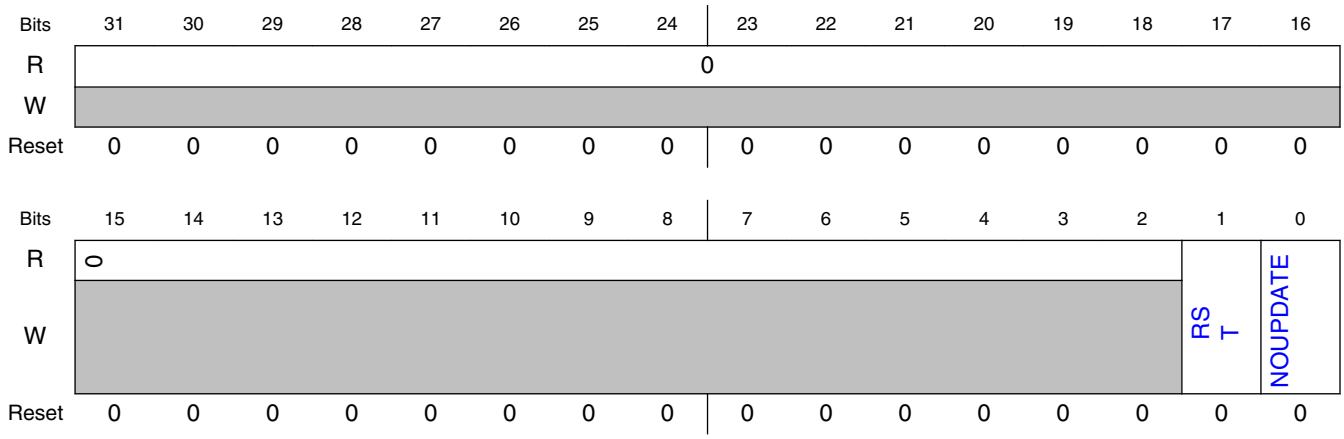
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>WIDTH | Counter Width<br>Width of the counter and timer channels. |
| 15-8<br>TRIG | Trigger Count<br>Number of trigger inputs implemented. |
| 7-0<br>CHAN | Channel Count<br>Number of timer channels implemented. |

### 51.4.3.4 TPM Global Register (GLOBAL)

#### 51.4.3.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 51.4.3.4.2 Diagram




### 51.4.3.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>NOUPDATE | No Update<br>Blocks updates to all internal double buffered registers while it remains set. Software can still update the software visible registers, but changes cannot take effect until this bit is cleared at which point they update as normal.<br>0b - Internal double buffered registers update as normal.<br>1b - Internal double buffered registers do not update. |

## 51.4.3.5 Status and Control (SC)

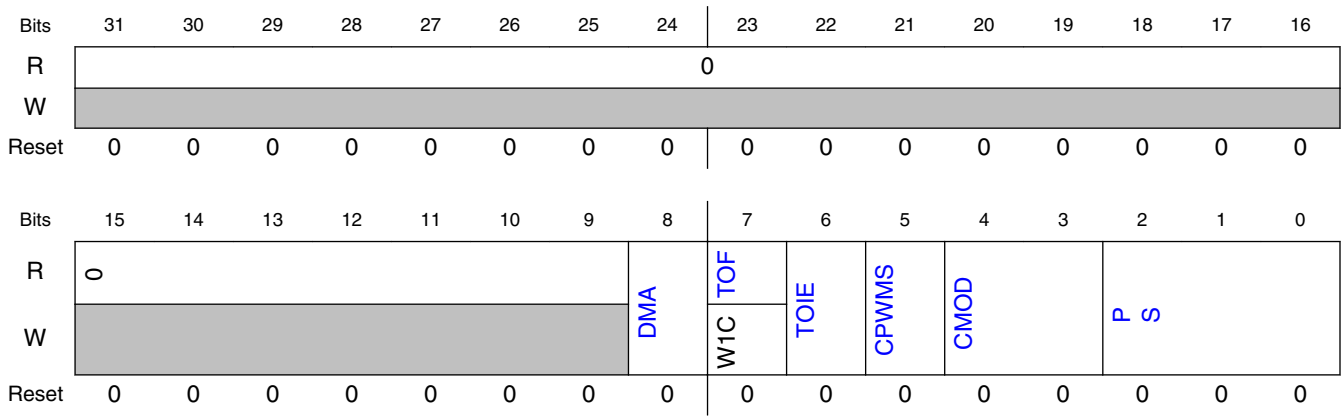
### 51.4.3.5.1 Offset

| Register | Offset |
|---|---|
| SC | 10h |

### 51.4.3.5.2 Function

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

### 51.4.3.5.3 Diagram




### 51.4.3.5.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>DMA | DMA Enable<br>Enables DMA transfers for the overflow flag.<br>0b - Disables DMA transfers.<br>1b - Enables DMA transfers. |
| 7<br>TOF | Timer Overflow Flag<br>Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect.<br>If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 6<br>TOIE | Timer Overflow Interrupt Enable<br>Enables TPM overflow interrupts.<br>0b - Disable TOF interrupts. Use software polling or DMA request.<br>1b - Enable TOF interrupts. An interrupt is generated when TOF equals one. |
| 5<br>CPWMS | Center-Aligned PWM Select<br>Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode.<br>This field is write protected. It can be written only when the counter is disabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - TPM counter operates in up counting mode.<br>1b - TPM counter operates in up-down counting mode. |
| 4-3<br>CMOD | Clock Mode Selection<br>Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the TPM clock domain.<br>00b - TPM counter is disabled<br>01b - TPM counter increments on every TPM counter clock<br>10b - TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock<br>11b - TPM counter increments on rising edge of the selected external input trigger. |
| 2-0<br>PS | Prescale Factor Selection<br>Selects one of 8 division factors for the clock mode selected by CMOD.<br>This field is write protected. It can be written only when the counter is disabled.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 51.4.3.6 Counter (CNT)

### 51.4.3.6.1 Offset

| Register | Offset |
|---|---|
| CNT | 14h |

### 51.4.3.6.2 Function

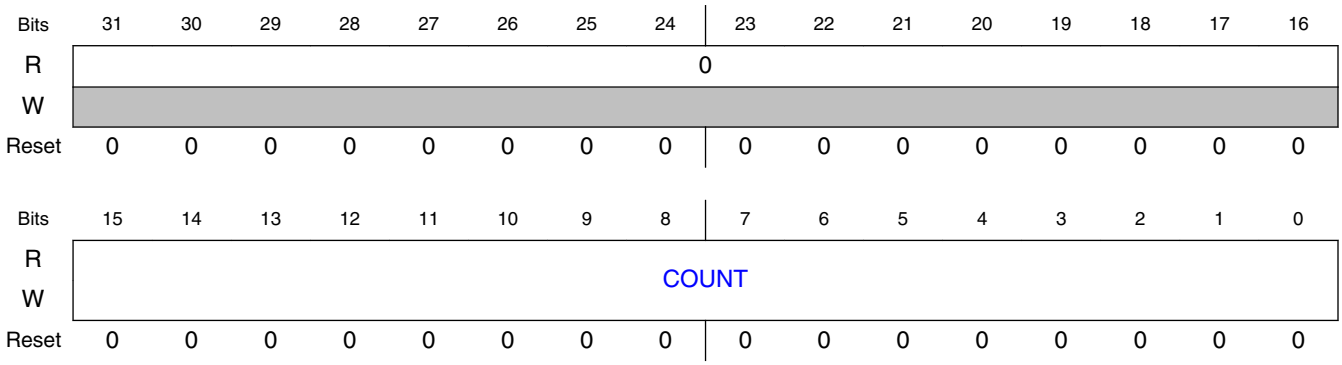
The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT will trigger a reload of the counter, this will force PWM outputs to their reload state..

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

#### 51.4.3.6.3 Diagram




#### 51.4.3.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>COUNT | Counter value |

### 51.4.3.7 Modulo (MOD)

#### 51.4.3.7.1 Offset

| Register | Offset |
|---|---|
| MOD | 18h |

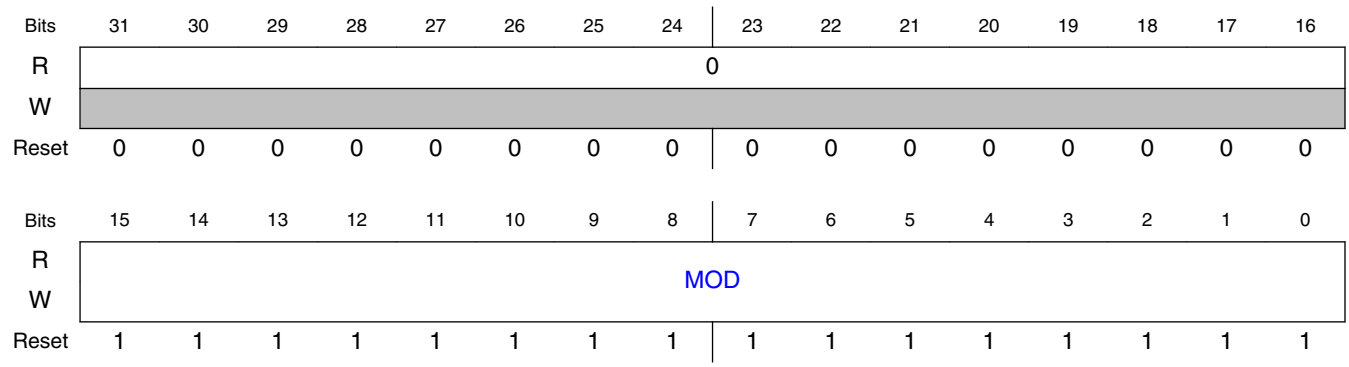
#### 51.4.3.7.2 Function

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM counter depends on the selected counting method (see Counter ).

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to MOD Register Update . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write to CNT) before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

#### 51.4.3.7.3 Diagram




#### 51.4.3.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>MOD | Modulo value<br>This field must be written with single 16-bit or 32-bit access. |

### 51.4.3.8 Capture and Compare Status (STATUS)

#### 51.4.3.8.1 Offset

| Register | Offset |
|---|---|
| STATUS | 1Ch |

#### 51.4.3.8.2 Function

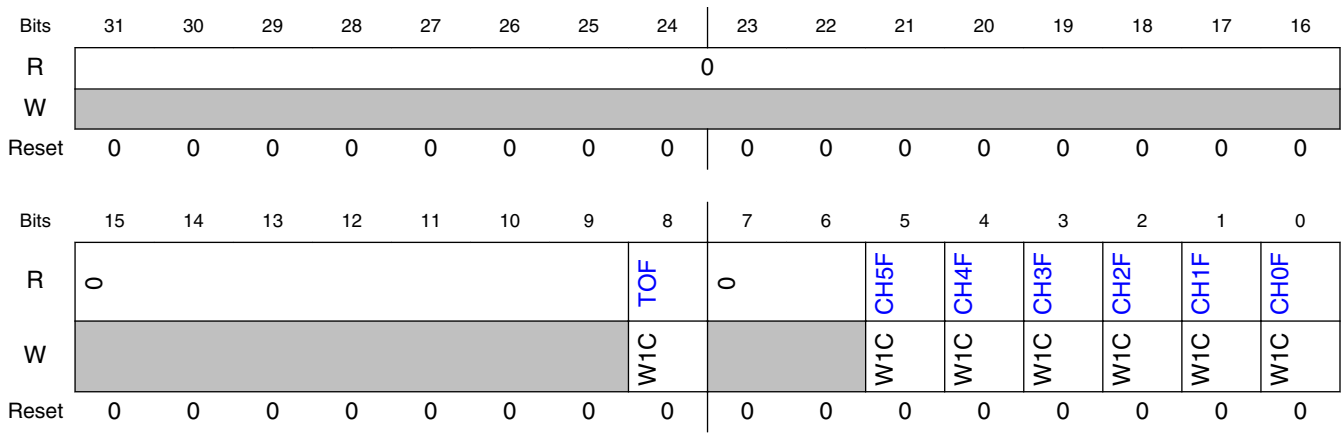
The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

### 51.4.3.8.3 Diagram




### 51.4.3.8.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TOF | Timer Overflow Flag<br>See register description<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 7-6<br>— | Reserved |
| 5<br>CH5F | Channel 5 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 4 | Channel 4 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CH4F | See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 3<br>CH3F | Channel 3 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 2<br>CH2F | Channel 2 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 1<br>CH1F | Channel 1 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 0<br>CH0F | Channel 0 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |

## 51.4.3.9 Channel (n) Status and Control (C0SC - C5SC)

### 51.4.3.9.1 Offset

For a = 0 to 5:

| Register | Offset |
|---|---|
| CaSC | 20h + (a × 8h) |

### 51.4.3.9.2 Function

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 51-5. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 00 | None | Channel disabled |
| 0 | 01 | 00 | Software compare | Pin not used for TPM |
| 0 | 00 | 01 | Input capture | Capture on Rising Edge Only |

*Table continues on the next page...*

## Table 51-5. Mode, Edge, and Level Selection (continued)

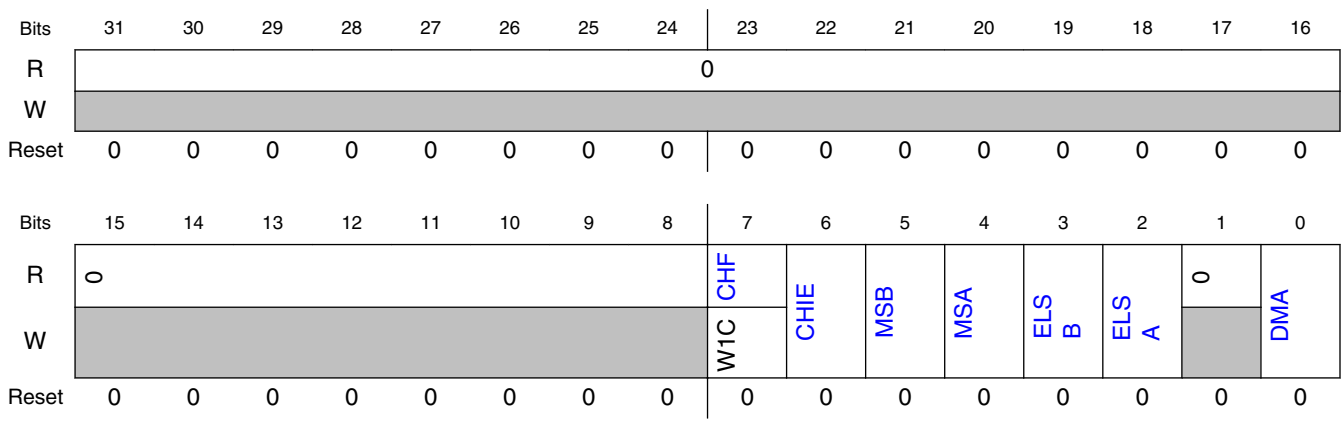
| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | 10 | | Capture on Falling Edge Only |
| | | 11 | | Capture on Rising or Falling Edge |
| | 01 | 01 | Output compare | Toggle Output on match |
| | | 10 | | Clear Output on match |
| | | 11 | | Set Output on match |
| | 10 | 10 | Edge-aligned PWM | High-true pulses (clear Output on counter match, set Output on counter reload, set Output when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match, set Output on counter reload, clear Output when counter first enabled or paused) |
| | | 01 | | Low-true pulses (set Output on counter match, clear Output on counter reload, clear Output when counter first enabled or paused) |
| | | 11 | | Low-true pulses (set Output on counter match, clear Output on counter reload, set Output when counter first enabled or paused) |
| | 11 | 10 | Output compare | Pulse Output low on match |
| | | 01 | | Pulse Output high on match |
| 1 | 10 | 10 | Center-aligned PWM | High-true pulses (clear Output on counter match-up, set Output on counter match-down, set Output on counter reload or when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match-up, set Output on counter match-down, clear Output on |

*Table continues on the next page...*

**Table 51-5. Mode, Edge, and Level Selection (continued)**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | | | counter reload or when counter first enabled or paused)) |
| | | 01 | | Low-true pulses (set Output on match-up, clear Output on match-down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 11 | | Low-true pulses (set Output on match-up, clear Output on match-down, set Output on counter reload or when counter first enabled or paused)) |

## 51.4.3.9.3 Diagram




## 51.4.3.9.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>CHF | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect.<br>If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 6<br>CHIE | Channel Interrupt Enable<br>Enables channel interrupts.<br>0b - Disable channel interrupts.<br>1b - Enable channel interrupts. |
| 5<br>MSB | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 4<br>MSA | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 3<br>ELSB | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. ph> |
| 2<br>ELSA | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. |
| 1<br>— | Reserved |
| 0<br>DMA | DMA Enable<br>Enables DMA transfers for the channel.<br>0b - Disable DMA transfers.<br>1b - Enable DMA transfers. |

## 51.4.3.10 Channel (n) Value (C0V - C5V)

### 51.4.3.10.1 Offset

For a = 0 to 5:

| Register | Offset |
|---|---|
| CaV | 24h + (a × 8h) |

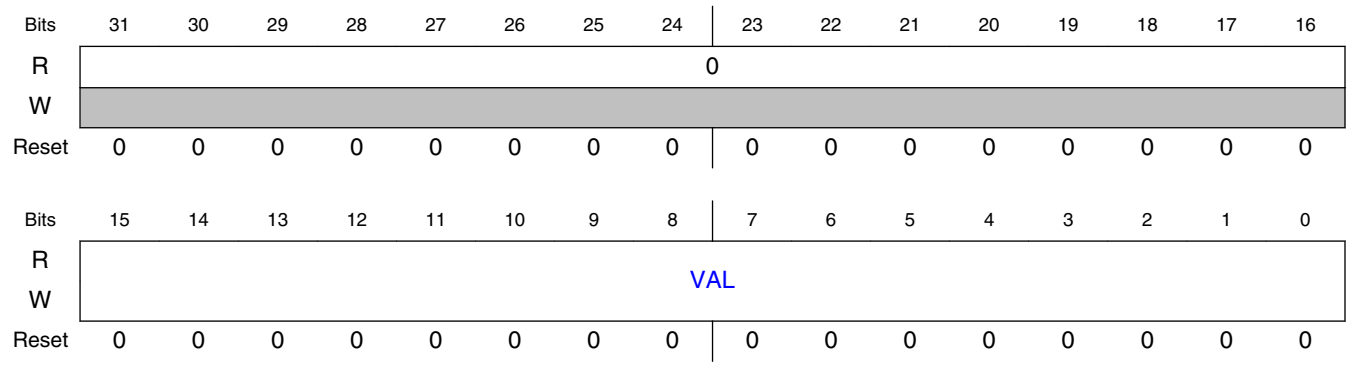
### 51.4.3.10.2 Function

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to CnV Register Update . Additional writes to the CnV write buffer are ignored until the register has been updated.

### 51.4.3.10.3 Diagram




### 51.4.3.10.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>VAL | Channel Value<br>Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access. |

## 51.4.3.11 Combine Channel Register (COMBINE)

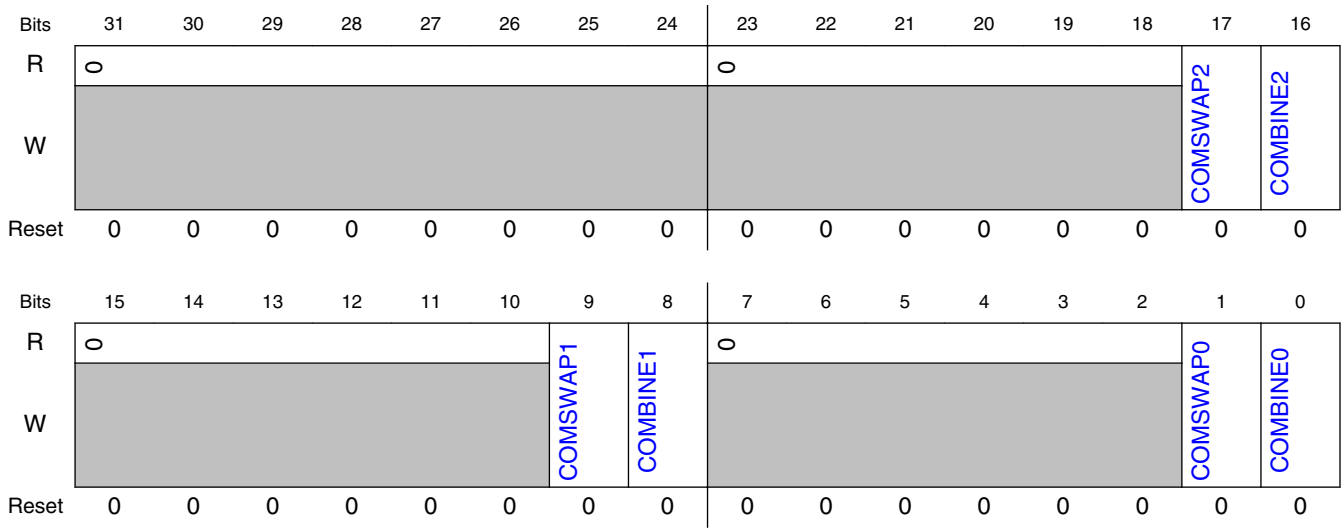
### 51.4.3.11.1 Offset

| Register | Offset |
|---|---|
| COMBINE | 64h |

### 51.4.3.11.2 Function

This register contains the control bits used to configure the combine channel modes for each pair of channels (n) and (n+1), where n is all the even numbered channels.

## 51.4.3.11.3 Diagram




## 51.4.3.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17<br>COMSWAP2 | Combine Channels 4 and 5 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 16<br>COMBINE2 | Combine Channels 4 and 5<br>Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 4 and 5 are independent.<br>1b - Channels 4 and 5 are combined. |
| 15-10<br>— | Reserved |
| 9<br>COMSWAP1 | Combine Channels 2 and 3 Swap<br>When set in combine mode, the odd channel is used for the input capture and 1st compare, the even channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 8<br>COMBINE1 | Combine Channels 2 and 3 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Enables the combine feature for channels 2 and 3. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 2 and 3 are independent.<br>1b - Channels 2 and 3 are combined. |
| 7-2<br>— | Reserved |
| 1<br>COMSWAP0 | Combine Channel 0 and 1 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 0<br>COMBINE0 | Combine Channels 0 and 1<br>Enables the combine feature for channels 0 and 1. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 0 and 1 are independent.<br>1b - Channels 0 and 1 are combined. |

## 51.4.3.12 Channel Trigger (TRIG)

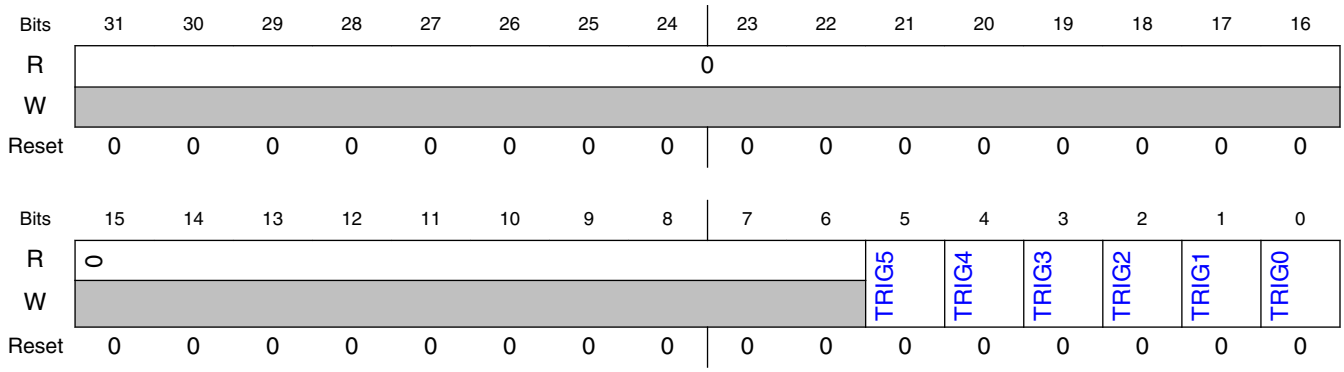
### 51.4.3.12.1 Offset

| Register | Offset |
|---|---|
| TRIG | 6Ch |

### 51.4.3.12.2 Function

In input capture mode, configures the trigger input that is used by the channel to capture the counter value. In output compare or PWM mode, configures the trigger input used to modulate the channel output. When modulating the output, the output is forced to the channel initial value whenever the trigger is not asserted. Note that the even numbered channels share the first input trigger source and the odd numbered channels share the second input trigger source.

### 51.4.3.12.3 Diagram




### 51.4.3.12.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>TRIG5 | Channel 5 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 5. |
| 4<br>TRIG4 | Channel 4 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 4. |
| 3<br>TRIG3 | Channel 3 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 3. |
| 2<br>TRIG2 | Channel 2 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 2. |
| 1<br>TRIG1 | Channel 1 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 1. |
| 0<br>TRIG0 | Channel 0 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 0. |

## 51.4.3.13 Channel Polarity (POL)

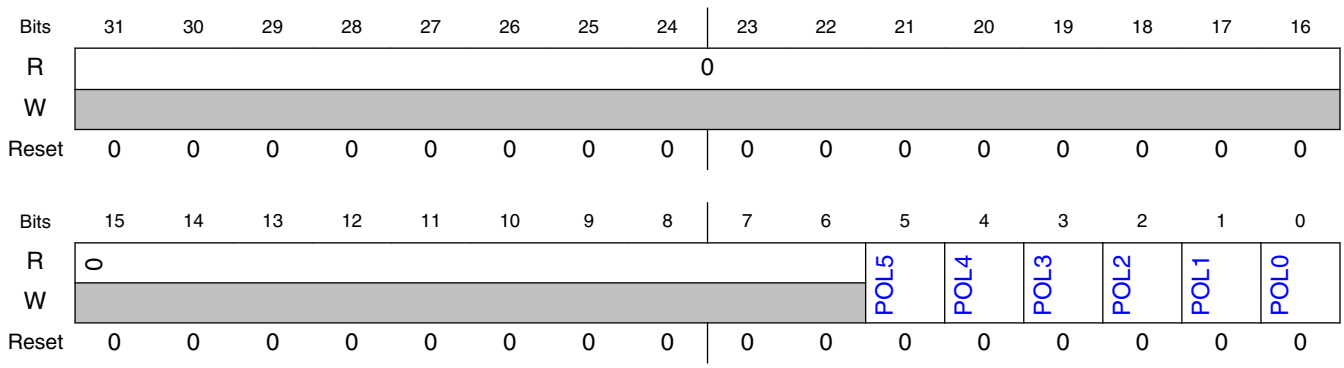
### 51.4.3.13.1 Offset

| Register | Offset |
|---|---|
| POL | 70h |

### 51.4.3.13.2 Function

This register defines the input and output polarity of each of the channels.

### 51.4.3.13.3 Diagram




### 51.4.3.13.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>POL5 | Channel 5 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 4<br>POL4 | Channel 4 Polarity<br>0b - The channel polarity is active high<br>1b - The channel polarity is active low. |
| 3<br>POL3 | Channel 3 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 2<br>POL2 | Channel 2 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 1<br>POL1 | Channel 1 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 0<br>POL0 | Channel 0 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |

# 51.4.3.14 Filter Control (FILTER)

## 51.4.3.14.1 Offset

| Register | Offset |
|---|---|
| FILTER | 78h |

## 51.4.3.14.2 Function

This register selects the filter value of the channel inputs, and an additional output delay value for the channel outputs. In PWM combine modes, the filter can be used to implement deadtime insertion.

## 51.4.3.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CH5FVAL | | | | CH4FVAL | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH3FVAL | | | | CH2FVAL | | | | CH1FVAL | | | | CH0FVAL | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 51.4.3.14.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20<br>CH5FVAL | Channel 5 Filter Value<br>Sets the filter value for the channel 5 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH5FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 5 in PWM modes. Deadtime insertion is disabled when CH5FVAL is zero, otherwise deadtime insertion for channel 5 is configured as (CH5FVAL * 4) clock cycles. |
| 19-16<br>CH4FVAL | Channel 4 Filter Value<br>Sets the filter value for the channel 4 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH4FVAL * 4) clock cycles. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Sets the deadtime insertion time for channel 4 in PWM modes. Deadtime insertion is disabled when CH4FVAL is zero, otherwise deadtime insertion for channel 4 is configured as (CH4FVAL * 4) clock cycles. |
| 15-12<br>CH3FVAL | Channel 3 Filter Value<br>Sets the filter value for the channel 3 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH3FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 3 in PWM modes. Deadtime insertion is disabled when CH3FVAL is zero, otherwise deadtime insertion for channel 3 is configured as (CH3FVAL * 4) clock cycles. |
| 11-8<br>CH2FVAL | Channel 2 Filter Value<br>Sets the filter value for the channel 2 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH2FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 2 in PWM modes. Deadtime insertion is disabled when CH2FVAL is zero, otherwise deadtime insertion for channel 2 is configured as (CH2FVAL * 4) clock cycles. |
| 7-4<br>CH1FVAL | Channel 1 Filter Value<br>Sets the filter value for the channel 1 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH1FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 1 in PWM modes. Deadtime insertion is disabled when CH1FVAL is zero, otherwise deadtime insertion for channel 1 is configured as (CH1FVAL * 4) clock cycles. |
| 3-0<br>CH0FVAL | Channel 0 Filter Value<br>Sets the filter value for the channel 0 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH0FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 0 in PWM modes. Deadtime insertion is disabled when CH0FVAL is zero, otherwise deadtime insertion for channel 0 is configured as (CH0FVAL * 4) clock cycles. |

## 51.4.3.15 Quadrature Decoder Control and Status (QDCTRL)

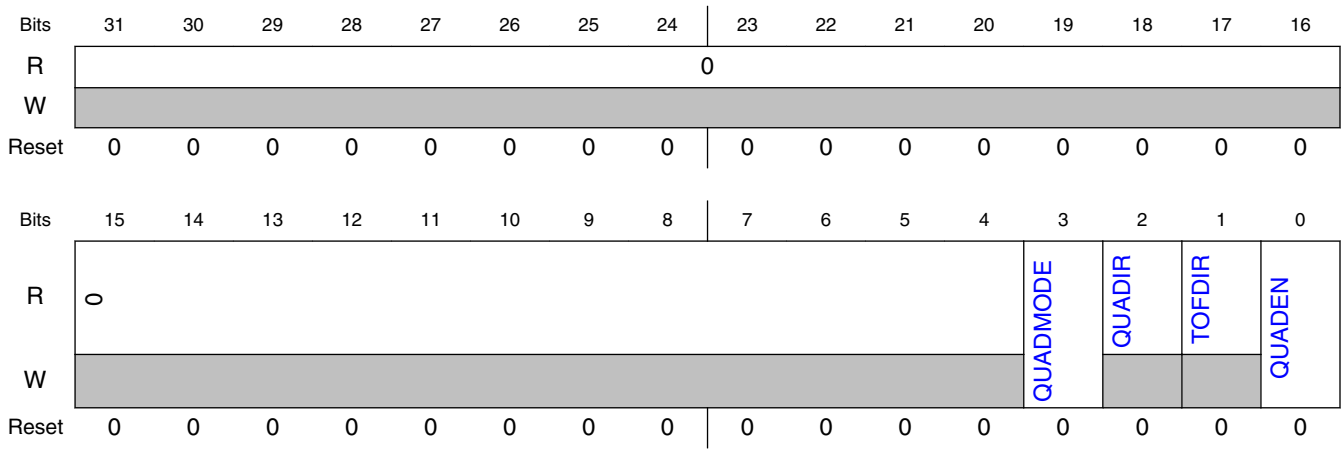
### 51.4.3.15.1 Offset

| Register | Offset |
|---|---|
| QDCTRL | 80h |

### 51.4.3.15.2 Function

This register has the control and status bits for the quadrature decoder mode.

## 51.4.3.15.3 Diagram




## 51.4.3.15.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>QUADMODE | Quadrature Decoder Mode<br>Selects the encoding mode used in the quadrature decoder mode.<br>0b - Phase encoding mode.<br>1b - Count and direction encoding mode. |
| 2<br>QUADIR | Counter Direction in Quadrature Decode Mode<br>Indicates the counting direction.<br>0b - Counter direction is decreasing (counter decrement).<br>1b - Counter direction is increasing (counter increment). |
| 1<br>TOFDIR | TOFDIR<br>Indicates if the TOF bit was set on the top or the bottom of counting.<br>0b - TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (zero) to its maximum value (MOD register).<br>1b - TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (zero). |
| 0<br>QUADEN | QUADEN<br>Enables the quadrature decoder mode. In this mode, the channel 0 and channel 1 inputs control the TPM counter direction and can only be used for software compare. The quadrature decoder mode has precedence over the other modes.<br>0b - Quadrature decoder mode is disabled.<br>1b - Quadrature decoder mode is enabled. |

## 51.4.3.16 Configuration (CONF)

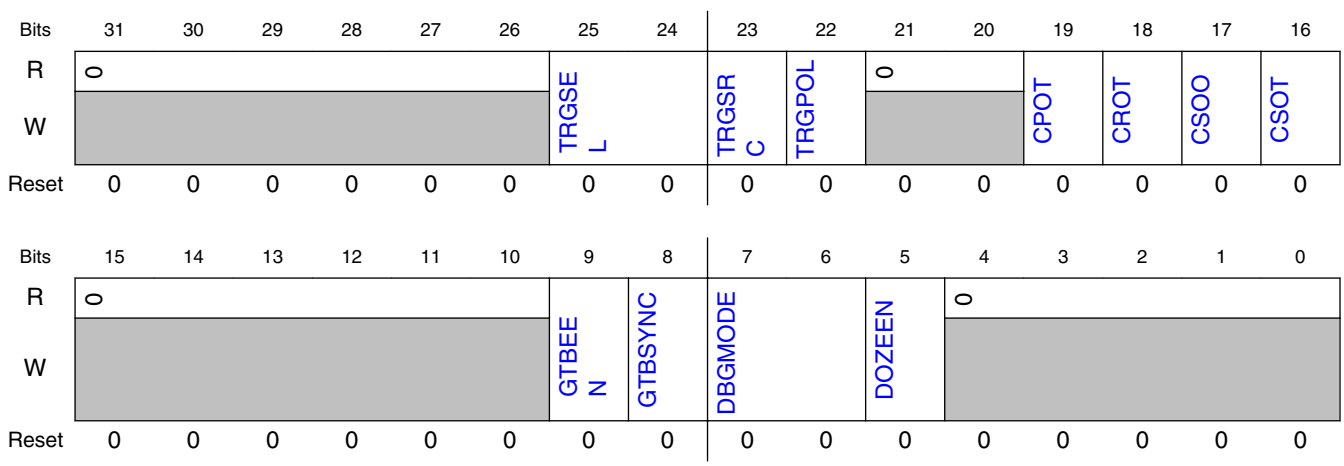
### 51.4.3.16.1 Offset

| Register | Offset |
|---|---|
| CONF | 84h |

### 51.4.3.16.2 Function

This register selects the behavior in debug and wait modes and the use of an external global time base.

### 51.4.3.16.3 Diagram




### 51.4.3.16.4 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>TRGSEL | Trigger Select<br>Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled.<br>Refer to chip configuration section for available external trigger options.<br>The available internal trigger sources are listed below.<br>01b - Channel 0 pin input capture<br>10b - Channel 1 pin input capture |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 11b - Channel 0 or Channel 1 pin input capture |
| 23<br>TRGSRC | Trigger Source<br>Selects between internal (channel pin input capture) or external trigger sources.<br>When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins.<br>This field should only be changed when the TPM counter is disabled.<br>0b - Trigger source selected by TRGSEL is external.<br>1b - Trigger source selected by TRGSEL is internal (channel pin input capture). |
| 22<br>TRGPOL | Trigger Polarity<br>Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled.<br>0b - Trigger is active high.<br>1b - Trigger is active low. |
| 21-20<br>— | Reserved |
| 19<br>CPOT | Counter Pause On Trigger<br>When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. |
| 18<br>CROT | Counter Reload On Trigger<br>When set, the TPM counter will reload with zero (and set PWM outputs to their reload state) when a rising edge is detected on the selected trigger input.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - Counter is not reloaded due to a rising edge on the selected input trigger<br>1b - Counter is reloaded when a rising edge is detected on the selected input trigger |
| 17<br>CSOO | Counter Stop On Overflow<br>When set, the TPM counter will pause incrementing once the counter equals the MOD value and increments (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT is set. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter continues incrementing or decrementing after overflow<br>1b - TPM counter stops incrementing or decrementing after overflow. |
| 16<br>CSOT | Counter Start on Trigger<br>When set, the TPM counter will pause incrementing when it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is paused due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter starts to increment immediately, once it is enabled.<br>1b - TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow. |
| 15-10<br>— | Reserved |
| 9<br>GTBEEN | Global time base enable<br>Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag.<br>0b - All channels use the internally generated TPM counter as their timebase<br>1b - All channels use an externally generated global timebase as their timebase |
| 8<br>GTBSYNC | Global Time Base Synchronization<br>When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger, paused state and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled.<br>0b - Global timebase synchronization disabled.<br>1b - Global timebase synchronization enabled. |
| 7-6<br>DBGMODE | Debug Mode<br>Configures the TPM behavior in debug mode. All other configurations are reserved.<br>00b - TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state.<br>11b - TPM counter continues in debug mode. |
| 5<br>DOZEEN | Doze Enable<br>Configures the TPM behavior in wait mode.<br>0b - Internal TPM counter continues in Doze mode.<br>1b - Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state. |
| 4-0<br>— | Reserved |

## 51.4.4 TPM3 register descriptions

### 51.4.4.1 TPM3 Memory map

TPM3 base address: 4102_D000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0600_0007h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0010_0402h |
| 8h | TPM Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| 10h | Status and Control (SC) | 32 | RW | 0000_0000h |
| 14h | Counter (CNT) | 32 | RW | 0000_0000h |
| 18h | Modulo (MOD) | 32 | RW | 0000_FFFFh |
| 1Ch | Capture and Compare Status (STATUS) | 32 | W1C | 0000_0000h |
| 20h | Channel (n) Status and Control (C0SC) | 32 | RW | 0000_0000h |
| 24h | Channel (n) Value (C0V) | 32 | RW | 0000_0000h |
| 28h | Channel (n) Status and Control (C1SC) | 32 | RW | 0000_0000h |
| 2Ch | Channel (n) Value (C1V) | 32 | RW | 0000_0000h |
| 64h | Combine Channel Register (COMBINE) | 32 | RW | 0000_0000h |
| 6Ch | Channel Trigger (TRIG) | 32 | RW | 0000_0000h |
| 70h | Channel Polarity (POL) | 32 | RW | 0000_0000h |
| 78h | Filter Control (FILTER) | 32 | RW | 0000_0000h |
| 80h | Quadrature Decoder Control and Status (QDCTRL) | 32 | RW | 0000_0000h |
| 84h | Configuration (CONF) | 32 | RW | 0000_0000h |

## 51.4.4.2 Version ID Register (VERID)

### 51.4.4.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 51.4.4.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

#### 51.4.4.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with Filter and Combine registers implemented.<br>0000000000000111b - Standard feature set with Filter, Combine and Quadrature registers implemented. |

### 51.4.4.3 Parameter Register (PARAM)

#### 51.4.4.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 51.4.4.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WIDTH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRIG | | | | | | | | CHAN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 51.4.4.3.3 Fields

| Field | Function |
|---|---|
| 31-24 | Reserved |

*Table continues on the next page...*

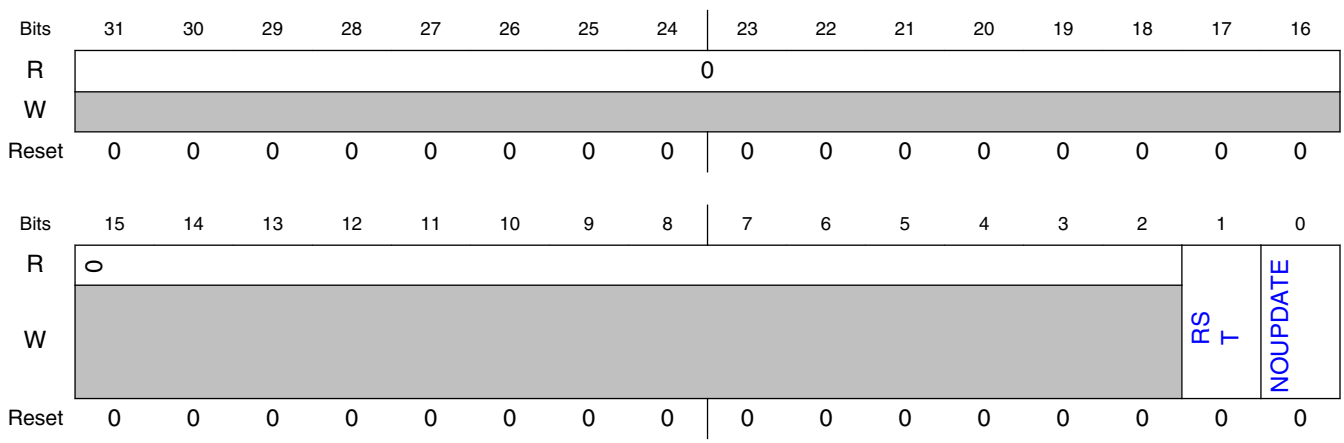
| Field | Function |
|---|---|
| — | |
| 23-16<br>WIDTH | Counter Width<br>Width of the counter and timer channels. |
| 15-8<br>TRIG | Trigger Count<br>Number of trigger inputs implemented. |
| 7-0<br>CHAN | Channel Count<br>Number of timer channels implemented. |

## 51.4.4.4 TPM Global Register (GLOBAL)

### 51.4.4.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 51.4.4.4.2 Diagram




### 51.4.4.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>NOUPDATE | No Update<br>Blocks updates to all internal double buffered registers while it remains set. Software can still update the software visible registers, but changes cannot take effect until this bit is cleared at which point they update as normal.<br>0b - Internal double buffered registers update as normal.<br>1b - Internal double buffered registers do not update. |

## 51.4.4.5 Status and Control (SC)

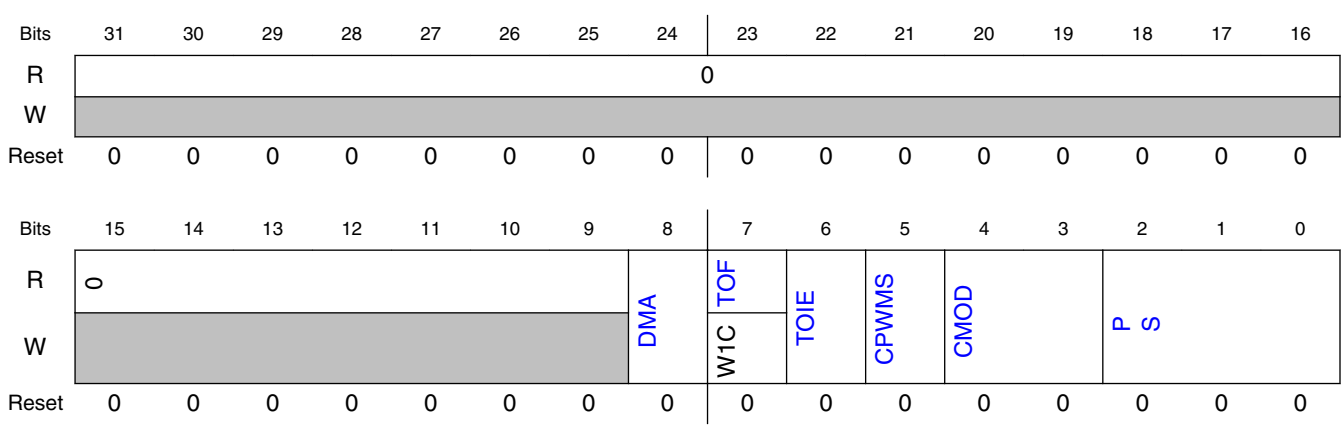
### 51.4.4.5.1 Offset

| Register | Offset |
|---|---|
| SC | 10h |

### 51.4.4.5.2 Function

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

### 51.4.4.5.3 Diagram



## 51.4.4.5.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>DMA | DMA Enable<br>Enables DMA transfers for the overflow flag.<br>0b - Disables DMA transfers.<br>1b - Enables DMA transfers. |
| 7<br>TOF | Timer Overflow Flag<br>Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect.<br>If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 6<br>TOIE | Timer Overflow Interrupt Enable<br>Enables TPM overflow interrupts.<br>0b - Disable TOF interrupts. Use software polling or DMA request.<br>1b - Enable TOF interrupts. An interrupt is generated when TOF equals one. |
| 5<br>CPWMS | Center-Aligned PWM Select<br>Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode.<br>This field is write protected. It can be written only when the counter is disabled.<br>0b - TPM counter operates in up counting mode.<br>1b - TPM counter operates in up-down counting mode. |
| 4-3<br>CMOD | Clock Mode Selection<br>Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the TPM clock domain.<br>00b - TPM counter is disabled<br>01b - TPM counter increments on every TPM counter clock<br>10b - TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock<br>11b - TPM counter increments on rising edge of the selected external input trigger. |
| 2-0<br>PS | Prescale Factor Selection<br>Selects one of 8 division factors for the clock mode selected by CMOD.<br>This field is write protected. It can be written only when the counter is disabled.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 51.4.4.6 Counter (CNT)

### 51.4.4.6.1 Offset

| Register | Offset |
|---|---|
| CNT | 14h |

### 51.4.4.6.2 Function

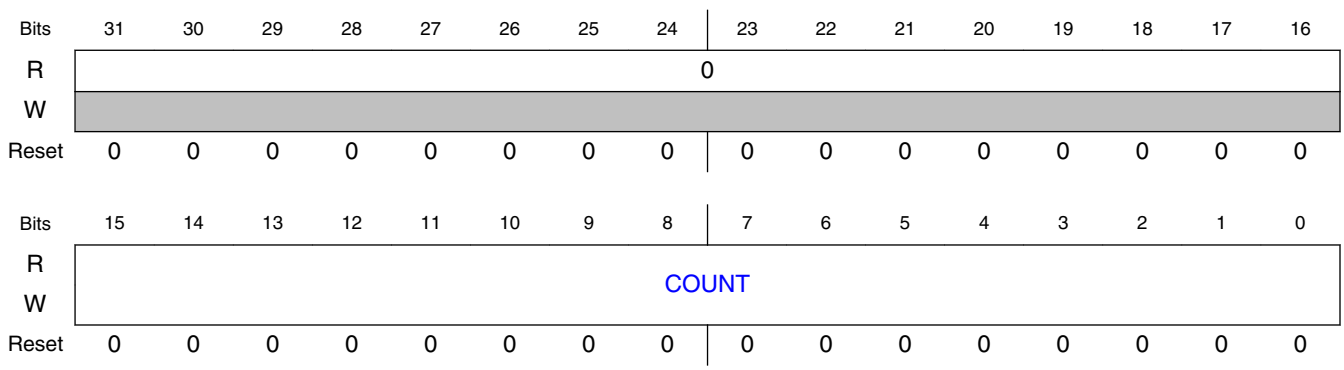
The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT will trigger a reload of the counter, this will force PWM outputs to their reload state..

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

### 51.4.4.6.3 Diagram




### 51.4.4.6.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>COUNT | Counter value |

## 51.4.4.7 Modulo (MOD)

### 51.4.4.7.1 Offset

| Register | Offset |
|---|---|
| MOD | 18h |

### 51.4.4.7.2 Function

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM counter depends on the selected counting method (see Counter ).

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to MOD Register Update . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write to CNT) before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

### 51.4.4.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MOD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 51.4.4.7.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0 | Modulo value |

| Field | Function |
|---|---|
| MOD | This field must be written with single 16-bit or 32-bit access. |

## 51.4.4.8 Capture and Compare Status (STATUS)

### 51.4.4.8.1 Offset

| Register | Offset |
|---|---|
| STATUS | 1Ch |

### 51.4.4.8.2 Function

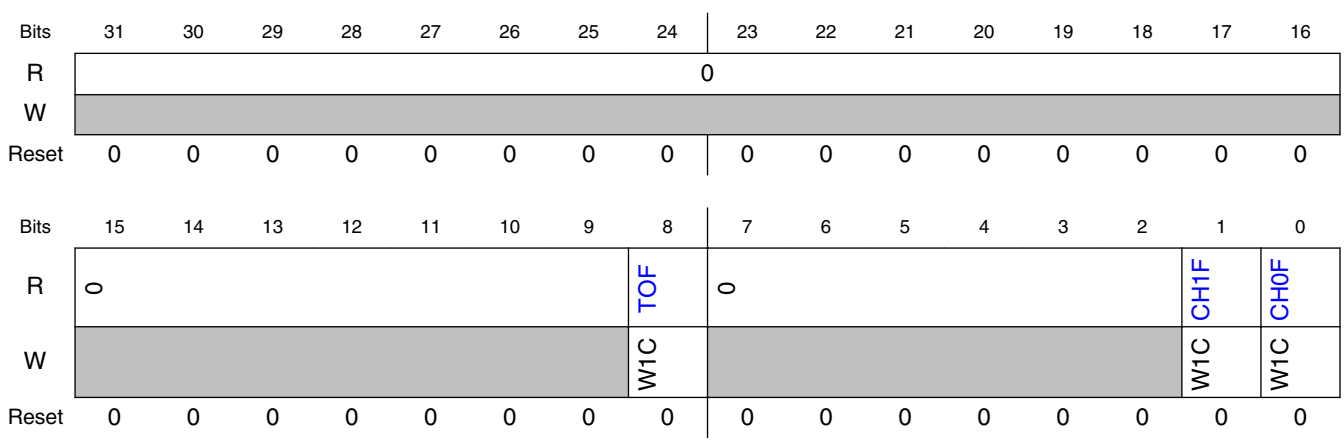
The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

### 51.4.4.8.3 Diagram



#### 51.4.4.8.4 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TOF | Timer Overflow Flag<br>See register description<br>0b - TPM counter has not overflowed.<br>1b - TPM counter has overflowed. |
| 7-2<br>— | Reserved |
| 1<br>CH1F | Channel 1 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 0<br>CH0F | Channel 0 Flag<br>See the register description.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |

### 51.4.4.9 Channel (n) Status and Control (C0SC - C1SC)

#### 51.4.4.9.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| CaSC | 20h + (a × 8h) |

#### 51.4.4.9.2 Function

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 51-6. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 00 | None | Channel disabled |
| 0 | 01 | 00 | Software compare | Pin not used for TPM |

*Table continues on the next page...*

## Table 51-6. Mode, Edge, and Level Selection (continued)

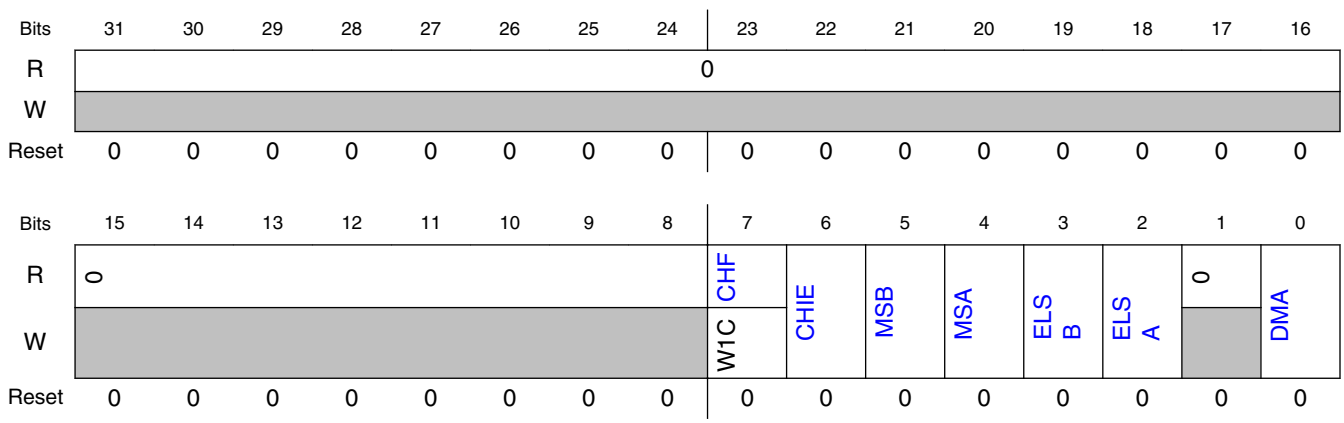
| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 01 | Input capture | Capture on Rising Edge Only |
| | | 10 | | Capture on Falling Edge Only |
| | | 11 | | Capture on Rising or Falling Edge |
| | 01 | 01 | Output compare | Toggle Output on match |
| | | 10 | | Clear Output on match |
| | | 11 | | Set Output on match |
| | 10 | 10 | Edge-aligned PWM | High-true pulses (clear Output on counter match, set Output on counter reload, set Output when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match, set Output on counter reload, clear Output when counter first enabled or paused) |
| | | 01 | | Low-true pulses (set Output on counter match, clear Output on counter reload, clear Output when counter first enabled or paused) |
| | | 11 | | Low-true pulses (set Output on counter match, clear Output on counter reload, set Output when counter first enabled or paused) |
| | 11 | 10 | Output compare | Pulse Output low on match |
| | | 01 | | Pulse Output high on match |
| 1 | 10 | 10 | Center-aligned PWM | High-true pulses (clear Output on counter match-up, set Output on counter match-down, set Output on counter reload or when counter first enabled or paused) |
| | | 00 | | High-true pulses (clear Output on counter match-up, set Output on counter match- |

*Table continues on the next page...*

**Table 51-6. Mode, Edge, and Level Selection (continued)**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| | | | | down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 01 | | Low-true pulses (set Output on match-up, clear Output on match-down, clear Output on counter reload or when counter first enabled or paused)) |
| | | 11 | | Low-true pulses (set Output on match-up, clear Output on match-down, set Output on counter reload or when counter first enabled or paused)) |

## 51.4.4.9.3 Diagram




## 51.4.4.9.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>CHF | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF.<br>0b - No channel event has occurred.<br>1b - A channel event has occurred. |
| 6<br>CHIE | Channel Interrupt Enable<br>Enables channel interrupts.<br>0b - Disable channel interrupts.<br>1b - Enable channel interrupts. |
| 5<br>MSB | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 4<br>MSA | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. |
| 3<br>ELSB | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. ph> |
| 2<br>ELSA | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. |
| 1<br>— | Reserved |
| 0<br>DMA | DMA Enable<br>Enables DMA transfers for the channel.<br>0b - Disable DMA transfers.<br>1b - Enable DMA transfers. |

## 51.4.4.10 Channel (n) Value (C0V - C1V)

### 51.4.4.10.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| CaV | 24h + (a × 8h) |

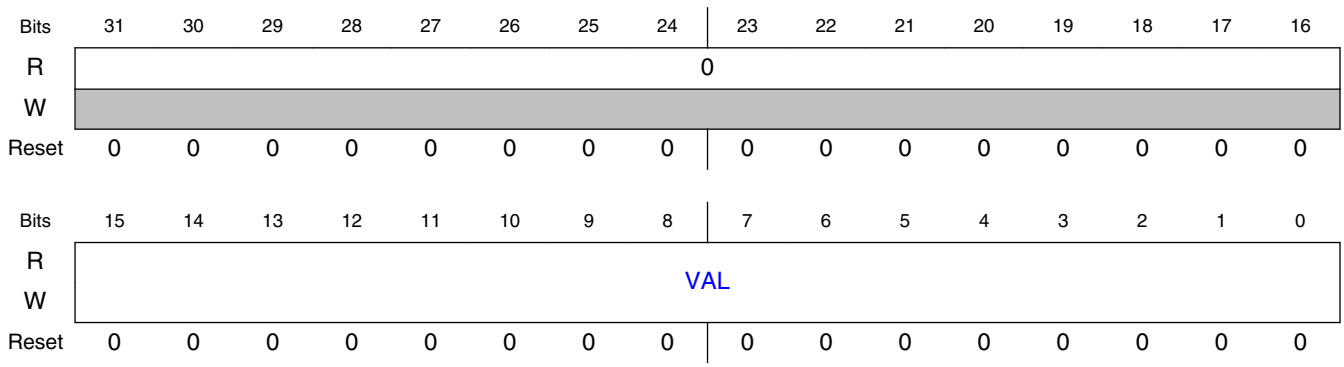
### 51.4.4.10.2 Function

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to CnV Register Update . Additional writes to the CnV write buffer are ignored until the register has been updated.

### 51.4.4.10.3 Diagram




### 51.4.4.10.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>VAL | Channel Value<br>Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access. |

## 51.4.4.11 Combine Channel Register (COMBINE)

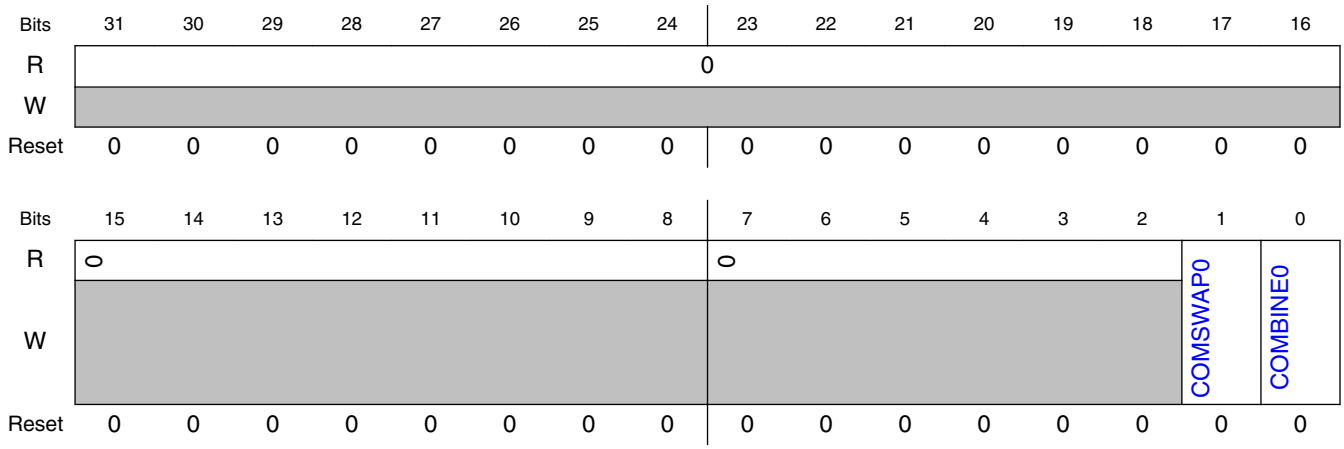
### 51.4.4.11.1 Offset

| Register | Offset |
|---|---|
| COMBINE | 64h |

### 51.4.4.11.2 Function

This register contains the control bits used to configure the combine channel modes for each pair of channels (n) and (n+1), where n is all the even numbered channels.

## 51.4.4.11.3 Diagram




## 51.4.4.11.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1<br>COMSWAP0 | Combine Channel 0 and 1 Swap<br>When set in combine mode, the even channel is used for the input capture and 1st compare, the odd channel is used for the 2nd compare.<br>0b - Even channel is used for input capture and 1st compare.<br>1b - Odd channel is used for input capture and 1st compare. |
| 0<br>COMBINE0 | Combine Channels 0 and 1<br>Enables the combine feature for channels 0 and 1. In input capture mode, the combined channels use the even channel input. In software compare modes, the even channel match asserts the output trigger and the odd channel match negates the output trigger. In PWM modes, the even channel match is used for the 1st compare and odd channel match for the 2nd compare.<br>0b - Channels 0 and 1 are independent.<br>1b - Channels 0 and 1 are combined. |

# 51.4.4.12 Channel Trigger (TRIG)

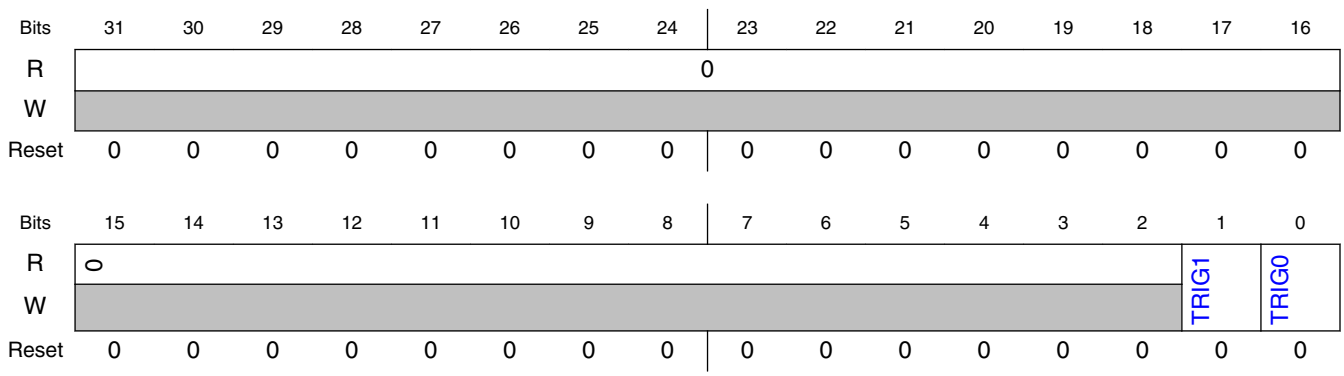
## 51.4.4.12.1 Offset

| Register | Offset |
|---|---|
| TRIG | 6Ch |

#### 51.4.4.12.2 Function

In input capture mode, configures the trigger input that is used by the channel to capture the counter value. In output compare or PWM mode, configures the trigger input used to modulate the channel output. When modulating the output, the output is forced to the channel initial value whenever the trigger is not asserted. Note that the even numbered channels share the first input trigger source and the odd numbered channels share the second input trigger source.

#### 51.4.4.12.3 Diagram




#### 51.4.4.12.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>TRIG1 | Channel 1 Trigger<br>0b - No effect.<br>1b - Configures trigger input 1 to be used by channel 1. |
| 0<br>TRIG0 | Channel 0 Trigger<br>0b - No effect.<br>1b - Configures trigger input 0 to be used by channel 0. |

### 51.4.4.13 Channel Polarity (POL)

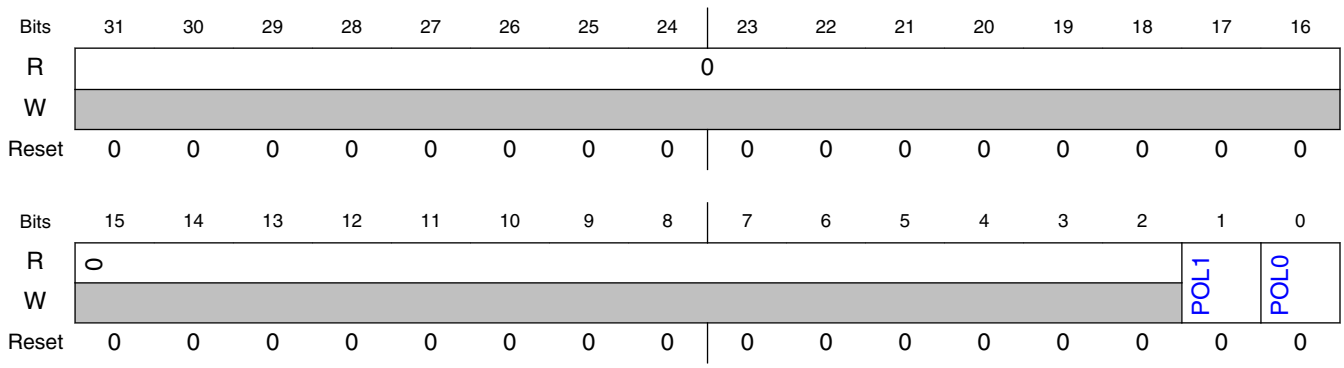
#### 51.4.4.13.1 Offset

| Register | Offset |
|---|---|
| POL | 70h |

### 51.4.4.13.2 Function

This register defines the input and output polarity of each of the channels.

### 51.4.4.13.3 Diagram




### 51.4.4.13.4 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>POL1 | Channel 1 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |
| 0<br>POL0 | Channel 0 Polarity<br>0b - The channel polarity is active high.<br>1b - The channel polarity is active low. |

## 51.4.4.14 Filter Control (FILTER)

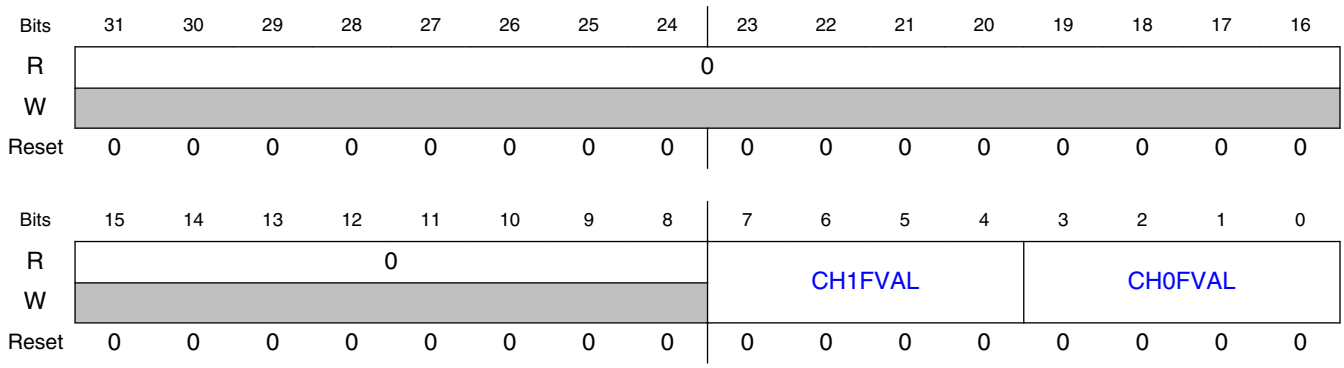
### 51.4.4.14.1 Offset

| Register | Offset |
|---|---|
| FILTER | 78h |

### 51.4.4.14.2 Function

This register selects the filter value of the channel inputs, and an additional output delay value for the channel outputs. In PWM combine modes, the filter can be used to implement deadtime insertion.

### 51.4.4.14.3 Diagram




### 51.4.4.14.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-4<br>CH1FVAL | Channel 1 Filter Value<br>Sets the filter value for the channel 1 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH1FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 1 in PWM modes. Deadtime insertion is disabled when CH1FVAL is zero, otherwise deadtime insertion for channel 1 is configured as (CH1FVAL * 4) clock cycles. |
| 3-0<br>CH0FVAL | Channel 0 Filter Value<br>Sets the filter value for the channel 0 input when configured for input capture or software compare modes. The filter is disabled when the value is zero, otherwise the filter is configured as (CH0FVAL * 4) clock cycles.<br>Sets the deadtime insertion time for channel 0 in PWM modes. Deadtime insertion is disabled when CH0FVAL is zero, otherwise deadtime insertion for channel 0 is configured as (CH0FVAL * 4) clock cycles. |

## 51.4.4.15 Quadrature Decoder Control and Status (QDCTRL)

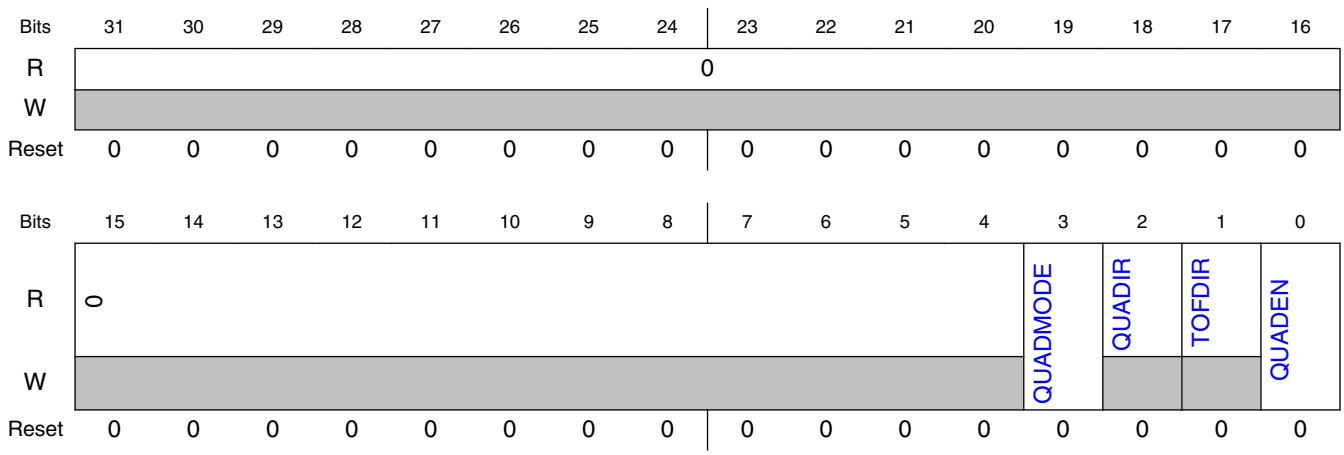
### 51.4.4.15.1 Offset

| Register | Offset |
|---|---|
| QDCTRL | 80h |

### 51.4.4.15.2 Function

This register has the control and status bits for the quadrature decoder mode.

### 51.4.4.15.3 Diagram




### 51.4.4.15.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>QUADMODE | Quadrature Decoder Mode<br>Selects the encoding mode used in the quadrature decoder mode.<br>0b - Phase encoding mode.<br>1b - Count and direction encoding mode. |
| 2<br>QUADIR | Counter Direction in Quadrature Decode Mode<br>Indicates the counting direction.<br>0b - Counter direction is decreasing (counter decrement).<br>1b - Counter direction is increasing (counter increment). |
| 1<br>TOFDIR | TOFDIR<br>Indicates if the TOF bit was set on the top or the bottom of counting.<br>0b - TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (zero) to its maximum value (MOD register).<br>1b - TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (zero). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>QUADEN | QUADEN<br>Enables the quadrature decoder mode. In this mode, the channel 0 and channel 1 inputs control the TPM counter direction and can only be used for software compare. The quadrature decoder mode has precedence over the other modes.<br>0b - Quadrature decoder mode is disabled.<br>1b - Quadrature decoder mode is enabled. |

## 51.4.4.16 Configuration (CONF)

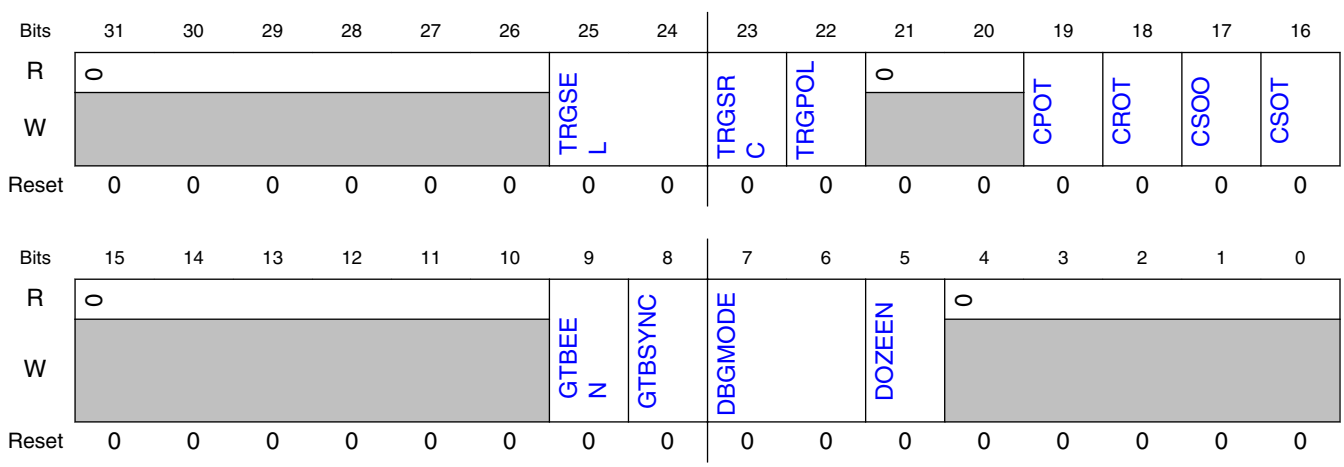
### 51.4.4.16.1 Offset

| Register | Offset |
|---|---|
| CONF | 84h |

### 51.4.4.16.2 Function

This register selects the behavior in debug and wait modes and the use of an external global time base.

### 51.4.4.16.3 Diagram




### 51.4.4.16.4 Fields

| Field | Function |
|---|---|
| 31-26 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 25-24<br>TRGSEL | Trigger Select<br>Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled.<br>Refer to chip configuration section for available external trigger options.<br>The available internal trigger sources are listed below.<br>01b - Channel 0 pin input capture<br>10b - Channel 1 pin input capture<br>11b - Channel 0 or Channel 1 pin input capture |
| 23<br>TRGSRC | Trigger Source<br>Selects between internal (channel pin input capture) or external trigger sources.<br>When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins.<br>This field should only be changed when the TPM counter is disabled.<br>0b - Trigger source selected by TRGSEL is external.<br>1b - Trigger source selected by TRGSEL is internal (channel pin input capture). |
| 22<br>TRGPOL | Trigger Polarity<br>Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled.<br>0b - Trigger is active high.<br>1b - Trigger is active low. |
| 21-20<br>— | Reserved |
| 19<br>CPOT | Counter Pause On Trigger<br>When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled. |
| 18<br>CROT | Counter Reload On Trigger<br>When set, the TPM counter will reload with zero (and set PWM outputs to their reload state) when a rising edge is detected on the selected trigger input.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - Counter is not reloaded due to a rising edge on the selected input trigger<br>1b - Counter is reloaded when a rising edge is detected on the selected input trigger |
| 17<br>CSOO | Counter Stop On Overflow<br>When set, the TPM counter will pause incrementing once the counter equals the MOD value and increments (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT is set. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter continues incrementing or decrementing after overflow<br>1b - TPM counter stops incrementing or decrementing after overflow. |
| 16<br>CSOT | Counter Start on Trigger<br>When set, the TPM counter will pause incrementing when it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is paused due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again. While the counter is paused input capture events are ignored, and PWM outputs are forced to their default state.<br>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.<br>0b - TPM counter starts to increment immediately, once it is enabled.<br>1b - TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow. |
| 15-10<br>— | Reserved |
| 9<br>GTBEEN | Global time base enable<br>Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag.<br>0b - All channels use the internally generated TPM counter as their timebase<br>1b - All channels use an externally generated global timebase as their timebase |
| 8<br>GTBSYNC | Global Time Base Synchronization<br>When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger, paused state and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled.<br>0b - Global timebase synchronization disabled.<br>1b - Global timebase synchronization enabled. |
| 7-6<br>DBGMODE | Debug Mode<br>Configures the TPM behavior in debug mode. All other configurations are reserved.<br>00b - TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state.<br>11b - TPM counter continues in debug mode. |
| 5<br>DOZEEN | Doze Enable<br>Configures the TPM behavior in wait mode.<br>0b - Internal TPM counter continues in Doze mode.<br>1b - Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are ignored, and PWM outputs are forced to their default state. |
| 4-0<br>— | Reserved |

## 51.5 Functional description

The following sections describe the TPM features.

### 51.5.1 Clock domains

The TPM module supports two clock domains.

The bus clock domain is used by the register interface and for synchronizing interrupts and DMA requests.

The TPM counter clock domain is used to clock the counter and prescaler along with the output compare and input capture logic. The TPM counter clock is considered asynchronous to the bus clock, can be a higher or lower frequency than the bus clock and can remain operational in Stop mode. Multiple TPM instances are all clocked by the same TPM counter clock in support of the external timebase feature.

#### 51.5.1.1 Counter Clock Mode

The CMOD[1:0] bits in the SC register either disable the TPM counter or select one of three possible clock sources for the TPM counter. After any reset, CMOD[1:0] = 0:0 so the TPM counter is disabled. The CMOD[1:0] can configure one of the following counter clock sources.

- Asynchronous counter clock
- External clock input pin
- External trigger source

The CMOD[1:0] bits may be read or written at any time. Disabling the TPM counter by writing zero to the CMOD[1:0] bits does not affect the TPM counter value or other registers, but must be acknowledged by the TPM counter clock domain before they read as zero.

The external clock input and external trigger source pass through a synchronizer clocked by the TPM counter clock to ensure that counter transitions are properly aligned to counter clock transitions. Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must be less than half of the counter clock frequency.

## 51.5.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter.

The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and TPM counter.

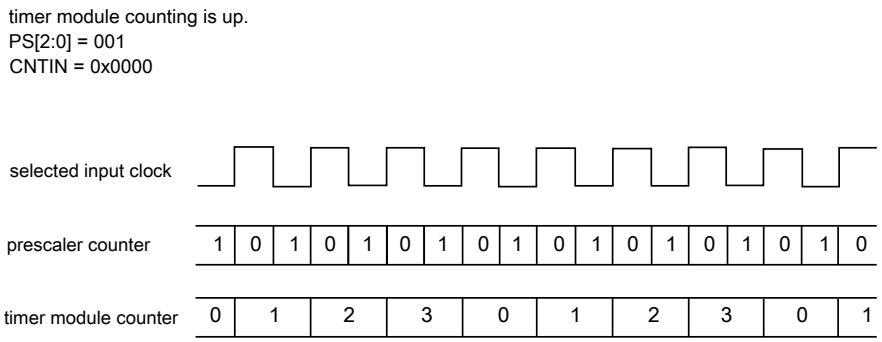



**Figure 51-2. Example of the Prescaler Counter**

## 51.5.3 Counter

The TPM has a 16-bit counter that is used by the channels either for input or output modes.

The counter updates from the selected clock divided by the prescaler.

The TPM counter has these modes of operation:

- up counting (see Up counting)
- up-down counting (see Up-down counting)

### 51.5.3.1 Up counting

Up counting is selected when SC[CPWMS] = 0.

The value of zero is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with zero.

The TPM period when using up counting is (MOD + 0x0001) × period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to zero.

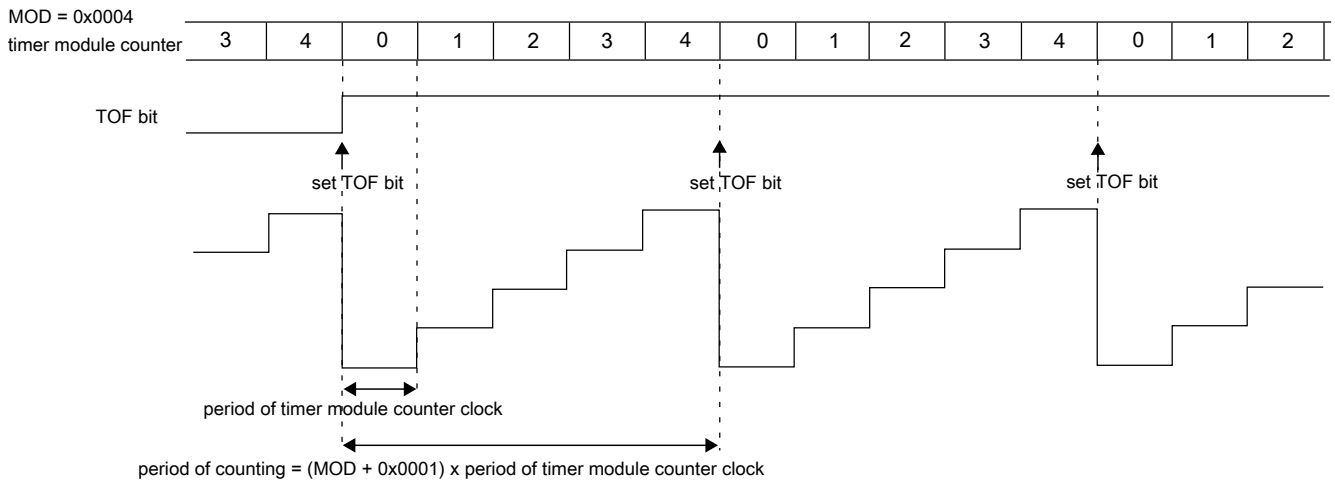



**Figure 51-3. Example of TPM Up Counting**

**Note**

- MOD = 0000 is a redundant condition. In this case, the TPM counter is always equal to MOD and the TOF bit is set in each rising edge of the TPM counter clock.

### 51.5.3.2 Up-down counting

Up-down counting is selected when SC[CPWMS] = 1. When configured for up-down counting, configuring CONF[MOD] to less than 2 is not supported.

The value of 0 is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to zero and the up-down counting restarts.

The TPM period when using up-down counting is 2 × MOD × period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to (MOD – 1).

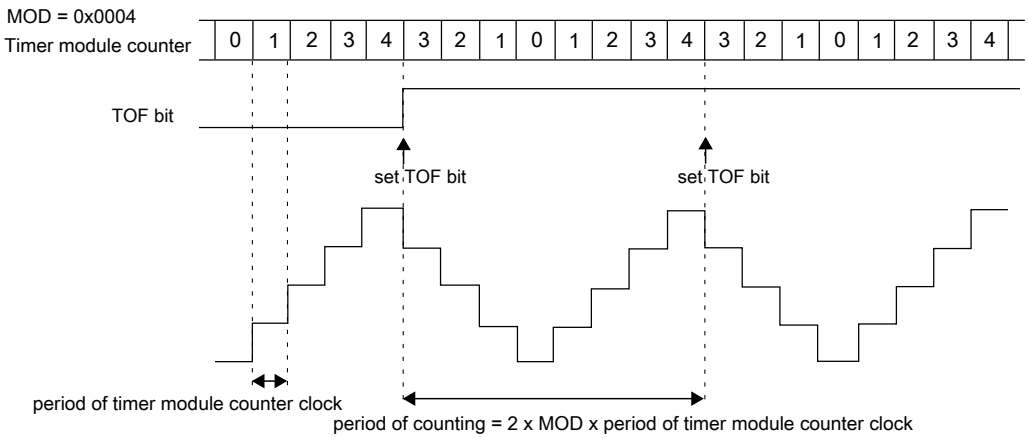



**Figure 51-4. Example of up-down counting**

### 51.5.3.3 Counter Reset

Any write to CNT resets the TPM counter and the channel outputs to their initial values (except for channels in output compare mode).

### 51.5.3.4 Global time base (GTB)

The global time base (GTB) is a TPM function that allows multiple TPM modules to share the same timebase. When the global time base is enabled (CONF[GTBEEN] = 1), the local TPM channels use the counter value, counter enable and overflow indication from the TPM generating the global time base. If the local TPM counter is not generating the global time base, then it can be used as an independent counter or pulse accumulator.

The local TPM counter can also be configured to synchronize to the global time base, by configuring (GTBSYNC = 1). When synchronized to the global time base, the local counter will use the counter enable and counter overflow indication from the TPM generating the global time base. This enables multiple TPM to be configured with the same phase, but with different periods (although the global time base must be configured with the longest period).

### 51.5.3.5 Counter trigger

The TPM counter can be configured to start, stop or reset in response to a hardware trigger input. The trigger input is synchronized to the asynchronous counter clock, so there is a 3 counter clock delay between the trigger assertion and the counter responding.

- When (CSOT = 1), the counter will not start incrementing until a rising edge is detected on the trigger input.
- When (CSOO= 1), the counter will stop incrementing whenever the TOF flag is set. The counter does not increment again unless it is disabled, or if CSOT = 1 and a rising edge is detected on the trigger input.
- When (CROT= 1), the counter will reset to zero as if an overflow occurred whenever a rising edge is detected on the trigger input. Any PWM channels will update to their reload state.
- When (CPOT = 1), the counter will pause incrementing whenever the trigger input is asserted. The counter will continue incrementing when the trigger input negates. PWM output channels are forced to their default state.

The polarity of the external input trigger can be configured by the TRGPOL register bit.

When an internal trigger source is selected, the trigger input is selected from one or more channel input capture events. The input capture filters are used with the internal trigger sources and the POLn bits can be used to invert the polarity of the input channels. Note that following restrictions apply with input capture channel sources.

- When (CSOT = 1), the counter will only start incrementing on a rising edge on the channel input, provided ELSnA = 1.
- When (CROT= 1), the counter will reset to zero on either edge of the channel input, as configured by ELSnB:ELSnA.
- When (CPOT = 1), the counter will pause incrementing whenever the channel input is asserted. PWM output channels are forced to their default state.

### 51.5.4 Input Capture Mode

The input capture mode is selected when (CPWMS = 0), (MSnB:MSnA = 0:0), and (ELSnB:ELSnA ≠ 0:0).

When a selected edge occurs on the channel input, the current value of the TPM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1 (see the following figure).

When a channel is configured for input capture, the TPM_CHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is counter clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register are ignored in input capture mode.

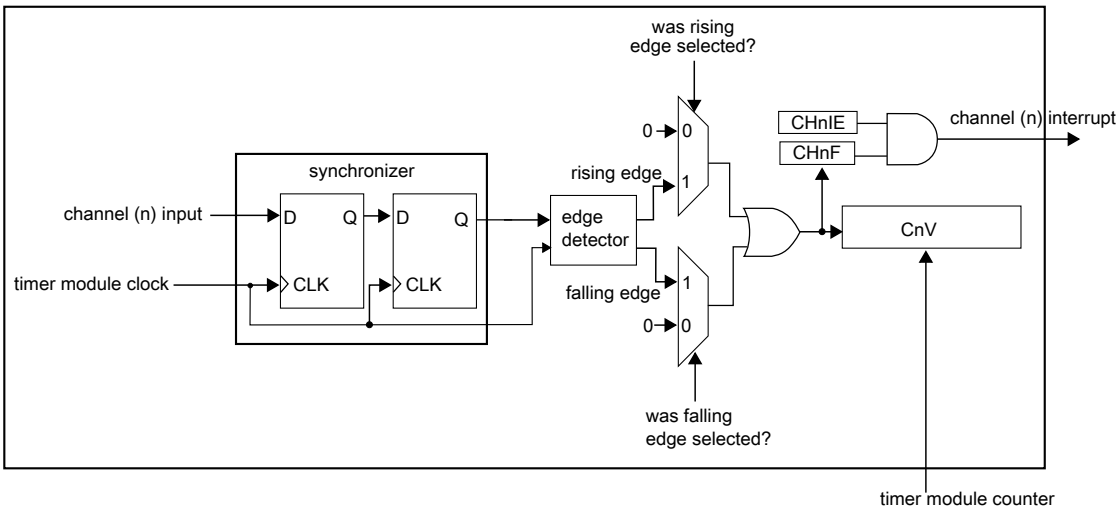



**Figure 51-5. Input capture mode**

The CHnF bit is set on the third rising edge of the counter clock after a valid edge occurs on the channel input.

## 51.5.5 Output Compare Mode

The output compare mode is selected when (CPWMS = 0), and (MSnB:MSnA = X:1).

In output compare mode, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared or toggled if MSnB is clear. If MSnB is set then the channel (n) output is pulsed high or low for as long as the counter matches the value in the CnV register.

When a channel is initially configured to output compare mode, the channel output updates with its negated value (logic 0 for set/toggle/pulse high and logic one for clear/ pulse low).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV).

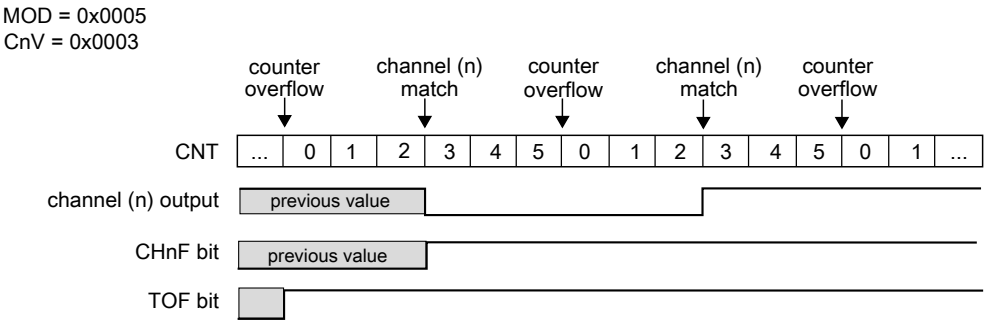



**Figure 51-6. Example of the output compare mode when the match toggles the channel output**

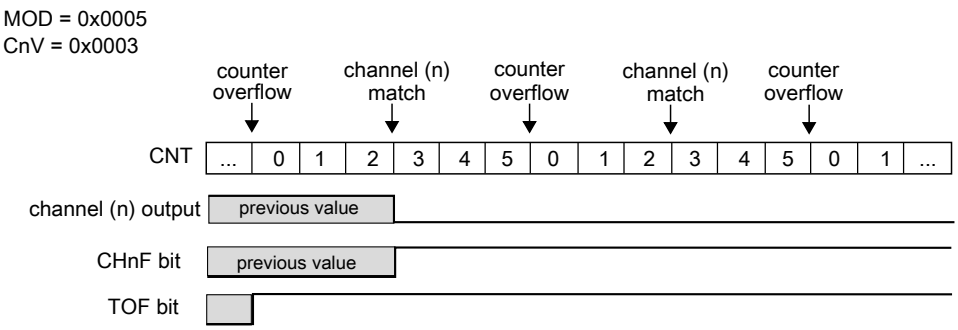



**Figure 51-7. Example of the output compare mode when the match clears the channel output**

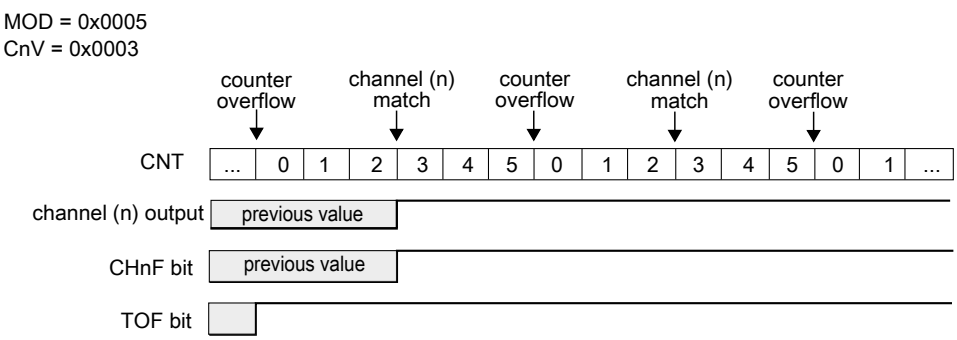



**Figure 51-8. Example of the output compare mode when the match sets the channel output**

It is possible to use the output compare mode with (ELSnB:ELSnA = 0:0). In this case, when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not modified and controlled by TPM.

## 51.5.6 Edge-Aligned PWM (EPWM) Mode

The edge-aligned mode is selected when (CPWMS = 0), and (MSnB:MSnA = 1:0).

The EPWM period is determined by (MOD + 0x0001) and the pulse width (duty cycle) is determined by CnV.

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an TPM.

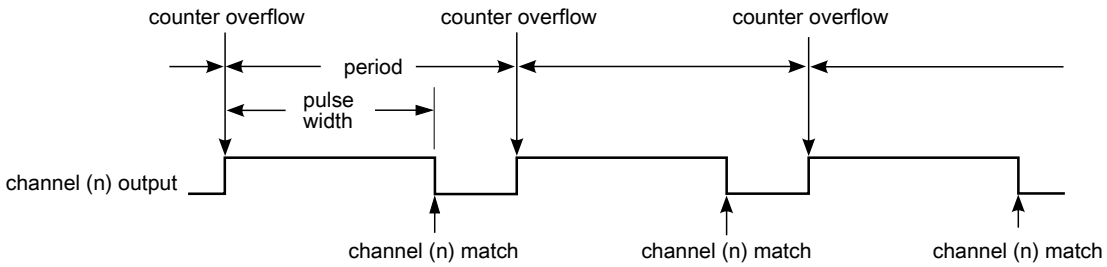



**Figure 51-9. EPWM period and pulse width with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow or reload (when the zero is loaded into the TPM counter), and it is forced low at the channel (n) match (TPM counter = CnV). When first enabled or whenever the TPM counter is paused, the channel (n) output is forced high.

If (ELSnB:ELSnA = 0:0), then the channel (n) output is forced high at the counter overflow or reload (when the zero is loaded into the TPM counter), and it is forced low at the channel (n) match (TPM counter = CnV). When first enabled or whenever the TPM counter is paused, the channel (n) output is forced low.

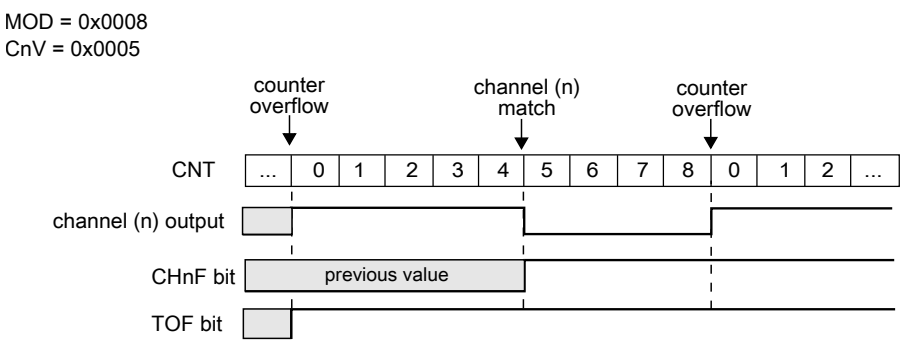



**Figure 51-10. EPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 0:1), then the channel (n) output is forced low at the counter overflow or reload (when zero is loaded into the TPM counter), and it is forced high at the channel (n) match (TPM counter = CnV). When first enabled or whenever the TPM counter is paused, the channel (n) output is forced low.

If (ELSnB:ELSnA = 1:1), then the channel (n) output is forced low at the counter overflow or reload (when zero is loaded into the TPM counter), and it is forced high at the channel (n) match (TPM counter = CnV). When first enabled or whenever the TPM counter is paused, the channel (n) output is forced high.

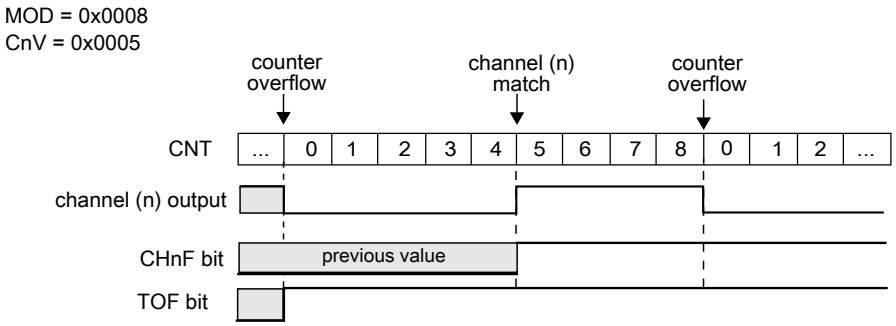



**Figure 51-11. EPWM signal with ELSnB:ELSnA = 0:1**

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal. If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set since there is never a channel (n) match. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

## 51.5.7 Center-Aligned PWM (CPWM) Mode

The center-aligned mode is selected when (CPWMS = 1) and (MSnB:MSnA = 1:0).

The CPWM pulse width (duty cycle) is determined by 2 × CnV and the period is determined by 2 × MOD (see the following figure). MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the TPM counter counts up until it reaches MOD and then counts down until it reaches zero.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV) when the TPM counting is down (at the begin of the pulse width) and when the TPM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are when the TPM counter is zero.

The other channel modes are not designed to be used with the up-down counter (CPWMS = 1). Therefore, all TPM channels should be used in CPWM mode when (CPWMS = 1).

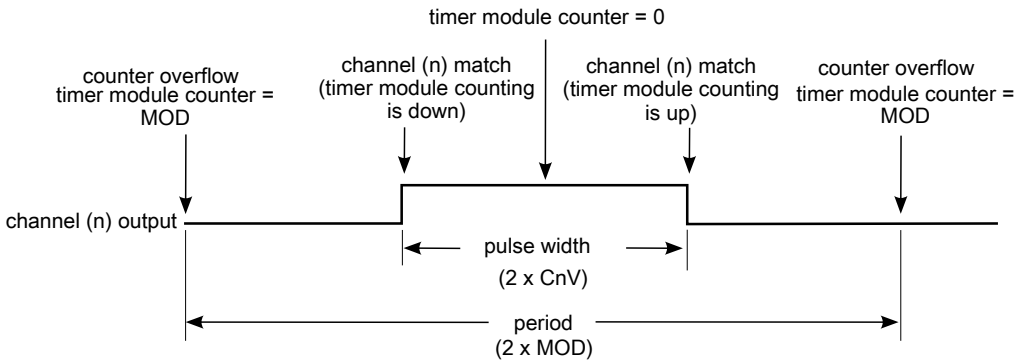



**Figure 51-12. CPWM period and pulse width with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (TPM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. When first enabled, when the TPM counter is reloaded, or whenever the TPM counter is paused, the channel (n) output is forced high.

If (ELSnB:ELSnA = 0:0), then the channel (n) output is forced high at the channel (n) match (TPM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. When first enabled, when the TPM counter is reloaded, or whenever the TPM counter is paused, the channel (n) output is forced low.

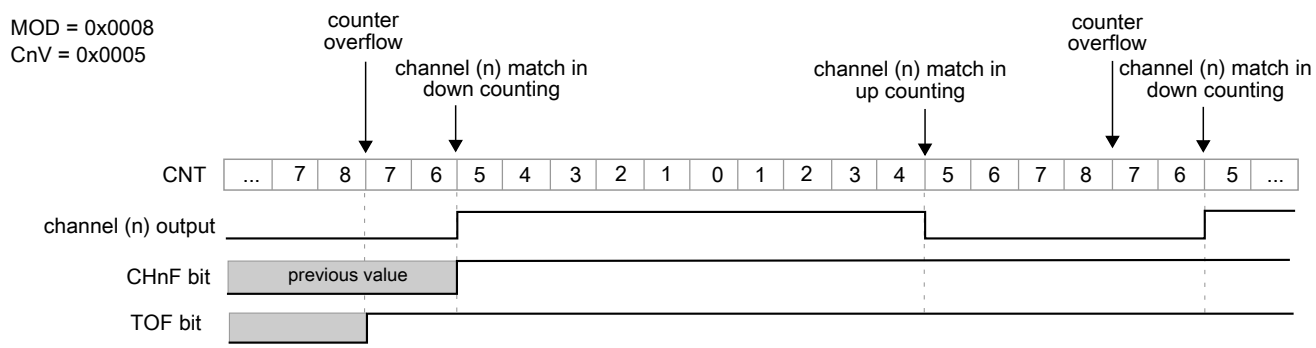



**Figure 51-13. CPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = 0:1), then the channel (n) output is forced low at the channel (n) match (TPM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. When first enabled, when the TPM counter is reloaded, or whenever the TPM counter is paused, the channel (n) output is forced low.

If (ELSnB:ELSnA = 1:1), then the channel (n) output is forced low at the channel (n) match (TPM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. When first enabled, when the TPM counter is reloaded, or whenever the TPM counter is paused, the channel (n) output is forced high.

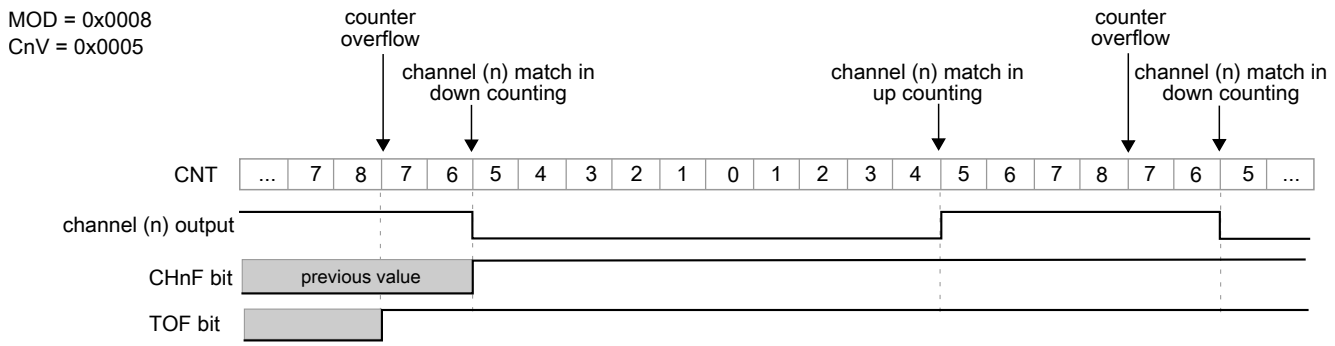



**Figure 51-14. CPWM signal with ELSnB:ELSnA = 0:1**

If (CnV = 0x0000) then the channel (n) output is a 0% duty cycle CPWM signal.

If (CnV > MOD), then the channel (n) output is a 100% duty cycle CPWM signal, although the CHnF bit is set when the counter changes from incrementing to decrementing. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle CPWM signal.

## 51.5.8 Combine PWM mode

The Combine PWM mode is selected when:

- MSnB:MSnA = 10
- COMBINEn = 1
- QUADEN = 0, and
- CPWMS = 0

In Combine PWM mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (TPM counter = C(n+1)V).

If channel (n) (ELSnB:ELSnA = 0:1), then the channel (n) output is forced low at the beginning of the period (TPM counter is zero) and at the channel (n+1) match (TPM counter = C(n+1)V). It is forced high at the channel (n) match (TPM counter = C(n)V).

If channel (n) (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the beginning of the period (TPM counter is zero) and at the channel (n+1) match (TPM counter = C(n+1)V). It is forced low at the channel (n) match (TPM counter = C(n)V).

When (COMSWAPn = 1), then the channel (n) output is forced low or high at the beginning of the period (TPM counter is zero) and at the channel (n) match (TPM counter = C(n)V). It is forced high or low at the channel (n+1) match (TPM counter = C(n+1)V).

The channel (n+1) output is generated the same as the channel (n) output, but the output polarity is controlled by the channel (n+1) ELSnB:ELSnA configuration.

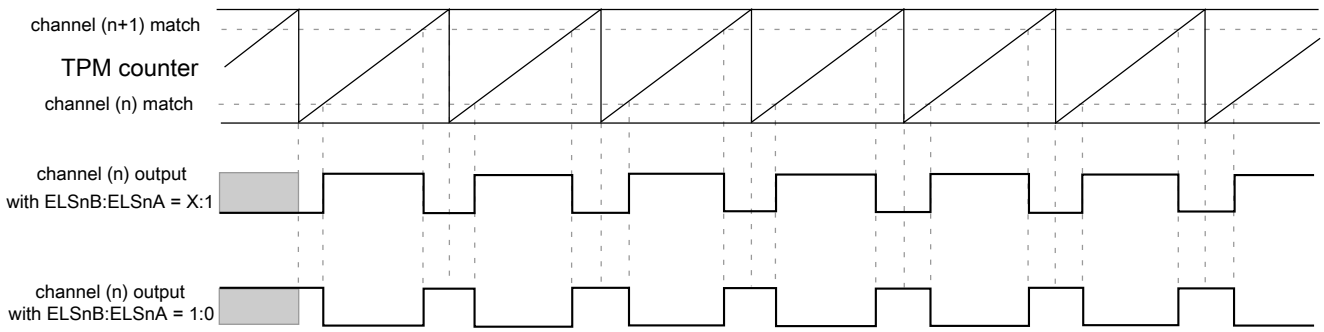



**Figure 51-15. Combine mode**

The following figures illustrate the PWM signals generation using Combine mode.

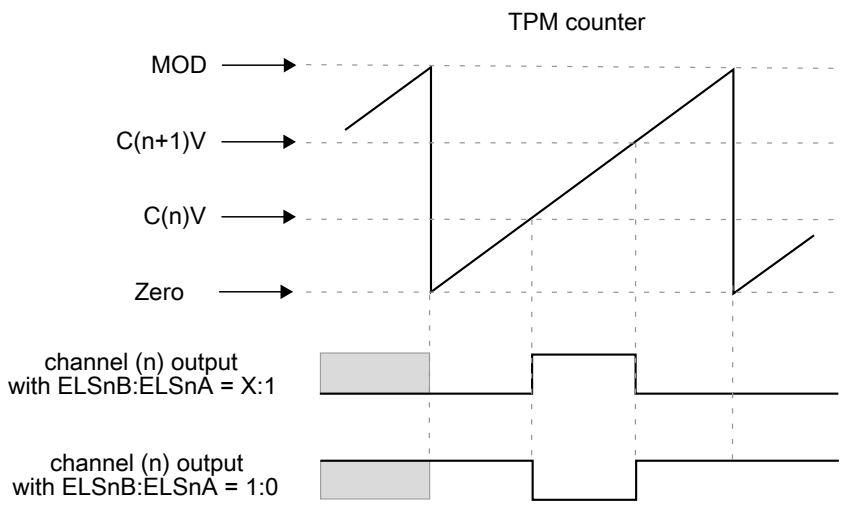



**Figure 51-16. Channel (n) output if (C(n)V < MOD) and (C(n+1)V < MOD) and (C(n)V < C(n+1)V)**

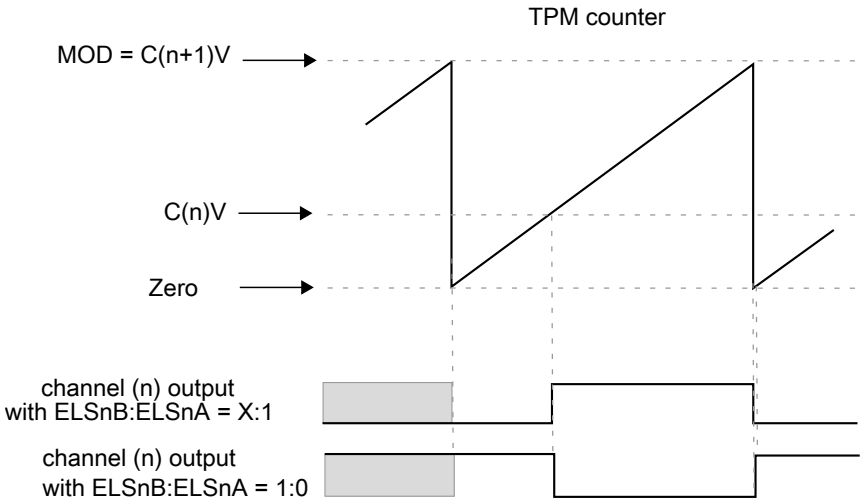



**Figure 51-17. Channel (n) output if (C(n)V < MOD) and (C(n+1)V = MOD)**

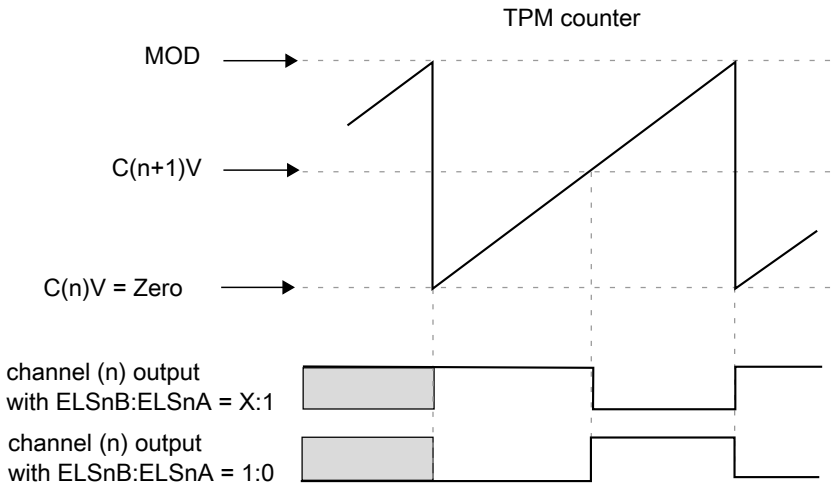



**Figure 51-18. Channel (n) output if (C(n)V = zero) and (C(n+1)V < MOD)**

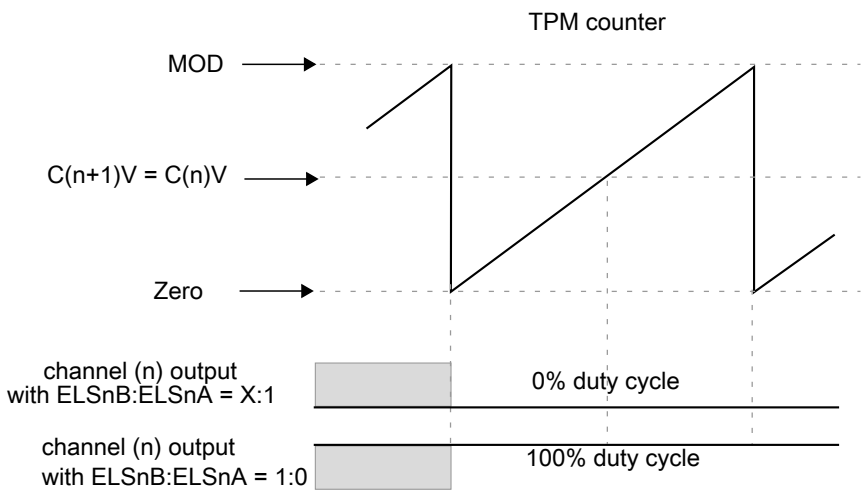



**Figure 51-19. Channel (n) output if (C(n)V < MOD) and (C(n+1)V < MOD) and (C(n)V = C(n +1)V)**

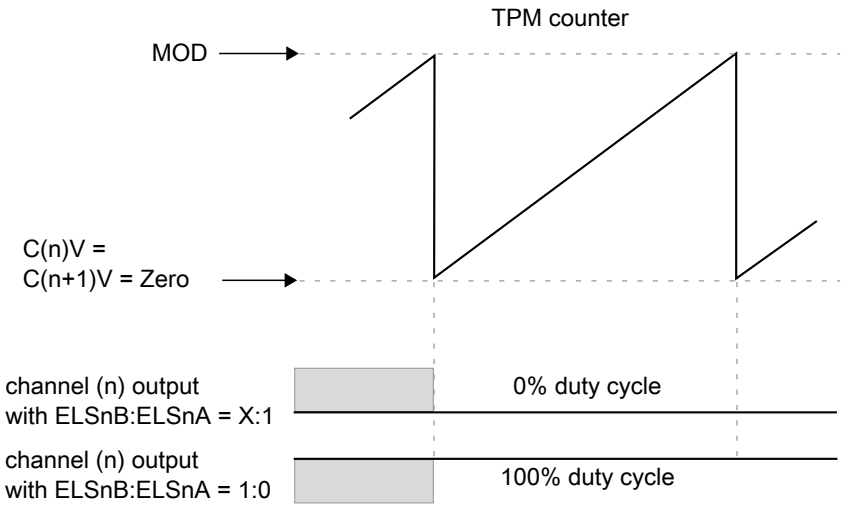



**Figure 51-20. Channel (n) output if (C(n)V = C(n+1)V = zero)**

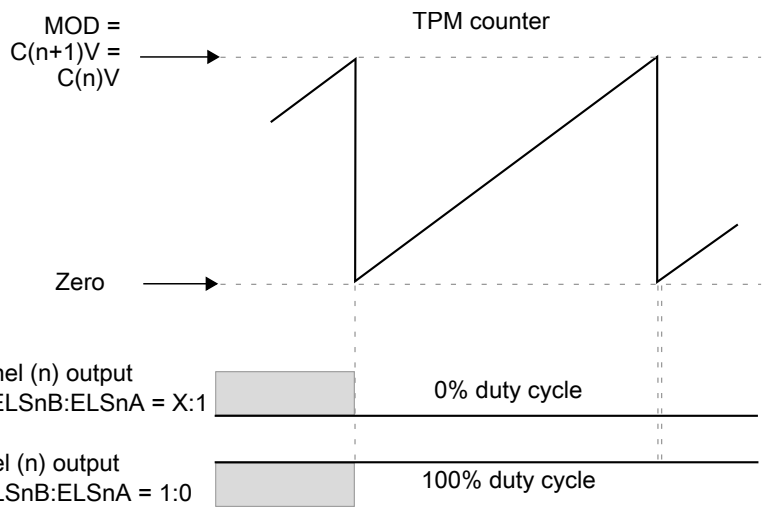



**Figure 51-21. Channel (n) output if (C(n)V = C(n+1)V = MOD)**

## 51.5.9 Combine Input Capture mode

The Combine Input Capture mode is selected if COMBINEn= 1 and MSnB:MSnA = 00 and ELSnB:ELSnA != 00. This mode allows to measure a pulse width of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode.

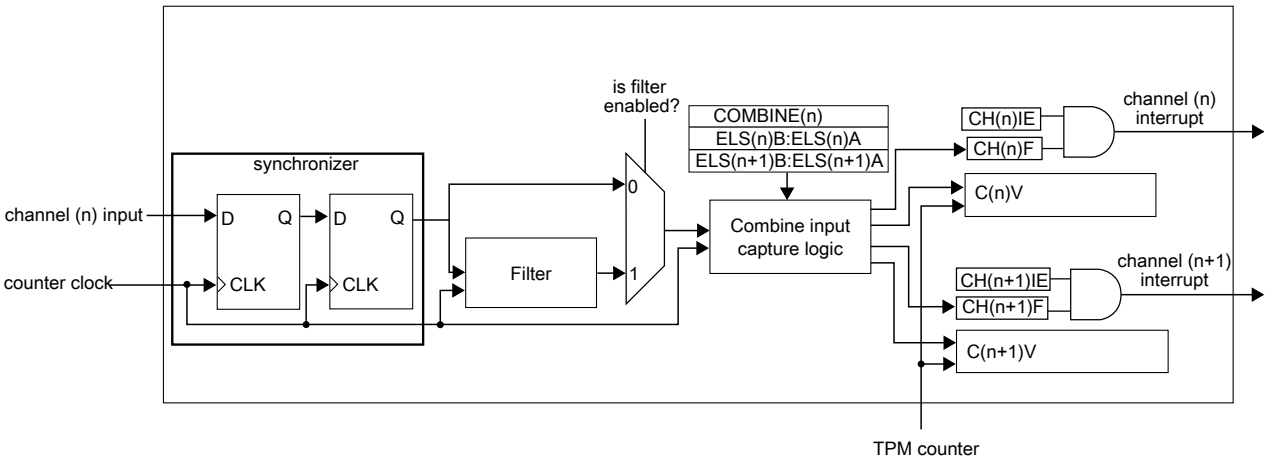



**Figure 51-22. Combine Input Capture mode block diagram**

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1).

In the Combine Input Capture mode, only channel (n) input is used and channel (n+1) input is ignored, when COMSWAPn=1 then only channel (n+1) input is used and channel (n) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input, then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of TPM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of TPM counter when the selected edge by channel (n+1) is detected at channel (n) input.

### Note

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.
- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Combine Input Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0.

## 51.5.10 Input Capture Filter

The input capture filter function is only in input capture mode, or in software compare mode when quadrature decoder mode is enabled.

First, the input signal is synchronized by the counter clock. Following synchronization, the input signal enters the filter block. See the following figure.

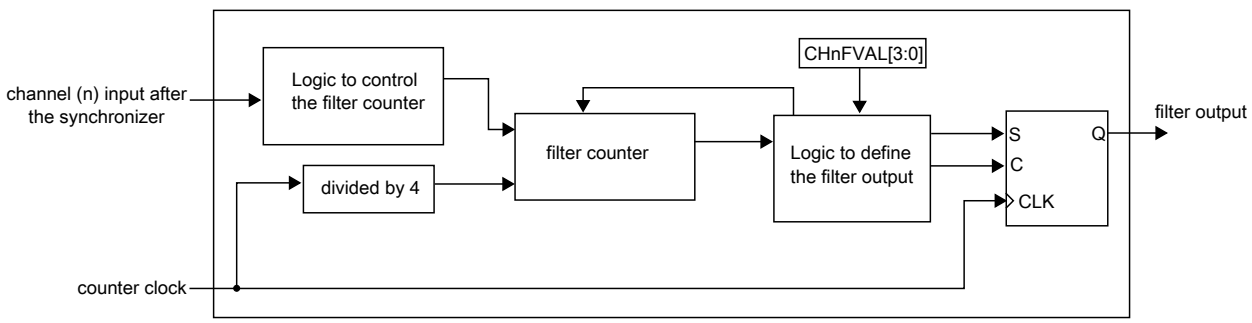



**Figure 51-23. Channel input filter**

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to (CHnFVAL[3:0] × 4), the state change of the input signal is validated.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by (CHnFVAL[3:0] × 4 counter clocks) is regarded as a glitch and is not passed through the filter. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed by 2 rising edges of the counter clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 3 rising edges of the system clock: two rising edges to the synchronizer, plus one more to the edge detector. In other words, CHnF is set (3 + 4 × CHnFVAL[3:0]) counter clock periods after a valid edge occurs on the channel input.

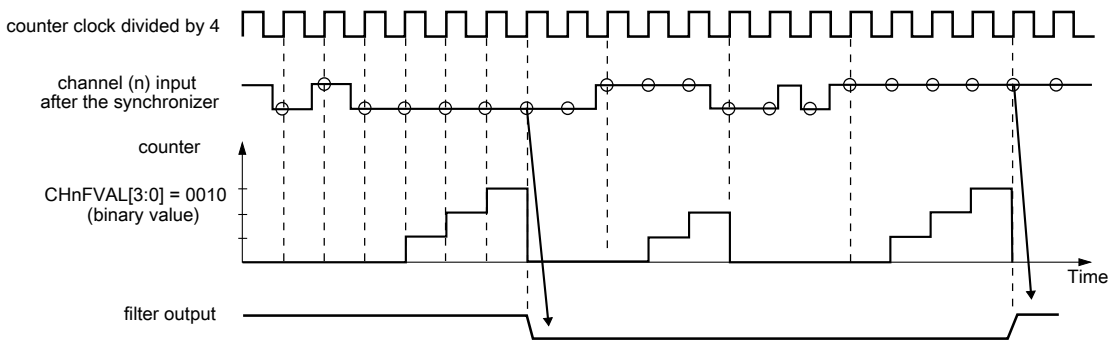



**Figure 51-24. Channel input filter example**

## 51.5.11 Deadtime insertion

The deadtime insertion is enabled in PWM combine modes for each pair of channels when CH(n)FVAL and CH(n+1)FVAL are non-zero. The deadtime delay that is used for each TPM channel is defined as (CH(n+1)FVAL[3:0] x 64) + (CH(n)FVAL[3:0] x 4).

The deadtime delay insertion ensures that no two complementary signals (even channel (n) and odd channel (n+1)) drive the active state at the same time.

If ELSnB:ELSnA = 0:1, ELS(n+1)B:ELS(n+1)A = 1:0, COMSWAPn = 0, and the deadtime is enabled, then when the channel (n) match (TPM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (TPM counter = C(n +1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If ELSnB:ELSnA = 1:0, ELS(n+1)B:ELS(n+1)A = 0:1, COMSWAPn = 1,and the deadtime is enabled, then when the channel (n+1) match (TPM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. Similarly, when the channel (n) match (TPM counter = CnV) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set.

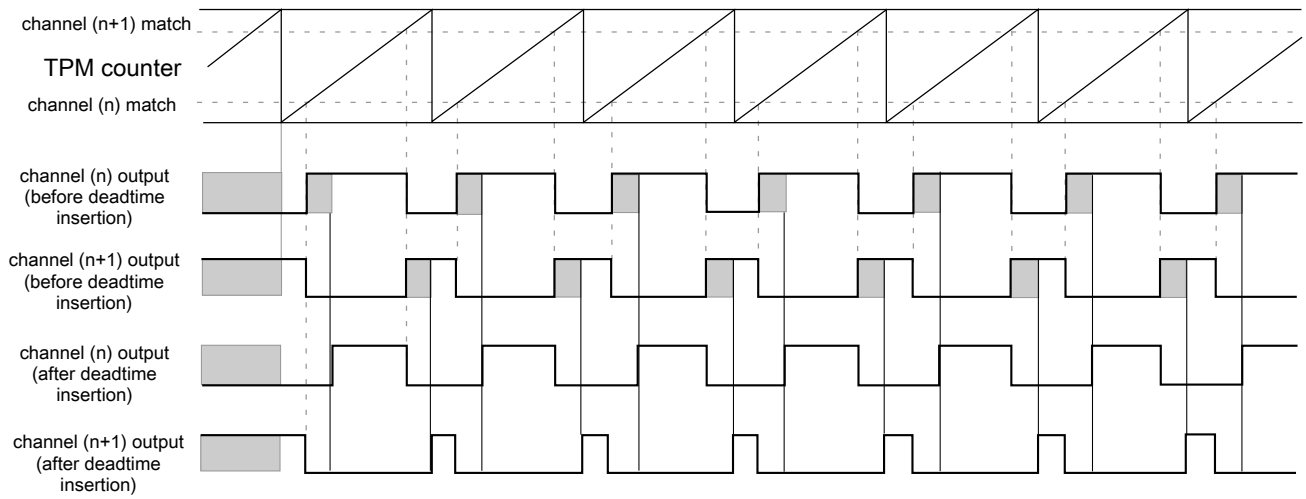



**Figure 51-25. Deadtime insertion with ELSnB:ELSnA = 0:1, ELS(n+1)B:ELS(n+1)A = 1:0, COMSWAPn = 0**

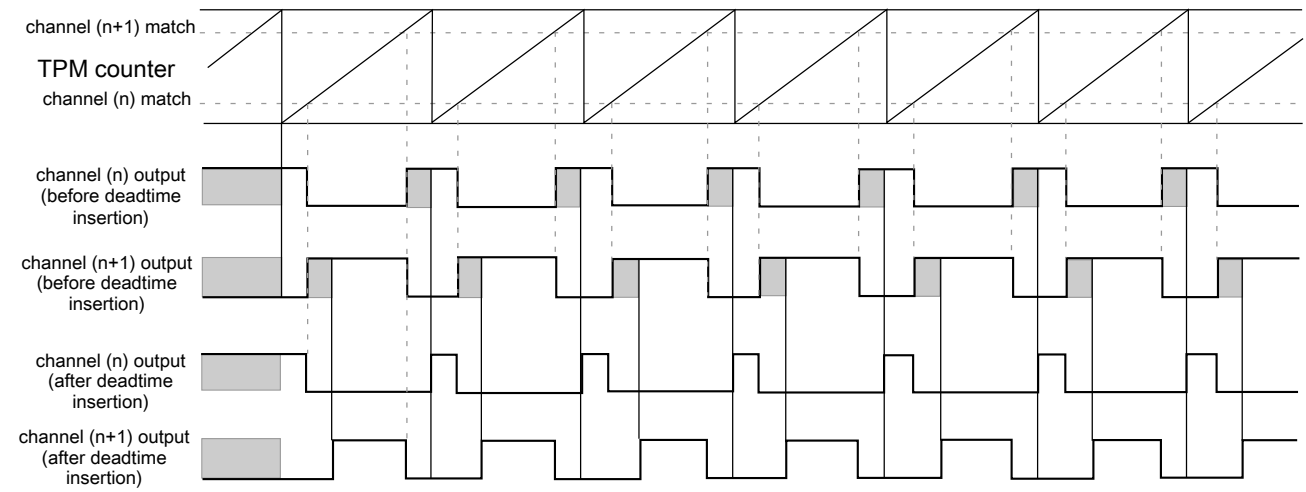



**Figure 51-26. Deadtime insertion with ELSnB:ELSnA = 1:0, ELS(n+1)B:ELS(n+1)A = 0:1, COMSWAPn = 1**

## 51.5.12 Quadrature Decoder mode

The Quadrature Decoder mode is selected if (QUADEN = 1). The Quadrature Decoder mode uses the channel 0 (phase A) and channel 1 (phase B) input signals to control the TPM counter increment and decrement. The following figure shows the quadrature decoder block diagram.

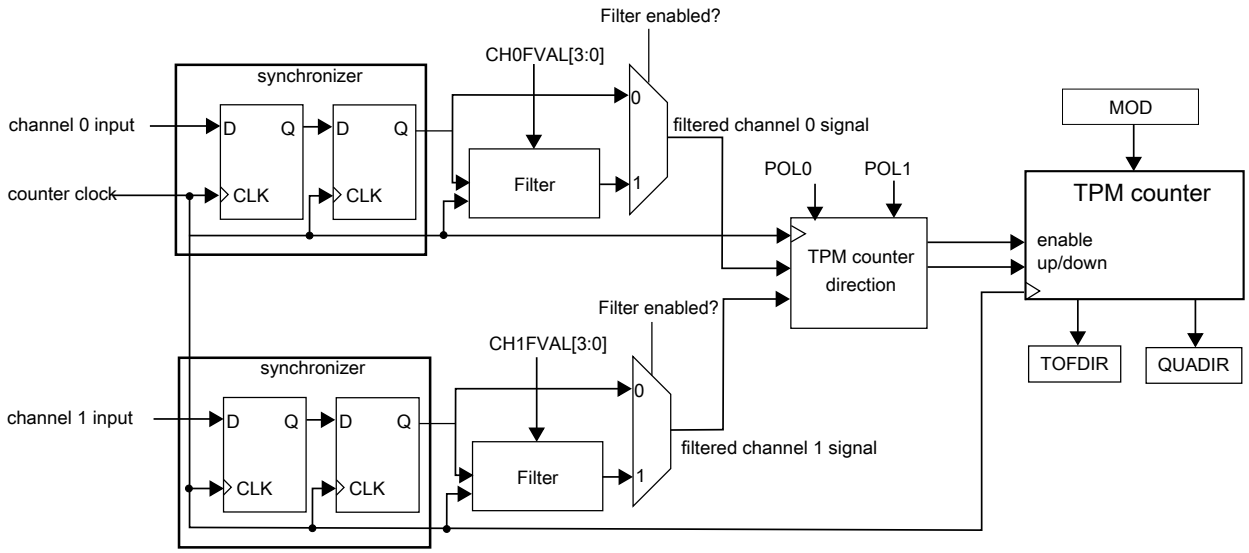



**Figure 51-27. Quadrature Decoder block diagram**

The input capture filter and channel polarity registers are used to configure the input filter and polarity for the channel 0 and channel 1 inputs in quadrature decode mode.

## Note

Notice that the TPM counter is clocked by the channel 0 and channel 1 input signals when quadrature decoder mode is selected. Therefore In quadrature decoder mode, channel 0 and channel 1 can only be used in software compare mode and other TPM channels can only be used in input capture or output compare modes.

The QUADMODE selects the encoding mode used in the Quadrature Decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled; see the following figure. In this mode, the channel 1 input value indicates the counting direction, and the channel 0 input defines the counting rate. The TPM counter is updated when there is a rising edge at channel 0 input signal.

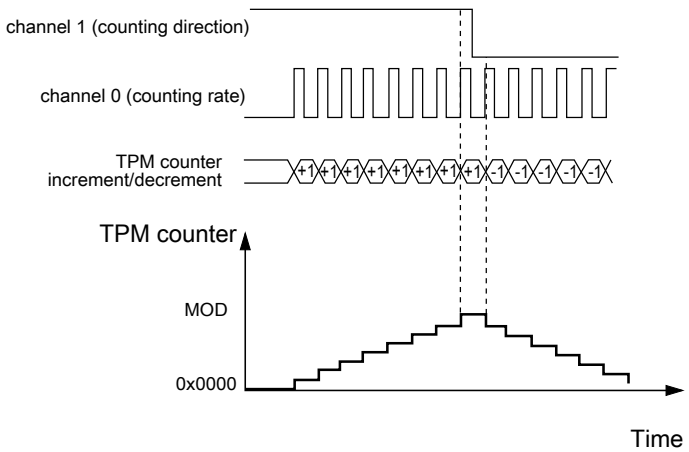



**Figure 51-28. Quadrature Decoder – Count and Direction Encoding mode**

If QUADMODE = 0, then the Phase Encoding mode is enabled; see the following figure. In this mode, the relationship between channel 0 and channel 1 signals indicates the counting direction, and channel 0 and channel 1 signals define the counting rate. The TPM counter is updated when there is an edge either at the channel 0 or channel 1 signals.

If CH0POL= 0 and CH1POL = 0, then the TPM counter increment happens when:

- there is a rising edge at channel 0 signal and channel 1 signal is at logic zero;
- there is a rising edge at channel 1 signal and channel 0 signal is at logic one;
- there is a falling edge at channel 1 signal and channel 0 signal is at logic zero;
- there is a falling edge at channel 0 signal and channel 1 signal is at logic one;

and the TPM counter decrement happens when:

- there is a falling edge at channel 0 signal and channel 1 signal is at logic zero;
- there is a falling edge at channel 1 signal and channel 0 signal is at logic one;
- there is a rising edge at channel 1 signal and channel 0 signal is at logic zero;
- there is a rising edge at channel 0 signal and channel 1 signal is at logic one.

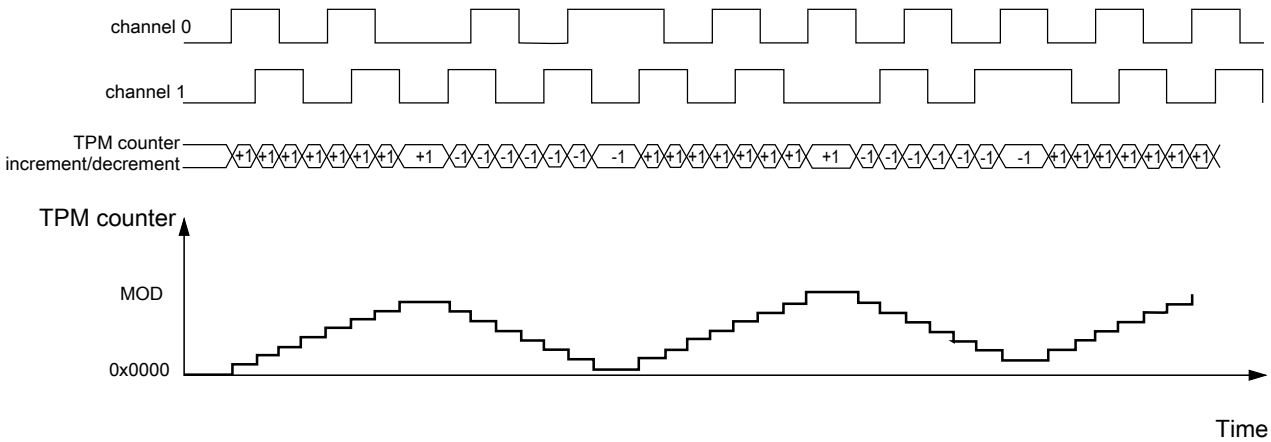



**Figure 51-29. Quadrature Decoder – Phase Encoding mode**

The following figure shows the TPM counter overflow in up counting. In this case, when the TPM counter changes from MOD to zero, TOF and TOFDIR bits are set. TOF bit indicates the TPM counter overflow occurred. TOFDIR indicates the counting was up when the TPM counter overflow occurred.

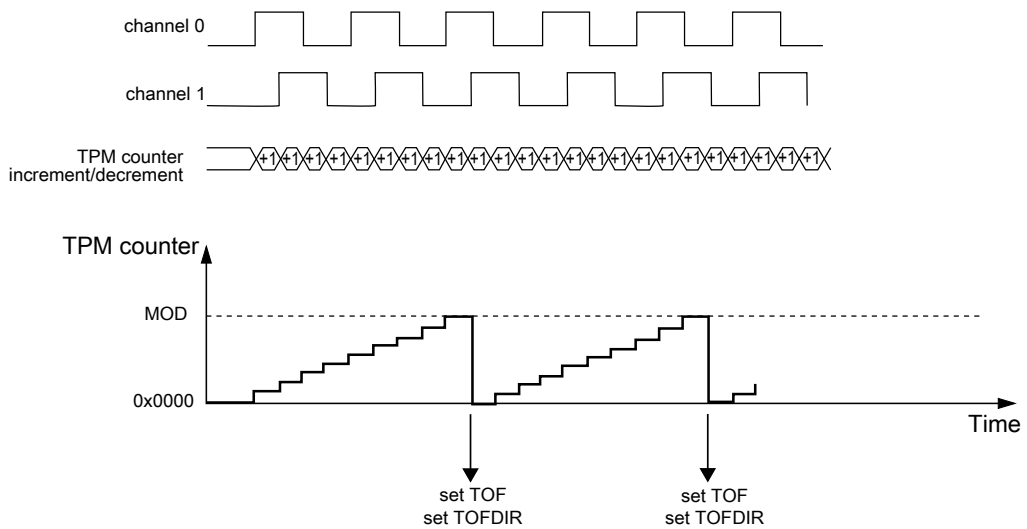



**Figure 51-30. TPM Counter overflow in up counting for Quadrature Decoder mode**

The following figure shows the TPM counter overflow in down counting. In this case, when the TPM counter changes from zero to MOD, TOF bit is set and TOFDIR bit is cleared. TOF bit indicates the TPM counter overflow occurred. TOFDIR indicates the counting was down when the TPM counter overflow occurred.

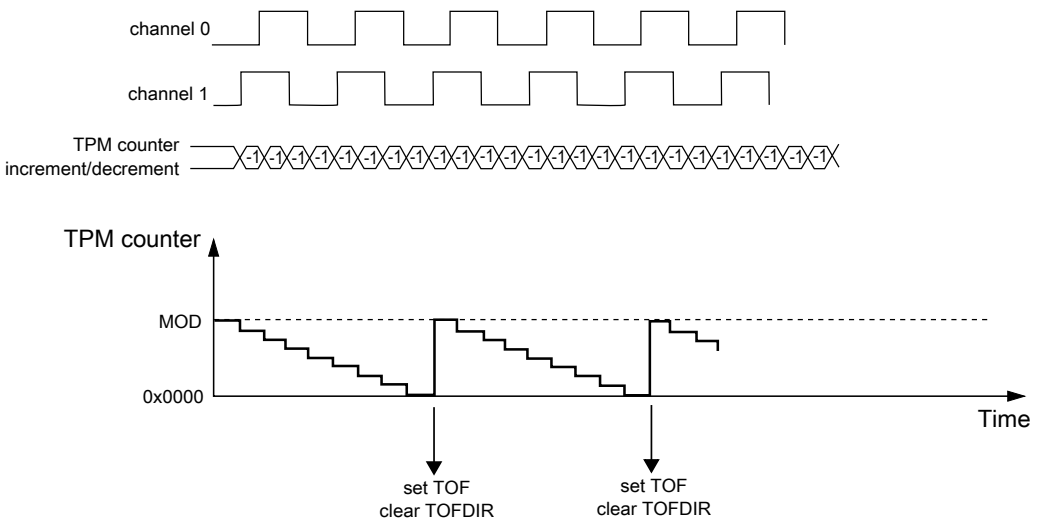



**Figure 51-31. TPM counter overflow in down counting for Quadrature Decoder mode**

## 51.5.13 Registers Updated from Write Buffers

### 51.5.13.1 MOD Register Update

If (CMOD[1:0] = 0:0) then MOD register is updated when MOD register is written.

If (CMOD[1:0] ≠ 0:0), then MOD register is updated according to the CPWMS bit, that is:

- If the selected mode is not CPWM then MOD register is updated after MOD register was written and the TPM counter changes from MOD to zero.
- If the selected mode is CPWM then MOD register is updated after MOD register was written and the TPM counter changes from MOD to (MOD – 1).

### 51.5.13.2 CnV Register Update

If (CMOD[1:0] = 0:0) then CnV register is updated when CnV register is written.

If (CMOD[1:0] ≠ 0:0), then CnV register is updated according to the selected mode, that is:

- If the selected mode is output compare then CnV register is updated on the next TPM counter increment (end of the prescaler counting) after CnV register was written.

- If the selected mode is EPWM then CnV register is updated after CnV register was written and the TPM counter changes from MOD to zero.
- If the selected mode is CPWM then CnV register is updated after CnV register was written and the TPM counter changes from MOD to (MOD – 1).

## 51.5.14 DMA

The channel and overflow flags generate a DMA transfer request according to DMA and CHnIE/TOIE bits.

See the following table for more information.

**Table 51-7. DMA Transfer Request**

| DMA | CHnIE/ TOIE | Channel/Overflow DMA Transfer Request | Channel/Overflow Interrupt |
|---|---|---|---|
| 0 | 0 | The channel/overflow DMA transfer request is not generated. | The channel/overflow interrupt is not generated. |
| 0 | 1 | The channel/overflow DMA transfer request is not generated. | The channel/overflow interrupt is generated if (CHnF/TOF = 1). |
| 1 | 0 | The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1). | The channel/overflow interrupt is not generated. |
| 1 | 1 | The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1). | The channel/overflow interrupt is generated if (CHnF/TOF = 1). |

If DMA = 1, the CHnF/TOF bit can be cleared either by DMA transfer done or writing a one to CHnF/TOF bit (see the following table).

**Table 51-8. Clear CHnF/TOF Bit**

| DMA | How CHnF/TOF Bit Can Be Cleared |
|---|---|
| 0 | CHnF/TOF bit is cleared by writing a 1 to CHnF/TOF bit. |
| 1 | CHnF/TOF bit is cleared either when the DMA transfer is done or by writing a 1 to CHnF/TOF bit. |

## 51.5.15 Output triggers

The TPM generates output triggers for the counter and each channel that can be used to trigger events in other peripherals.

The counter trigger asserts whenever the TOF is set and remains asserted until the next increment.

Each TPM channel generates both a pre-trigger output and a trigger output. The pre-trigger output asserts whenever the CHnF is set, the trigger output asserts on the first counter increment after the pre-trigger asserts, and then both the trigger and pre-trigger negate on the first counter increment after the trigger asserts.

When (COMBINEn = 1) in output compare modes, the pre-trigger output for both channel (n) and channel (n+1) will assert when CH(n)F is set and will negate when CH(n +1)F is set. The trigger continues to assert on the first counter increment after the pre-trigger asserts and negates at the same time as the pre-trigger negation.

## 51.5.16 Reset Overview

The TPM is reset whenever any chip reset occurs.

When the TPM exits from reset:

- the TPM counter and the prescaler counter are zero and are stopped (CMOD[1:0] = 0:0);
- the timer overflow interrupt is zero;
- the channels interrupts are zero;
- the channels are in input capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by TPM (ELS(n)B:ELS(n)A = 0:0).

## 51.5.17 TPM Interrupts

This section describes TPM interrupts.

### 51.5.17.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

### 51.5.17.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

# Chapter 52
# Time Stamp Timer Module (TSTMR)

## 52.1 Chip-specific TSTMR information

This module can be enabled or disabled by FTFE_FOPT[TSTMR_EN].

## 52.2 Introduction

The Time Stamp Timer (TSTMR) is a 56-bit clock cycle counter, reset by system reset.

### 52.2.1 Features

- Free-running Time Stamp Timer

## 52.3 TSTMR Memory map and register definition

The TSTMR Memory Map/Register Definition can be found here.

Each set of TSTMR registers can be accessed from the Processor it is associated with. For example, if there is a TSTMRA set of registers, then those can be accessed from ProcessorA.

**NOTE**

TSTMR registers can be read with 32-bit accesses only.

### 52.3.1 TSTMRA register descriptions

This section contains the detailed register descriptions for the TSTMRA registers.

### 52.3.1.1 TSTMRA Memory map

TSTMRA base address: 4003_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Time Stamp Timer Register Low (LOW) | 32 | RO | 0000_0000h |
| 4h | Time Stamp Timer Register High (HIGH) | 32 | RO | 0000_0000h |

### 52.3.1.2 Time Stamp Timer Register Low (LOW)

#### 52.3.1.2.1 Offset

| Register | Offset |
|---|---|
| LOW | 0h |

#### 52.3.1.2.2 Function

The Time Stamp Timer is a 56-bit clock cycle counter, reset by system reset. It is readable by way of the TSTMR HIGH and LOW registers. Only 32-bit read accesses are allowed. Any other access will generate a transfer error.

Note the following:

- A complete read operation should include both TSTMR LOW and HIGH reads. If a HIGH read does not follow a LOW read, the resulting read value is unknown.
- The TSTMR LOW read should occur first, followed by the TSTMR HIGH read.

#### 52.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 52.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>VALUE | Time Stamp Timer Low<br>Lower 32 bits of the 56-bit time stamp value. The software must read the LOW first, followed by a read to HIGH to retrieve the complete value. The all-zero reset value may not be readable because the value will increment before it can be read by the software. |

### 52.3.1.3 Time Stamp Timer Register High (HIGH)

#### 52.3.1.3.1 Offset

| Register | Offset |
|---|---|
| HIGH | 4h |

#### 52.3.1.3.2 Function

The Time Stamp Timer is a 56-bit counters clock cycle counter, reset by system reset. It is readable by way of the TSTMR HIGH and LOW registers. Only 32-bit read accesses are allowed. Any other access will generate a transfer error.

Note the following:

- A complete read operation should include both TSTMR LOW and HIGH reads. If a HIGH read does not follow a LOW read, the resulting read value is unknown.
- The TSTMR LOW read should occur first, followed by the TSTMR HIGH read.

#### 52.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | VALUE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 52.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>VALUE | Time Stamp Timer High<br>Upper 24 bits of the 56-bit time stamp value. The software must read the LOW first, followed by a read to HIGH to retrieve the complete value. The all-zero reset value may not be readable because the value will increment before it can be read by the software. |

## 52.3.2 TSTMRB register descriptions

This section contains the detailed register descriptions for the TSTMRB registers.

### 52.3.2.1 TSTMRB Memory map

TSTMRB base address: 4102_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Time Stamp Timer Register Low (LOW) | 32 | RO | 0000_0000h |
| 4h | Time Stamp Timer Register High (HIGH) | 32 | RO | 0000_0000h |

## 52.3.2.2 Time Stamp Timer Register Low (LOW)

### 52.3.2.2.1 Offset

| Register | Offset |
|---|---|
| LOW | 0h |

### 52.3.2.2.2 Function

The Time Stamp Timer is a 56-bit clock cycle counter, reset by system reset. It is readable by way of the TSTMR HIGH and LOW registers. Only 32-bit read accesses are allowed. Any other access will generate a transfer error.

Note the following:

- A complete read operation should include both TSTMR LOW and HIGH reads. If a HIGH read does not follow a LOW read, the resulting read value is unknown.
- The TSTMR LOW read should occur first, followed by the TSTMR HIGH read.

### 52.3.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 52.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>VALUE | Time Stamp Timer Low<br>Lower 32 bits of the 56-bit time stamp value. The software must read the LOW first, followed by a read to HIGH to retrieve the complete value. The all-zero reset value may not be readable because the value will increment before it can be read by the software. |

## 52.3.2.3 Time Stamp Timer Register High (HIGH)

### 52.3.2.3.1 Offset

| Register | Offset |
|---|---|
| HIGH | 4h |

### 52.3.2.3.2 Function

The Time Stamp Timer is a 56-bit counters clock cycle counter, reset by system reset. It is readable by way of the TSTMR HIGH and LOW registers. Only 32-bit read accesses are allowed. Any other access will generate a transfer error.

Note the following:

- A complete read operation should include both TSTMR LOW and HIGH reads. If a HIGH read does not follow a LOW read, the resulting read value is unknown.
- The TSTMR LOW read should occur first, followed by the TSTMR HIGH read.

### 52.3.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | VALUE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VALUE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 52.3.2.3.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>VALUE | Time Stamp Timer High<br>Upper 24 bits of the 56-bit time stamp value. The software must read the LOW first, followed by a read to HIGH to retrieve the complete value. The all-zero reset value may not be readable because the value will increment before it can be read by the software. |

## 52.4 Functional description

The TSTMR module is a free running incrementing counter that starts running after system reset de-assertion and can be read at any time by the software for determining the software ticks. However, the software must follow the read sequence as mentioned in the TSTMR register descriptions for correctly reading the TSTMR value. The TSTMR is a 56-bit counter and hence requires two 32-bit reads to read the full value. The TSTMR runs off the 1 MHz clock and resets on every system reset. The counter only stops when the clock to the TSTMR is disabled.

See the chip-specific TSTMR information for implementation details of this module's instances.

# Chapter 53
# Low-Power Timer (LPTMR)

## 53.1 Chip-specific LPTMR information

### 53.1.1 LPTMRx prescaler/glitch filter clocking options

The prescaler and glitch filter of the LPTMR module can be clocked from one of four sources determined by the LPTMRx_PSR[PCS] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

**NOTE**

The chosen clock must remain enabled if the LPTMR is to continue operating in all required low-power modes.

The LPTMR allows the maximum clock frequency of 24 MHz.

| LPTMRx_PSR[PCS] | Prescaler/glitch filter clock number | Chip clock |
|---|---|---|
| 00 | 0 | SIRC DIV3 |
| 01 | 1 | LPO — 1 kHz clock |
| 10 | 2 | RTC — 32 kHz clock |
| 11 | 3 | RFOSC |

See Clock Distribution for more details on these clocks.

### 53.1.2 LPTMRx pulse counter input options

The LPTMRx_CSR[TPS] bitfield configures the input source used in pulse counter mode. The following table shows the chip-specific input assignments for this bitfield for LPTMR0 and LPTMR1.

| LPTMR0/1_CSR[TPS] | Pulse counter input number | Chip input |
|---|---|---|
| 00 | 0 | LPCMP0 output |
| 01 | 1 | LPTMR0/1_ALT1 pin |
| 10 | 2 | LPTMR0/1_ALT2 pin |
| 11 | 3 | LPTMR0/1_ALT3 pin |

The following table shows the chip-specific input assignments for this bitfield for LPTMR2.

| LPTMR2_CSR[TPS] | Pulse counter input number | Chip input |
|---|---|---|
| 00 | 0 | LPCMP1 output |
| 01 | 1 | LPTMR2_ALT1 pin |
| 10 | 2 | LPTMR2_ALT2 pin |
| 11 | 3 | LPTMR2_ALT3 pin |

## 53.2 Introduction

The low-power timer (LPTMR) can be configured to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter, across all power modes, including the low-leakage modes. It can also continue operating through most system reset events, allowing it to be used as a time of day counter.

### 53.2.1 Features

The features of the LPTMR module include:

- 32-bit time counter or pulse counter with compare
  - Optional interrupt can generate asynchronous wakeup from any low-power mode
  - Hardware trigger output
  - Counter supports free-running mode or reset on compare
- Configurable clock source for prescaler/glitch filter
- Configurable input source for pulse counter
  - Rising-edge or falling-edge

### 53.2.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

**Table 53-1. Modes of operation**

| Modes | Description |
|---|---|
| Run | The LPTMR operates normally. |
| Wait | The LPTMR continues to operate normally and can be configured to exit the low-power mode by generating an interrupt request. |
| Stop | The LPTMR continues to operate normally and can be configured to exit the low-power mode by generating an interrupt request. |
| Low-Leakage | The LPTMR continues to operate normally and can be configured to exit the low-power mode by generating an interrupt request. |
| Debug | The LPTMR operates normally in Pulse Counter mode, but the counter does not increment in Time Counter mode. |

## 53.3 LPTMR signal descriptions

**Table 53-2. LPTMR signal descriptions**

| Signal | I/O | Description |
|---|---|---|
| LPTMR_ALT*n* | I | Pulse Counter Input pin |

### 53.3.1 Detailed signal descriptions

**Table 53-3. LPTMR interface—detailed signal descriptions**

| Signal | I/O | Description | |
|---|---|---|---|
| LPTMR_ALT*n* | I | Pulse Counter Input<br>The LPTMR can select one of the input pins to be used in Pulse Counter mode. | |
| | | State meaning | Assertion—If configured for pulse counter mode with active-high input, then assertion causes the CNR to increment.<br>Deassertion—If configured for pulse counter mode with active-low input, then deassertion causes the CNR to increment. |
| | | Timing | Assertion or deassertion may occur at any time; input may assert asynchronously to the bus clock. |

## 53.4 Memory map and register definition

**NOTE**

The LPTMR registers are reset only on a POR or LVD event.
See LPTMR power and reset for more details.

### 53.4.1 LPTMR0 register descriptions

#### 53.4.1.1 LPTMR0 Memory map

LPTMR0 base address: 4003_2000h

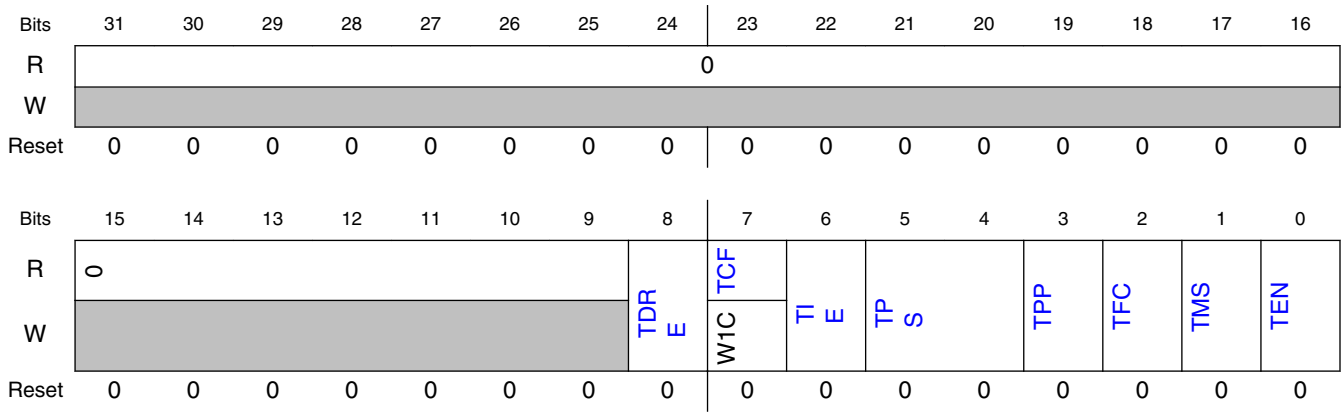
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Low Power Timer Control Status Register (CSR) | 32 | RW | 0000_0000h |
| 4h | Low Power Timer Prescale Register (PSR) | 32 | RW | 0000_0000h |
| 8h | Low Power Timer Compare Register (CMR) | 32 | RW | 0000_0000h |
| Ch | Low Power Timer Counter Register (CNR) | 32 | RW | 0000_0000h |

#### 53.4.1.2 Low Power Timer Control Status Register (CSR)

##### 53.4.1.2.1 Offset

| Register | Offset |
|---|---|
| CSR | 0h |

### 53.4.1.2.2 Diagram




### 53.4.1.2.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TDRE | Timer DMA Request Enable<br>When TDRE is set, the LPTMR DMA Request is generated whenever TCF is also set and the TCF is cleared when the DMA Controller is done.<br>0b - Timer DMA Request disabled.<br>1b - Timer DMA Request enabled. |
| 7<br>TCF | Timer Compare Flag<br>TCF is set when the LPTMR is enabled and the CNR equals the CMR and increments. TCF is cleared when the LPTMR is disabled or a logic 1 is written to it.<br>0b - The value of CNR is not equal to CMR and increments.<br>1b - The value of CNR is equal to CMR and increments. |
| 6<br>TIE | Timer Interrupt Enable<br>When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set.<br>0b - Timer interrupt disabled.<br>1b - Timer interrupt enabled. |
| 5-4<br>TPS | Timer Pin Select<br>Configures the input source to be used in Pulse Counter mode. TPS must be altered only when the LPTMR is disabled. The input connections vary by device. See the chip configuration information about connections to these inputs.<br>00b - Pulse counter input 0 is selected.<br>01b - Pulse counter input 1 is selected.<br>10b - Pulse counter input 2 is selected.<br>11b - Pulse counter input 3 is selected. |
| 3<br>TPP | Timer Pin Polarity<br>Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled.<br>0b - Pulse Counter input source is active-high, and the CNR will increment on the rising-edge.<br>1b - Pulse Counter input source is active-low, and the CNR will increment on the falling-edge. |

*Table continues on the next page...*

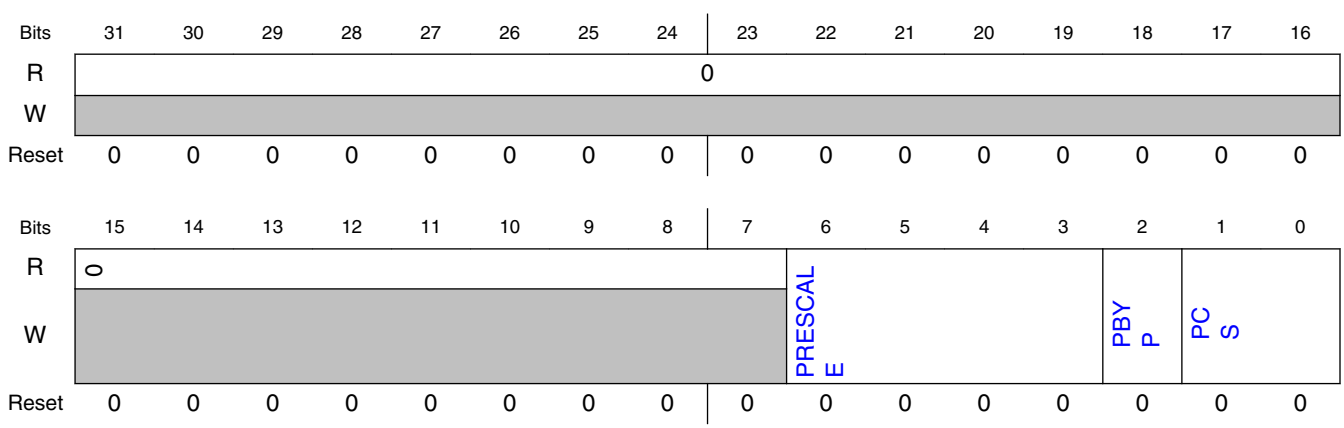
| Field | Function |
|---|---|
| 2<br>TFC | Timer Free-Running Counter<br>When clear, TFC configures the CNR to reset whenever TCF is set. When set, TFC configures the CNR to reset on overflow. TFC must be altered only when the LPTMR is disabled.<br>0b - CNR is reset whenever TCF is set.<br>1b - CNR is reset on overflow. |
| 1<br>TMS | Timer Mode Select<br>Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled.<br>0b - Time Counter mode.<br>1b - Pulse Counter mode. |
| 0<br>TEN | Timer Enable<br>When TEN is clear, it resets the LPTMR internal logic, including the CNR and TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this field, CSR[5:1] must not be altered.<br>0b - LPTMR is disabled and internal logic is reset.<br>1b - LPTMR is enabled. |

## 53.4.1.3 Low Power Timer Prescale Register (PSR)

### 53.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PSR | 4h |

### 53.4.1.3.2 Diagram



### 53.4.1.3.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-3<br>PRESCALE | Prescale Value<br>Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. The width of the glitch filter can vary by 1 cycle due to synchronization of the pulse counter input. PRESCALE must be altered only when the LPTMR is disabled.<br>0000b - Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration.<br>0001b - Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges.<br>0010b - Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges.<br>0011b - Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges.<br>0100b - Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges.<br>0101b - Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges.<br>0110b - Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges.<br>0111b - Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges.<br>1000b - Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges.<br>1001b - Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges.<br>1010b - Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges.<br>1011b - Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges.<br>1100b - Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges.<br>1101b - Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges.<br>1110b - Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges.<br>1111b - Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges. |
| 2<br>PBYP | Prescaler Bypass<br>When PBYP is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNR. When PBYP is clear, the CNR is clocked by the output of the prescaler/glitch filter. PBYP must be altered only when the LPTMR is disabled.<br>0b - Prescaler/glitch filter is enabled.<br>1b - Prescaler/glitch filter is bypassed. |
| 1-0<br>PCS | Prescaler Clock Select<br>Selects the clock to be used by the LPTMR prescaler/glitch filter. In time counter mode, this field selects the input clock to the prescaler. In pulse counter mode, this field selects the input clock to the glitch filter. PCS must be altered only when the LPTMR is disabled. The clock connections vary by device.<br>**NOTE:** See the chip configuration details for information on the connections to these inputs.<br>00b - Prescaler/glitch filter clock 0 selected.<br>01b - Prescaler/glitch filter clock 1 selected.<br>10b - Prescaler/glitch filter clock 2 selected.<br>11b - Prescaler/glitch filter clock 3 selected. |

## 53.4.1.4 Low Power Timer Compare Register (CMR)

### 53.4.1.4.1 Offset

| Register | Offset |
|---|---|
| CMR | 8h |

### 53.4.1.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 53.4.1.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPARE | Compare Value<br>When the LPTMR is enabled and the CNR equals the value in the CMR and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger will remain asserted until the LPTMR is disabled. If the LPTMR is enabled, the CMR must be altered only when TCF is set. |

## 53.4.1.5 Low Power Timer Counter Register (CNR)

### 53.4.1.5.1 Offset

| Register | Offset |
|---|---|
| CNR | Ch |

#### 53.4.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 53.4.1.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COUNTER | Counter Value<br>The CNR returns the current value of the LPTMR counter at the time this register was last written. |

## 53.4.2 LPTMR1 register descriptions

### 53.4.2.1 LPTMR1 Memory map

LPTMR1 base address: 4003_3000h

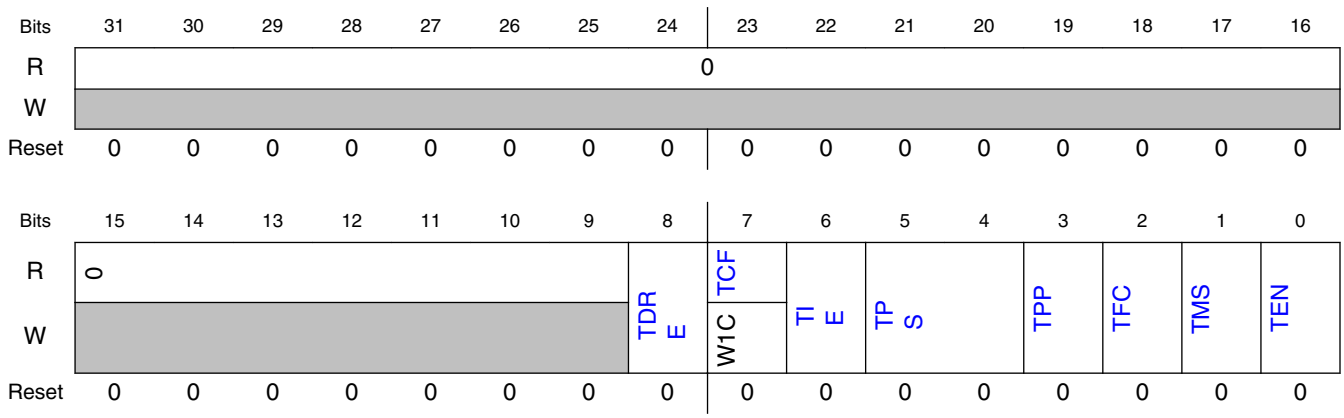
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Low Power Timer Control Status Register (CSR) | 32 | RW | 0000_0000h |
| 4h | Low Power Timer Prescale Register (PSR) | 32 | RW | 0000_0000h |
| 8h | Low Power Timer Compare Register (CMR) | 32 | RW | 0000_0000h |
| Ch | Low Power Timer Counter Register (CNR) | 32 | RW | 0000_0000h |

### 53.4.2.2 Low Power Timer Control Status Register (CSR)

### 53.4.2.2.1 Offset

| Register | Offset |
|---|---|
| CSR | 0h |

### 53.4.2.2.2 Diagram




### 53.4.2.2.3 Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TDRE | Timer DMA Request Enable<br>When TDRE is set, the LPTMR DMA Request is generated whenever TCF is also set and the TCF is cleared when the DMA Controller is done.<br>0b - Timer DMA Request disabled.<br>1b - Timer DMA Request enabled. |
| 7<br>TCF | Timer Compare Flag<br>TCF is set when the LPTMR is enabled and the CNR equals the CMR and increments. TCF is cleared when the LPTMR is disabled or a logic 1 is written to it.<br>0b - The value of CNR is not equal to CMR and increments.<br>1b - The value of CNR is equal to CMR and increments. |
| 6<br>TIE | Timer Interrupt Enable<br>When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set.<br>0b - Timer interrupt disabled.<br>1b - Timer interrupt enabled. |
| 5-4<br>TPS | Timer Pin Select<br>Configures the input source to be used in Pulse Counter mode. TPS must be altered only when the LPTMR is disabled. The input connections vary by device. See the chip configuration information about connections to these inputs.<br>00b - Pulse counter input 0 is selected. |

*Table continues on the next page...*

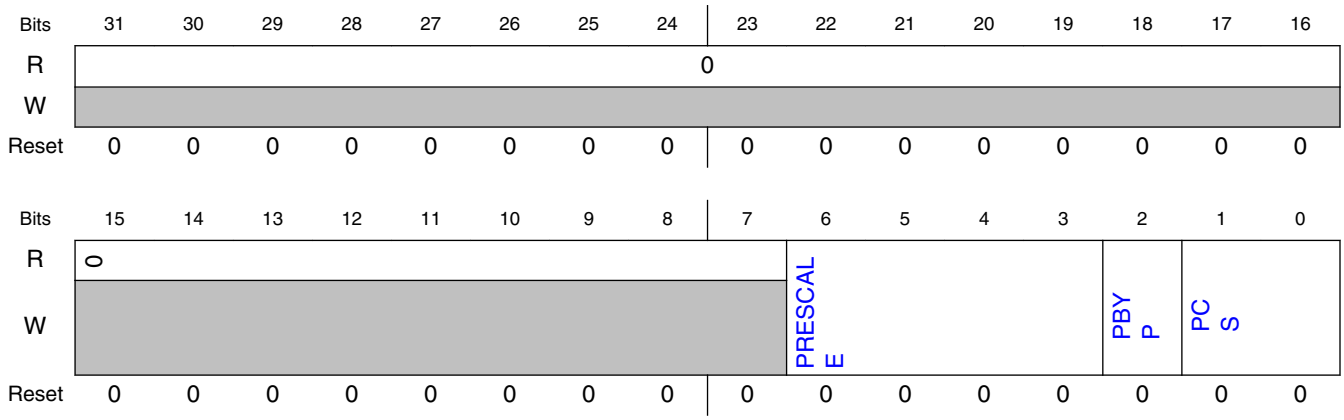
| Field | Function |
|---|---|
| | 01b - Pulse counter input 1 is selected.<br>10b - Pulse counter input 2 is selected.<br>11b - Pulse counter input 3 is selected. |
| 3<br>TPP | Timer Pin Polarity<br>Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled.<br>0b - Pulse Counter input source is active-high, and the CNR will increment on the rising-edge.<br>1b - Pulse Counter input source is active-low, and the CNR will increment on the falling-edge. |
| 2<br>TFC | Timer Free-Running Counter<br>When clear, TFC configures the CNR to reset whenever TCF is set. When set, TFC configures the CNR to reset on overflow. TFC must be altered only when the LPTMR is disabled.<br>0b - CNR is reset whenever TCF is set.<br>1b - CNR is reset on overflow. |
| 1<br>TMS | Timer Mode Select<br>Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled.<br>0b - Time Counter mode.<br>1b - Pulse Counter mode. |
| 0<br>TEN | Timer Enable<br>When TEN is clear, it resets the LPTMR internal logic, including the CNR and TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this field, CSR[5:1] must not be altered.<br>0b - LPTMR is disabled and internal logic is reset.<br>1b - LPTMR is enabled. |

## 53.4.2.3 Low Power Timer Prescale Register (PSR)

### 53.4.2.3.1 Offset

| Register | Offset |
|---|---|
| PSR | 4h |

## 53.4.2.3.2 Diagram




## 53.4.2.3.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-3<br>PRESCALE | Prescale Value<br>Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. The width of the glitch filter can vary by 1 cycle due to synchronization of the pulse counter input. PRESCALE must be altered only when the LPTMR is disabled.<br>0000b - Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration.<br>0001b - Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges.<br>0010b - Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges.<br>0011b - Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges.<br>0100b - Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges.<br>0101b - Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges.<br>0110b - Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges.<br>0111b - Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges.<br>1000b - Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges.<br>1001b - Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges.<br>1010b - Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges.<br>1011b - Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges.<br>1100b - Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges.<br>1101b - Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1110b - Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges.<br>1111b - Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges. |
| 2<br>PBYP | Prescaler Bypass<br>When PBYP is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNR. When PBYP is clear, the CNR is clocked by the output of the prescaler/glitch filter. PBYP must be altered only when the LPTMR is disabled.<br>0b - Prescaler/glitch filter is enabled.<br>1b - Prescaler/glitch filter is bypassed. |
| 1-0<br>PCS | Prescaler Clock Select<br>Selects the clock to be used by the LPTMR prescaler/glitch filter. In time counter mode, this field selects the input clock to the prescaler. In pulse counter mode, this field selects the input clock to the glitch filter. PCS must be altered only when the LPTMR is disabled. The clock connections vary by device.<br>**NOTE:** See the chip configuration details for information on the connections to these inputs.<br>00b - Prescaler/glitch filter clock 0 selected.<br>01b - Prescaler/glitch filter clock 1 selected.<br>10b - Prescaler/glitch filter clock 2 selected.<br>11b - Prescaler/glitch filter clock 3 selected. |

## 53.4.2.4 Low Power Timer Compare Register (CMR)

### 53.4.2.4.1 Offset

| Register | Offset |
|---|---|
| CMR | 8h |

### 53.4.2.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 53.4.2.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPARE | Compare Value<br>When the LPTMR is enabled and the CNR equals the value in the CMR and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger will remain asserted until the LPTMR is disabled. If the LPTMR is enabled, the CMR must be altered only when TCF is set. |

### 53.4.2.5 Low Power Timer Counter Register (CNR)

#### 53.4.2.5.1 Offset

| Register | Offset |
|---|---|
| CNR | Ch |

#### 53.4.2.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 53.4.2.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COUNTER | Counter Value<br>The CNR returns the current value of the LPTMR counter at the time this register was last written. |

## 53.4.3 LPTMR2 register descriptions

### 53.4.3.1 LPTMR2 Memory map

LPTMR2 base address: 4102_B000h

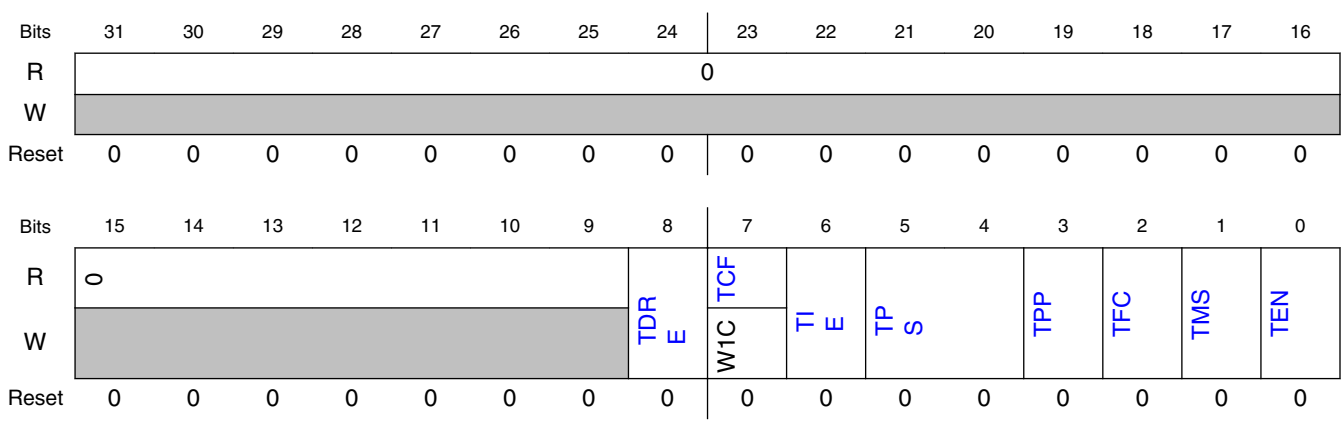
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Low Power Timer Control Status Register (CSR) | 32 | RW | 0000_0000h |
| 4h | Low Power Timer Prescale Register (PSR) | 32 | RW | 0000_0000h |
| 8h | Low Power Timer Compare Register (CMR) | 32 | RW | 0000_0000h |
| Ch | Low Power Timer Counter Register (CNR) | 32 | RW | 0000_0000h |

### 53.4.3.2 Low Power Timer Control Status Register (CSR)

#### 53.4.3.2.1 Offset

| Register | Offset |
|---|---|
| CSR | 0h |

#### 53.4.3.2.2 Diagram



### 53.4.3.2.3 Fields

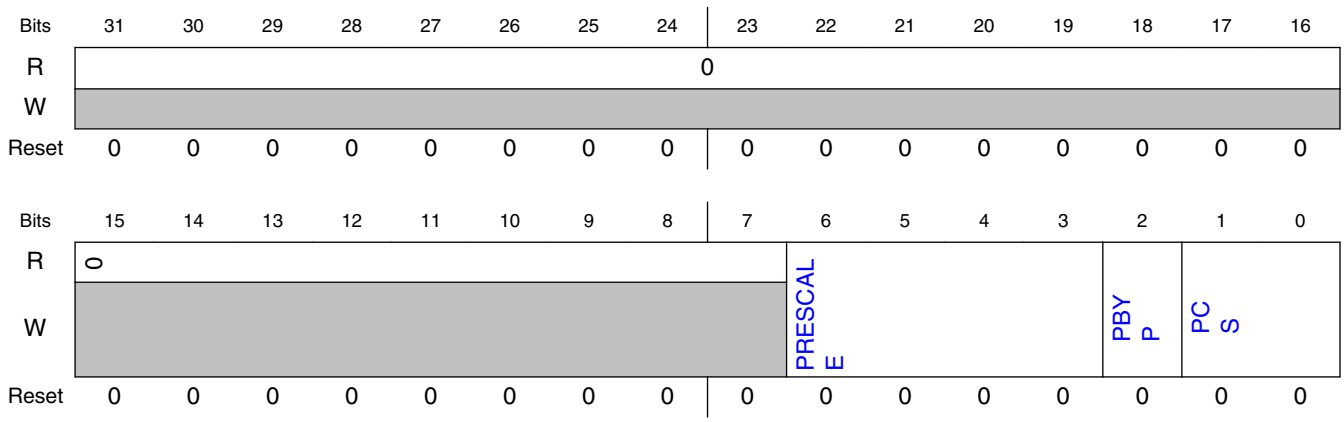
| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>TDRE | Timer DMA Request Enable<br>When TDRE is set, the LPTMR DMA Request is generated whenever TCF is also set and the TCF is cleared when the DMA Controller is done.<br>0b - Timer DMA Request disabled.<br>1b - Timer DMA Request enabled. |
| 7<br>TCF | Timer Compare Flag<br>TCF is set when the LPTMR is enabled and the CNR equals the CMR and increments. TCF is cleared when the LPTMR is disabled or a logic 1 is written to it.<br>0b - The value of CNR is not equal to CMR and increments.<br>1b - The value of CNR is equal to CMR and increments. |
| 6<br>TIE | Timer Interrupt Enable<br>When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set.<br>0b - Timer interrupt disabled.<br>1b - Timer interrupt enabled. |
| 5-4<br>TPS | Timer Pin Select<br>Configures the input source to be used in Pulse Counter mode. TPS must be altered only when the LPTMR is disabled. The input connections vary by device. See the chip configuration information about connections to these inputs.<br>00b - Pulse counter input 0 is selected.<br>01b - Pulse counter input 1 is selected.<br>10b - Pulse counter input 2 is selected.<br>11b - Pulse counter input 3 is selected. |
| 3<br>TPP | Timer Pin Polarity<br>Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled.<br>0b - Pulse Counter input source is active-high, and the CNR will increment on the rising-edge.<br>1b - Pulse Counter input source is active-low, and the CNR will increment on the falling-edge. |
| 2<br>TFC | Timer Free-Running Counter<br>When clear, TFC configures the CNR to reset whenever TCF is set. When set, TFC configures the CNR to reset on overflow. TFC must be altered only when the LPTMR is disabled.<br>0b - CNR is reset whenever TCF is set.<br>1b - CNR is reset on overflow. |
| 1<br>TMS | Timer Mode Select<br>Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled.<br>0b - Time Counter mode.<br>1b - Pulse Counter mode. |
| 0<br>TEN | Timer Enable<br>When TEN is clear, it resets the LPTMR internal logic, including the CNR and TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this field, CSR[5:1] must not be altered.<br>0b - LPTMR is disabled and internal logic is reset.<br>1b - LPTMR is enabled. |

# 53.4.3.3 Low Power Timer Prescale Register (PSR)

## 53.4.3.3.1 Offset

| Register | Offset |
|---|---|
| PSR | 4h |

## 53.4.3.3.2 Diagram




## 53.4.3.3.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-3<br>PRESCALE | Prescale Value<br>Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. The width of the glitch filter can vary by 1 cycle due to synchronization of the pulse counter input. PRESCALE must be altered only when the LPTMR is disabled.<br>0000b - Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration.<br>0001b - Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges.<br>0010b - Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges.<br>0011b - Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges.<br>0100b - Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges.<br>0101b - Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges.<br>0110b - Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0111b - Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges.<br>1000b - Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges.<br>1001b - Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges.<br>1010b - Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges.<br>1011b - Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges.<br>1100b - Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges.<br>1101b - Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges.<br>1110b - Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges.<br>1111b - Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges. |
| 2<br>PBYP | Prescaler Bypass<br>When PBYP is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNR. When PBYP is clear, the CNR is clocked by the output of the prescaler/glitch filter. PBYP must be altered only when the LPTMR is disabled.<br>0b - Prescaler/glitch filter is enabled.<br>1b - Prescaler/glitch filter is bypassed. |
| 1-0<br>PCS | Prescaler Clock Select<br>Selects the clock to be used by the LPTMR prescaler/glitch filter. In time counter mode, this field selects the input clock to the prescaler. In pulse counter mode, this field selects the input clock to the glitch filter. PCS must be altered only when the LPTMR is disabled. The clock connections vary by device.<br>**NOTE:** See the chip configuration details for information on the connections to these inputs.<br>00b - Prescaler/glitch filter clock 0 selected.<br>01b - Prescaler/glitch filter clock 1 selected.<br>10b - Prescaler/glitch filter clock 2 selected.<br>11b - Prescaler/glitch filter clock 3 selected. |

## 53.4.3.4 Low Power Timer Compare Register (CMR)

### 53.4.3.4.1 Offset

| Register | Offset |
|---|---|
| CMR | 8h |

#### 53.4.3.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 53.4.3.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COMPARE | Compare Value<br>When the LPTMR is enabled and the CNR equals the value in the CMR and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger will remain asserted until the LPTMR is disabled. If the LPTMR is enabled, the CMR must be altered only when TCF is set. |

### 53.4.3.5 Low Power Timer Counter Register (CNR)

#### 53.4.3.5.1 Offset

| Register | Offset |
|---|---|
| CNR | Ch |

#### 53.4.3.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNTER | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 53.4.3.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>COUNTER | Counter Value<br>The CNR returns the current value of the LPTMR counter at the time this register was last written. |

## 53.5 Functional description

### 53.5.1 LPTMR power and reset

The LPTMR remains powered in all power modes, including low-leakage modes. If the LPTMR is not required to remain operating during a low-power mode, then it must be disabled before entering the mode.

The LPTMR is reset only on global Power On Reset (POR) or Low Voltage Detect (LVD). When configuring the LPTMR registers, the CSR must be initially written with the timer disabled, before configuring the PSR and CMR. Then, CSR[TIE] must be set as the last step in the initialization. This ensures the LPTMR is configured correctly and the LPTMR counter is reset to zero following a warm reset.

### 53.5.2 LPTMR clocking

The LPTMR prescaler/glitch filter can be clocked by one of the four clocks. The clock source must be enabled before the LPTMR is enabled.

**NOTE**

The clock source selected by PSR[PCS] may need to be configured to remain enabled in low-power modes, otherwise the LPTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNR and no other clock source is required. To minimize power in this case, configure the prescaler clock source for a clock that is not toggling.

**NOTE**

The clock source or pulse input source selected for the LPTMR should not exceed the frequency $f_{LPTMR}$ defined in the device datasheet.

## 53.5.3 LPTMR prescaler/glitch filter

The LPTMR prescaler and glitch filter share the same logic which operates as a prescaler in Time Counter mode and as a glitch filter in Pulse Counter mode.

**NOTE**

The prescaler/glitch filter configuration must not be altered when the LPTMR is enabled.

### 53.5.3.1 Prescaler enabled

In Time Counter mode, when the prescaler is enabled, the output of the prescaler directly clocks the CNR. When the LPTMR is enabled, the CNR will increment every $2^2$ to $2^{16}$ prescaler clock cycles. After the LPTMR is enabled, the first increment of the CNR will take an additional one or two prescaler clock cycles due to synchronization logic.

### 53.5.3.2 Prescaler bypassed

In Time Counter mode, when the prescaler is bypassed, the selected prescaler clock increments the CNR on every clock cycle. When the LPTMR is enabled, the first increment will take an additional one or two prescaler clock cycles due to synchronization logic.

### 53.5.3.3 Glitch filter

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks the CNR. When the LPTMR is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. The following table shows the change in glitch filter output with the selected input source.

| If | Then |
|---|---|
| The selected input source remains deasserted for at least $2^1$ to $2^{15}$ consecutive prescaler clock rising edges | The glitch filter output will also deassert. |
| The selected input source remains asserted for at least $2^1$ to $2^{15}$ consecutive prescaler clock rising-edges | The glitch filter output will also assert. |

**NOTE**

The input is only sampled on the rising clock edge.

The CNR will increment each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which the CNR can increment is once every $2^2$ to $2^{16}$ prescaler clock edges. When first enabled, the glitch filter will wait an additional one or two prescaler clock edges due to synchronization logic.

#### 53.5.3.4 Glitch filter bypassed

In Pulse Counter mode, when the glitch filter is bypassed, the selected input source increments the CNR every time it asserts. Before the LPTMR is first enabled, the selected input source is forced to be asserted. This prevents the CNR from incrementing if the selected input source is already asserted when the LPTMR is first enabled.

### 53.5.4 LPTMR compare

When the CNR equals the value of the CMR and increments, the following events occur:

- CSR[TCF] is set.
- LPTMR interrupt is generated if CSR[TIE] is also set.
- LPTMR hardware trigger is generated.
- CNR is reset if CSR[TFC] is clear.

When the LPTMR is enabled, the CMR can be altered only when CSR[TCF] is set. When updating the CMR, the CMR must be written and CSR[TCF] must be cleared before the LPTMR counter has incremented past the new LPTMR compare value.

### 53.5.5 LPTMR counter

The CNR increments by one on every:

- Prescaler clock in Time Counter mode with prescaler bypassed
- Prescaler output in Time Counter mode with prescaler enabled
- Input source assertion in Pulse Counter mode with glitch filter bypassed
- Glitch filter output in Pulse Counter mode with glitch filter enabled

The CNR is reset when the LPTMR is disabled or if the counter register overflows. If CSR[TFC] is cleared, then the CNR is also reset whenever CSR[TCF] is set.

When the core is halted in Debug mode:

- If configured for Pulse Counter mode, the CNR continues incrementing.
- If configured for Time Counter mode, the CNR stops incrementing.

The CNR cannot be initialized, but can be read at any time. On each read of the CNR, software must first write to the CNR with any value. This will synchronize and register the current value of the CNR into a temporary register. The contents of the temporary register are returned on each read of the CNR.

When reading the CNR, the bus clock must be at least two times faster than the rate at which the LPTMR counter is incrementing, otherwise incorrect data may be returned.

### 53.5.6 LPTMR hardware trigger

The LPTMR hardware trigger asserts at the same time the CSR[TCF] is set and can be used to trigger hardware events in other peripherals without software intervention. The hardware trigger is always enabled.

| When | Then |
|---|---|
| The CMR is set to 0 with CSR[TFC] clear | The LPTMR hardware trigger will assert on the first compare and does not deassert. |
| The CMR is set to a nonzero value, or, if CSR[TFC] is set | The LPTMR hardware trigger will assert on each compare and deassert on the following increment of the CNR. |

### 53.5.7 LPTMR interrupt

The LPTMR interrupt is generated whenever CSR[TIE] and CSR[TCF] are set. CSR[TCF] is cleared by disabling the LPTMR or by writing a logic 1 to it.

CSR[TIE] can be altered and CSR[TCF] can be cleared while the LPTMR is enabled.

The LPTMR interrupt is generated asynchronously to the system clock and can be used to generate a wakeup from any low-power mode, including the low-leakage modes, provided the LPTMR is enabled as a wakeup source.

# Chapter 54
# Real Time Clock (RTC)

## 54.1 Chip-specific RTC information

### 54.1.1 RTC_CLKOUT signal

When the RTC is enabled and the port control module selects the RTC_CLKOUT function, the RTC_CLKOUT signal outputs a 1 Hz or 32 kHz output derived from RTC oscillator as shown below.

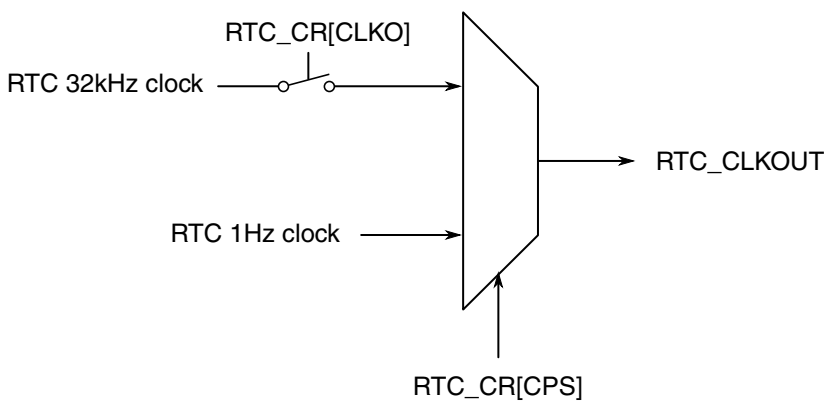



**Figure 54-1. RTC_CLKOUT generation**

**NOTE**

It is possible on variants of this device with RTC_WAKEUP pin to output the RTC clock signal to the RTC_WAKEUP pin. Please refer to the RTC Control Register for details.

### 54.1.2 Number of tamper inputs

This device contains up to 4 tamper input pins. See the device pinouts to see which tamper pins are available in a given package.

## 54.2 Introduction

### 54.2.1 Features

The RTC module features include:

- Independent power supply, POR, and 32.768 kHz crystal oscillator
- 32-bit seconds counter with roll-over protection and 32-bit alarm
- 16-bit prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm
- Option to increment prescaler using a 1 kHz LPO (prescaler increments by 32 every clock edge)
- Register write protection
  - Lock register requires VBAT POR or software reset to enable write access
  - Access control registers require system reset to enable read and/or write access
- Configurable 1, 2, 4, 8, 16, 32, 64 or 128 Hz square wave output with optional interrupt
- 64-bit monotonic counter with roll-over protection
- Up to 4 tamper input pins with optional interrupt and seconds timestamp when interrupt asserts

### 54.2.2 Modes of operation

The RTC remains functional in all low power modes and can generate an interrupt to exit any low power mode. It operates in one of two modes of operation: chip power-up and chip power-down.

During chip power-down, RTC is powered from the backup power supply (VBAT) and is electrically isolated from the rest of the chip but continues to increment the time counter (if enabled) and retain the state of the RTC registers. The RTC registers are not accessible.

During chip power-up, RTC remains powered from the backup power supply (VBAT). All RTC registers are accessible by software and all functions are operational. If enabled, the 32.768 kHz clock can be supplied to the rest of the chip.

## 54.2.3 RTC signal descriptions

**Table 54-1. RTC signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| EXTAL32 | 32.768 kHz oscillator input | I |
| XTAL32 | 32.768 kHz oscillator output | O |
| RTC_CLKOUT | Prescaler square-wave output or RTC 32.768 kHz clock | O |
| RTC_WAKEUP_b | Active low wakeup for external device | O |
| RTC_TAMPER[3:0] | Tamper pin input | I |

### 54.2.3.1 RTC clock output

The RTC_CLKOUT signal can output either a square wave prescaler output (configurable to 1, 2, 4, 8, 16, 32, 64 or 128 Hz) or the RTC 32.768 kHz clock.

### 54.2.3.2 RTC wakeup pin

The RTC wakeup pin is an open drain, active low, output that allows the RTC to wakeup the chip via an external component. The wakeup pin asserts when the wakeup pin enable is set and either the RTC interrupt is asserted or the wakeup pin is turned on via a register bit. The wakeup pin does not assert from the RTC seconds interrupt.

**NOTE**

The wakeup pin is optional and may not be implemented on all devices.

### 54.2.3.3 RTC Tamper pins

The RTC tamper pins are input pins that can be used to generate an interrupt or invalidate the time counter. The first tamper pin to assert an interrupt will also capture a timestamp of the seconds register.

## 54.3 Register definition

All registers must be accessed using 32-bit writes and all register accesses incur three wait states.

Write accesses to any register and read accesses to tamper and monotonic registers by non-supervisor mode software, when the supervisor access bit in the control register is clear, will terminate with a bus error.

Read accesses to other registers by non-supervisor mode software complete as normal.

Writing to a register protected by the write access register or lock register does not generate a bus error, but the write will not complete.

Reading a register protected by the read access register does not generate a bus error, but the register will read zero.

### 54.3.1 RTC register descriptions

#### 54.3.1.1 RTC Memory map

RTC base address: 4003_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | RTC Time Seconds Register (TSR) | 32 | RW | 0000_0000h |
| 4h | RTC Time Prescaler Register (TPR) | 32 | RW | 0000_0000h |
| 8h | RTC Time Alarm Register (TAR) | 32 | RW | 0000_0000h |
| Ch | RTC Time Compensation Register (TCR) | 32 | RW | 0000_0000h |
| 10h | RTC Control Register (CR) | 32 | RW | 0000_0000h |
| 14h | RTC Status Register (SR) | 32 | RW | 0000_0001h |
| 18h | RTC Lock Register (LR) | 32 | RW | 000F_FFFFh |
| 1Ch | RTC Interrupt Enable Register (IER) | 32 | RW | 0000_0007h |
| 20h | RTC Tamper Time Seconds Register (TTSR) | 32 | RO | See description. |
| 24h | RTC Monotonic Enable Register (MER) | 32 | RW | 0000_0000h |
| 28h | RTC Monotonic Counter Low Register (MCLR) | 32 | RW | 0000_0000h |
| 2Ch | RTC Monotonic Counter High Register (MCHR) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 34h | RTC Tamper Detect Register (TDR) | 32 | W1C | 0000_0001h |
| 3Ch | RTC Tamper Interrupt Register (TIR) | 32 | RW | 0000_0000h |
| 40h - 4Ch | RTC Pin Configuration Register (PCR0 - PCR3) | 32 | RW | 0000_0000h |
| 800h | RTC Write Access Register (WAR) | 32 | RW | 000F_FFFFh |
| 804h | RTC Read Access Register (RAR) | 32 | RW | 000F_FFFFh |

## 54.3.1.2 RTC Time Seconds Register (TSR)

### 54.3.1.2.1 Offset

| Register | Offset |
|---|---|
| TSR | 0h |

### 54.3.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 54.3.1.2.3 Fields

| Field | Function |
|---|---|
| 31-0<br>TSR | Time Seconds Register<br>When the time counter is enabled, the TSR is read only and increments once a second provided SR[TOF] or SR[TIF] are not set. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TSR can be read or written. Writing to the TSR when the time counter is disabled will clear the SR[TOF] and/or the SR[TIF]. Writing to TSR with zero is supported, but not recommended because TSR will read as zero when SR[TIF] or SR[TOF] are set (indicating the time is invalid). |

## 54.3.1.3 RTC Time Prescaler Register (TPR)

### 54.3.1.3.1 Offset

| Register | Offset |
|---|---|
| TPR | 4h |

### 54.3.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TPR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 54.3.1.3.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TPR | Time Prescaler Register<br>When the time counter is enabled, the TPR is read only and increments every 32.768 kHz clock cycle. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TPR can be read or written. The TSR[TSR] increments when bit 14 of the TPR transitions from a logic one to a logic zero. |

## 54.3.1.4 RTC Time Alarm Register (TAR)

### 54.3.1.4.1 Offset

| Register | Offset |
|---|---|
| TAR | 8h |

### 54.3.1.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TAR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TAR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 54.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-0<br>TAR | Time Alarm Register<br>When the time counter is enabled, the SR[TAF] is set whenever the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. Writing to the TAR clears the SR[TAF]. |

## 54.3.1.5 RTC Time Compensation Register (TCR)

### 54.3.1.5.1 Offset

| Register | Offset |
|---|---|
| TCR | Ch |

### 54.3.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIC | | | | | | | | TCV | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIR | | | | | | | | TCR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 54.3.1.5.3 Fields

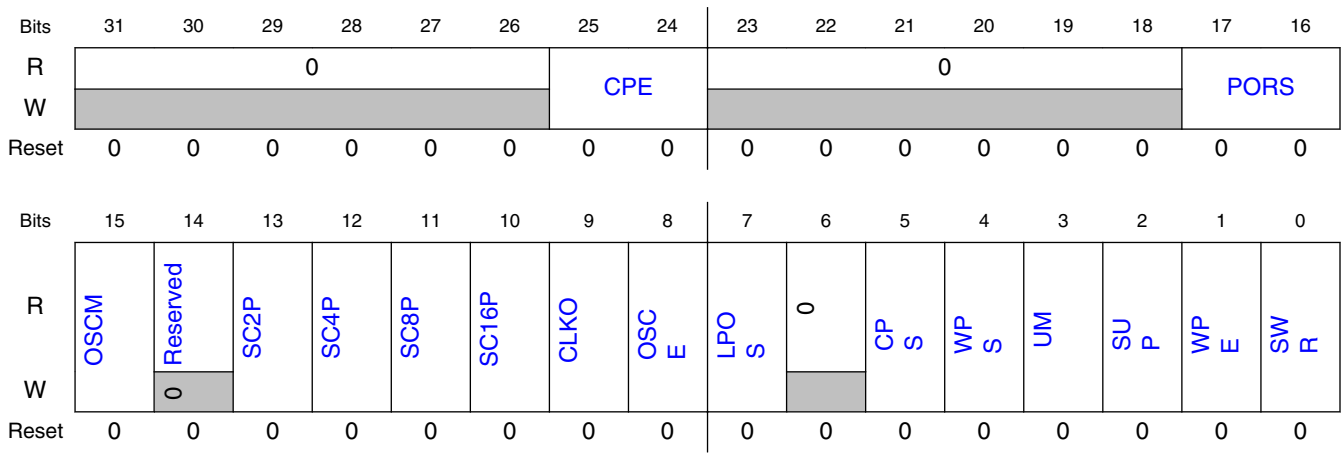
| Field | Function |
|---|---|
| 31-24<br>CIC | Compensation Interval Counter<br>Current value of the compensation interval counter. If the compensation interval counter equals zero then it is loaded with the contents of the CIR. If the CIC does not equal zero then it is decremented once a second. Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read. |
| 23-16<br>TCV | Time Compensation Value<br>Current value used by the compensation logic for the present second interval. Updated once a second if the CIC equals 0 with the contents of the TCR field. If the CIC does not equal zero then it is loaded with zero (compensation is not enabled for that second increment). Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read. |
| 15-8<br>CIR | Compensation Interval Register<br>Configures the compensation interval in seconds from 1 to 256 to control how frequently the TCR should adjust the number of 32.768 kHz cycles in each second. The value written should be one less than the number of seconds. For example, write zero to configure for a compensation interval of one second. This register is double buffered and writes do not take affect until the end of the current compensation interval. |
| 7-0<br>TCR | Time Compensation Register<br>Configures the number of 32.768 kHz clock cycles in each second, equal to 32,768 - TCR (where TCR is a twos complement sign extended value). This register is double buffered and writes do not take affect until the end of the current compensation interval. Some example values are show below.<br>00000000b - Time Prescaler Register overflows every 32768 clock cycles.<br>00000001b - Time Prescaler Register overflows every 32767 clock cycles.<br>01111110b - Time Prescaler Register overflows every 32642 clock cycles.<br>01111111b - Time Prescaler Register overflows every 32641 clock cycles.<br>10000000b - Time Prescaler Register overflows every 32896 clock cycles.<br>10000001b - Time Prescaler Register overflows every 32895 clock cycles.<br>11111111b - Time Prescaler Register overflows every 32769 clock cycles. |

## 54.3.1.6 RTC Control Register (CR)

### 54.3.1.6.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 54.3.1.6.2 Diagram




### 54.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>CPE | Clock Pin Enable<br>00b - The RTC_CLKOUT function is disabled.<br>01b - Enable RTC_CLKOUT pin on pin 1.<br>10b - Enable RTC_CLKOUT pin on pin 2.<br>11b - Enable RTC_CLKOUT pin on pin 3. |
| 23-18<br>— | Reserved |
| 17-16<br>PORS | POR Select<br>Configures pulsing of the POR brownout, this reduces average power consumption. POR brownout is always enabled on a loss of clock event, if the time invalid flag is set or if the 32.768kHz crystal oscillator is disabled.<br>00b - POR brownout enabled for 120us every 128ms.<br>01b - POR brownout enabled for 120us every 64ms.<br>10b - POR brownout enabled for 120us every 32ms.<br>11b - POR brownout always enabled. |
| 15<br>OSCM | Oscillator Mode Select<br>0b - Configures the 32.768kHz crystal oscillator for robust operation supporting a wide range of crystals. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Configures the 32.768kHz crystal oscillator for low power operation supporting a more limited range of crystals. |
| 14<br>— | Reserved<br>It must always be written to 0. |
| 13<br>SC2P | Oscillator 2pF Load Configure<br>0b - Disable the load.<br>1b - Enable the additional load. |
| 12<br>SC4P | Oscillator 4pF Load Configure<br>0b - Disable the load.<br>1b - Enable the additional load. |
| 11<br>SC8P | Oscillator 8pF Load Configure<br>0b - Disable the load.<br>1b - Enable the additional load. |
| 10<br>SC16P | Oscillator 16pF Load Configure<br>0b - Disable the load.<br>1b - Enable the additional load. |
| 9<br>CLKO | Clock Output<br>0b - The 32 kHz clock is output to other peripherals.<br>1b - The 32 kHz clock is not output to other peripherals. |
| 8<br>OSCE | Oscillator Enable<br>0b - 32.768 kHz oscillator is disabled.<br>1b - 32.768 kHz oscillator is enabled. After setting this bit, wait the oscillator startup time before enabling the time counter to allow the 32.768 kHz clock time to stabilize. |
| 7<br>LPOS | LPO Select<br>When set, the RTC prescaler increments using the LPO 1 kHz clock and not the RTC 32.768 kHz clock. The LPO increments the prescaler from bit TPR[5] (TPR[4:0] are ignored), supporting close to 1 second increment of the seconds register. Although compensation is supported when clocked from the LPO, TCR[4:0] of the compensation register are also ignored and only TCR[7:5] set the compensation value (can overflow after 1020 to 1027 cycles).<br>0b - RTC prescaler increments using 32.768 kHz clock.<br>1b - RTC prescaler increments using 1 kHz LPO, bits [4:0] of the prescaler are ignored. |
| 6<br>— | Reserved |
| 5<br>CPS | Clock Pin Select<br>0b - The prescaler output clock (as configured by TSIC) is output on RTC_CLKOUT.<br>1b - The RTC 32.768 kHz clock is output on RTC_CLKOUT, provided it is output to other peripherals. |
| 4<br>WPS | Wakeup Pin Select<br>The RTC_WAKEUP pin is optional and not available on all devices.<br>0b - RTC_WAKEUP pin asserts (active low, open drain) if the RTC interrupt asserts or the wakeup pin is turned on.<br>1b - RTC_WAKEUP pin outputs the RTC 32kHz clock, provided the wakeup pin is turned on and the 32kHz clock is output to other peripherals. |
| 3<br>UM | Update Mode<br>Allows SR[TCE] to be written even when the Status Register is locked. When set, the SR[TCE] can be written if the SR[TIF] or SR[TOF] are set or if the SR[TCE] is clear. Whenever both SR[TCE] and CR[UM] are set, then SR[TCE] should only be written once either SR[TIF] or SR[TOF] are set. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Allows the monotonic enable register to be written when it is locked. When set, the monotonic enable register can always be written if the SR[TIF] or SR[MOF] are set or if the montonic counter enable is clear.<br>Allows the tamper detect register to be written when it is locked. When set, the tamper detect register can always be written if the SR[TIF] is clear.<br>0b - Registers cannot be written when locked.<br>1b - Registers can be written when locked under limited conditions. |
| 2<br>SUP | Supervisor Access<br>Configures non-supervisor mode write access to all RTC registers and non-supervisor mode read access to RTC tamper/monotonic registers<br>0b - Non-supervisor mode write accesses are not supported and generate a bus error.<br>1b - Non-supervisor mode write accesses are supported. |
| 1<br>WPE | Wakeup Pin Enable<br>The RTC_WAKEUP pin is optional and not available on all devices.<br>0b - RTC_WAKEUP pin is disabled.<br>1b - RTC_WAKEUP pin is enabled and asserts if the RTC interrupt asserts or if the wakeup pin is forced on. |
| 0<br>SWR | Software Reset<br>0b - No effect.<br>1b - Resets all RTC registers except for the SWR bit and the RTC_WAR and RTC_RAR registers . The SWR bit is cleared by VBAT POR and by software explicitly clearing it. |

## 54.3.1.7 RTC Status Register (SR)

### 54.3.1.7.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 54.3.1.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TIDF | 0 | | TCE | MOF | TAF | TOF | TIF |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 54.3.1.7.3 Fields

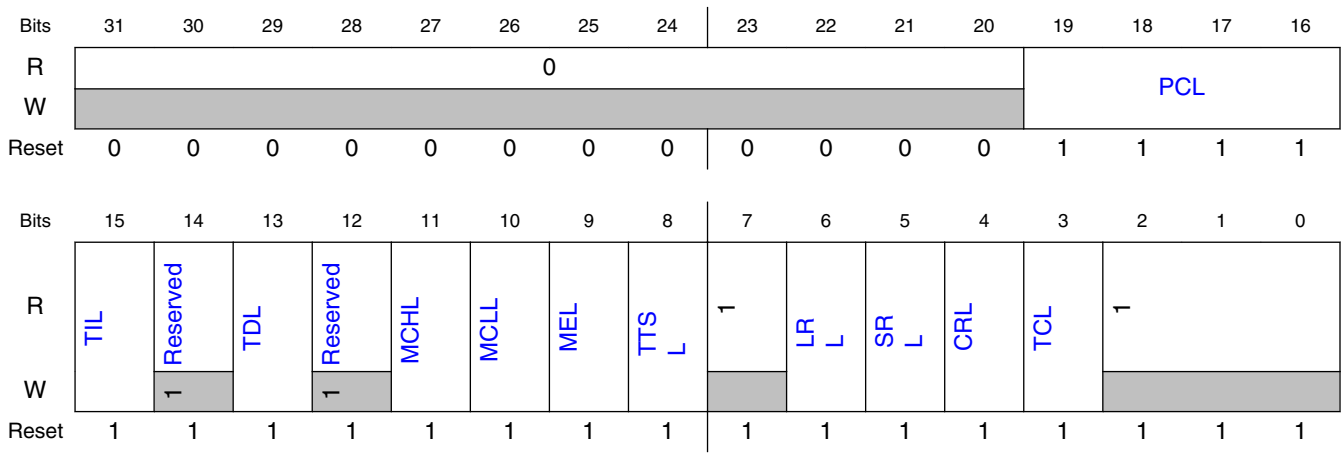
| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>TIDF | Tamper Interrupt Detect Flag<br>The Tamper Interrupt Detect Flag is set whenever the Tamper interrupt is asserted, as configured by the Tamper Status Register and Tamper Interrupt Enable Register.<br>0b - Tamper interrupt has not asserted.<br>1b - Tamper interrupt has asserted. |
| 6-5<br>— | Reserved |
| 4<br>TCE | Time Counter Enable<br>When time counter is disabled the TSR register and TPR register are writeable, but do not increment. When time counter is enabled the TSR register and TPR register are not writeable, but increment.<br>0b - Time counter is disabled.<br>1b - Time counter is enabled. |
| 3<br>MOF | Monotonic Overflow Flag<br>Monotonic overflow flag is set when the monotonic counter is enabled and the monotonic counter high overflows. The monotonic counter does not increment and will read as zero when this bit is set. This bit is cleared by writing the monotonic counter high register when the monotonic counter is disabled.<br>0b - Monotonic counter overflow has not occurred.<br>1b - Monotonic counter overflow has occurred and monotonic counter is read as zero. |
| 2<br>TAF | Time Alarm Flag<br>Time alarm flag is set when the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. This bit is cleared by writing the TAR register.<br>0b - Time alarm has not occurred.<br>1b - Time alarm has occurred. |
| 1<br>TOF | Time Overflow Flag<br>Time overflow flag is set when the time counter is enabled and overflows. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.<br>0b - Time overflow has not occurred.<br>1b - Time overflow has occurred and time counter is read as zero. |
| 0<br>TIF | Time Invalid Flag<br>The time invalid flag is set on VBAT POR or software reset. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.<br>The monotonic counter register is held in reset whenever the time invalid flag is set.<br>0b - Time is valid.<br>1b - Time is invalid and time counter is read as zero. |

# 54.3.1.8 RTC Lock Register (LR)

## 54.3.1.8.1 Offset

| Register | Offset |
|---|---|
| LR | 18h |

## 54.3.1.8.2 Diagram




## 54.3.1.8.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>PCL | Pin Configuration Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Pin Configuration Register is locked and writes are ignored.<br>1b - Pin Configuration Register is not locked and writes complete as normal. |
| 15<br>TIL | Tamper Interrupt Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Tamper Interrupt Register is locked and writes are ignored.<br>1b - Tamper Interrupt Register is not locked and writes complete as normal. |
| 14<br>— | Reserved |
| 13<br>TDL | Tamper Detect Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Tamper Detect Register is locked and writes are ignored.<br>1b - Tamper Detect Register is not locked and writes complete as normal. |

*Table continues on the next page...*

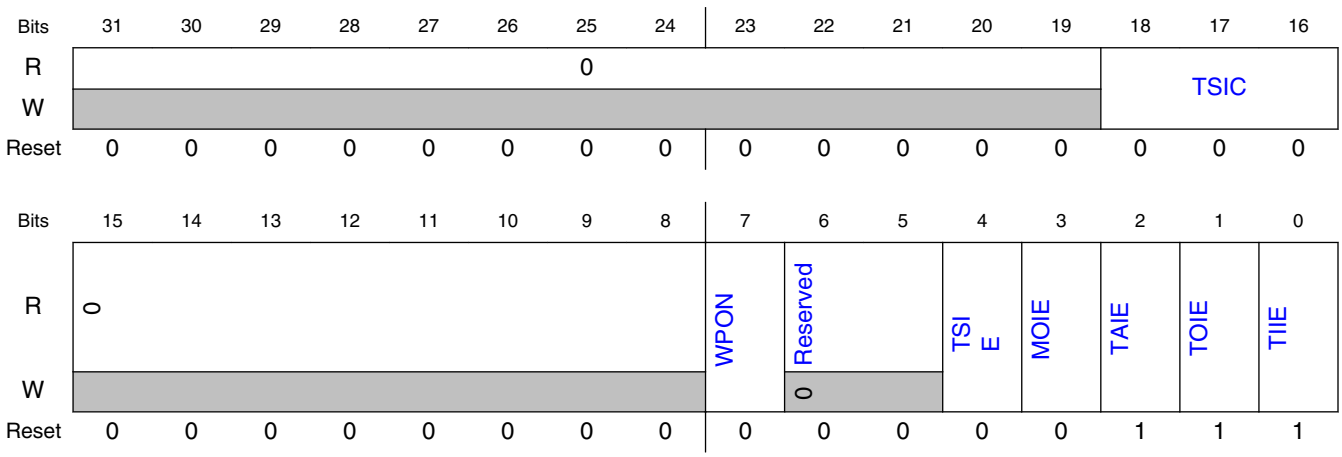
| Field | Function |
|---|---|
| 12<br>— | Reserved |
| 11<br>MCHL | Monotonic Counter High Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Monotonic Counter High Register is locked and writes are ignored.<br>1b - Monotonic Counter High Register is not locked and writes complete as normal. |
| 10<br>MCLL | Monotonic Counter Low Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Monotonic Counter Low Register is locked and writes are ignored.<br>1b - Monotonic Counter Low Register is not locked and writes complete as normal. |
| 9<br>MEL | Monotonic Enable Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Monotonic Enable Register is locked and writes are ignored.<br>1b - Monotonic Enable Register is not locked and writes complete as normal. |
| 8<br>TTSL | Tamper Time Seconds Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Tamper Time Seconds Register is locked and writes are ignored.<br>1b - Tamper Time Seconds Register is not locked and writes complete as normal. |
| 7<br>— | Reserved |
| 6<br>LRL | Lock Register Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Lock Register is locked and writes are ignored.<br>1b - Lock Register is not locked and writes complete as normal. |
| 5<br>SRL | Status Register Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Status Register is locked and writes are ignored.<br>1b - Status Register is not locked and writes complete as normal. |
| 4<br>CRL | Control Register Lock<br>After being cleared, this bit can only be set by VBAT POR.<br>0b - Control Register is locked and writes are ignored.<br>1b - Control Register is not locked and writes complete as normal. |
| 3<br>TCL | Time Compensation Lock<br>After being cleared, this bit can be set only by VBAT POR or software reset.<br>0b - Time Compensation Register is locked and writes are ignored.<br>1b - Time Compensation Register is not locked and writes complete as normal. |
| 2-0<br>— | Reserved |

## 54.3.1.9 RTC Interrupt Enable Register (IER)

### 54.3.1.9.1 Offset

| Register | Offset |
|---|---|
| IER | 1Ch |

### 54.3.1.9.2 Diagram




### 54.3.1.9.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>TSIC | Timer Seconds Interrupt Configuration<br>Configures the frequency of the RTC Seconds interrupt and the RTC_CLKOUT prescaler output. This field should only be altered when TSIE is clear.<br>000b - 1 Hz.<br>001b - 2 Hz.<br>010b - 4 Hz.<br>011b - 8 Hz.<br>100b - 16 Hz.<br>101b - 32 Hz.<br>110b - 64 Hz.<br>111b - 128 Hz. |
| 15-8<br>— | Reserved |
| 7<br>WPON | Wakeup Pin On<br>The RTC_WAKEUP pin is optional and not available on all devices. Whenever the RTC_WAKEUP pin is enabled and this bit is set, it will assert.<br>0b - No effect.<br>1b - If the RTC_WAKEUP pin is enabled, then the pin will assert. |
| 6-5 | Reserved |

*Table continues on the next page...*

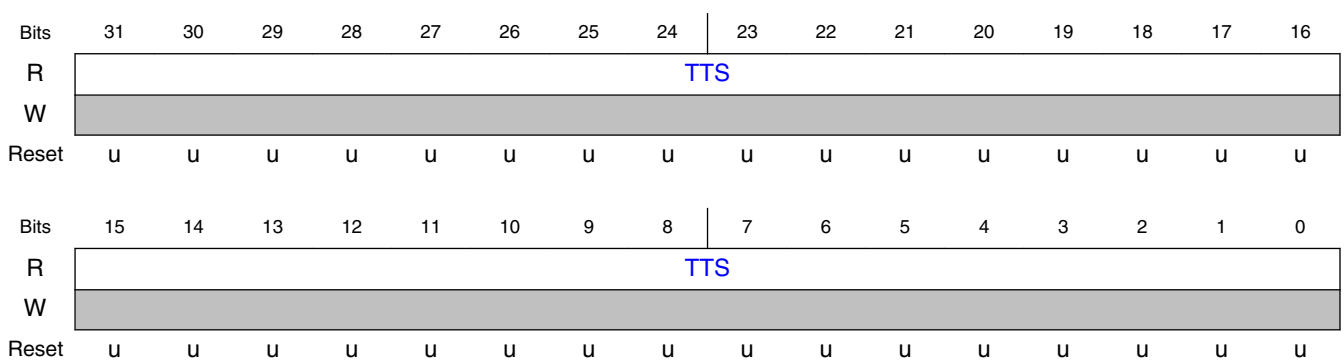
| Field | Function |
|---|---|
| — | |
| 4<br>TSIE | Time Seconds Interrupt Enable<br>The seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector. It is generated at least once a second and requires no software overhead (there is no corresponding status flag to clear). The frequency of the seconds interrupt is configured by TSIC.<br>0b - Seconds interrupt is disabled.<br>1b - Seconds interrupt is enabled. |
| 3<br>MOIE | Monotonic Overflow Interrupt Enable<br>0b - Monotonic overflow flag does not generate an interrupt.<br>1b - Monotonic overflow flag does generate an interrupt. |
| 2<br>TAIE | Time Alarm Interrupt Enable<br>0b - Time alarm flag does not generate an interrupt.<br>1b - Time alarm flag does generate an interrupt. |
| 1<br>TOIE | Time Overflow Interrupt Enable<br>0b - Time overflow flag does not generate an interrupt.<br>1b - Time overflow flag does generate an interrupt. |
| 0<br>TIIE | Time Invalid Interrupt Enable<br>0b - Time invalid flag does not generate an interrupt.<br>1b - Time invalid flag does generate an interrupt. |

## 54.3.1.10 RTC Tamper Time Seconds Register (TTSR)

### 54.3.1.10.1 Offset

| Register | Offset |
|---|---|
| TTSR | 20h |

### 54.3.1.10.2 Diagram



#### 54.3.1.10.3 Fields

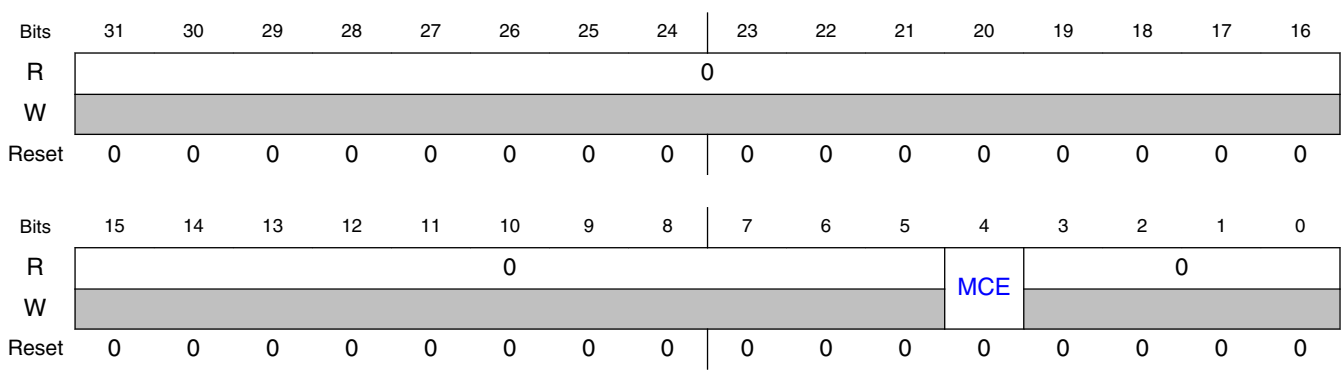
| Field | Function |
|---|---|
| 31-0<br>TTS | Tamper Time Seconds<br>If the time invalid flag is set then reading this register returns the contents of the time seconds register at the point at which the time invalid flag was set. If the time invalid flag is clear, but the tamper interrupt detect flag is set then reading this register returns the point at which the tamper interrupt was set. If both the time invalid flag and tamper detect interrupt are clear, then reading this register returns zero. Writing the tamper time seconds register with any value will set the time invalid flag, allowing software to invalidate the time. |

### 54.3.1.11 RTC Monotonic Enable Register (MER)

#### 54.3.1.11.1 Offset

| Register | Offset |
|---|---|
| MER | 24h |

#### 54.3.1.11.2 Diagram




#### 54.3.1.11.3 Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>MCE | Monotonic Counter Enable<br>0b - Writes to the monotonic counter load the counter with the value written.<br>1b - Writes to the monotonic counter increment the counter. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>— | Reserved |

## 54.3.1.12 RTC Monotonic Counter Low Register (MCLR)

### 54.3.1.12.1 Offset

| Register | Offset |
|---|---|
| MCLR | 28h |

### 54.3.1.12.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCL | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 54.3.1.12.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MCL | Monotonic Counter Low<br>When the time invalid flag is set, the monotonic counter is held in reset. When the monotonic counter enable is clear, a write to this register will load the counter with the value written. When the monotonic counter enable is set, a write to this register will cause it to increment. A write to monotonic counter low that causes it to overflow will also increment monotonic counter high. |

## 54.3.1.13 RTC Monotonic Counter High Register (MCHR)

#### 54.3.1.13.1 Offset

| Register | Offset |
|---|---|
| MCHR | 2Ch |

#### 54.3.1.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 54.3.1.13.3 Fields

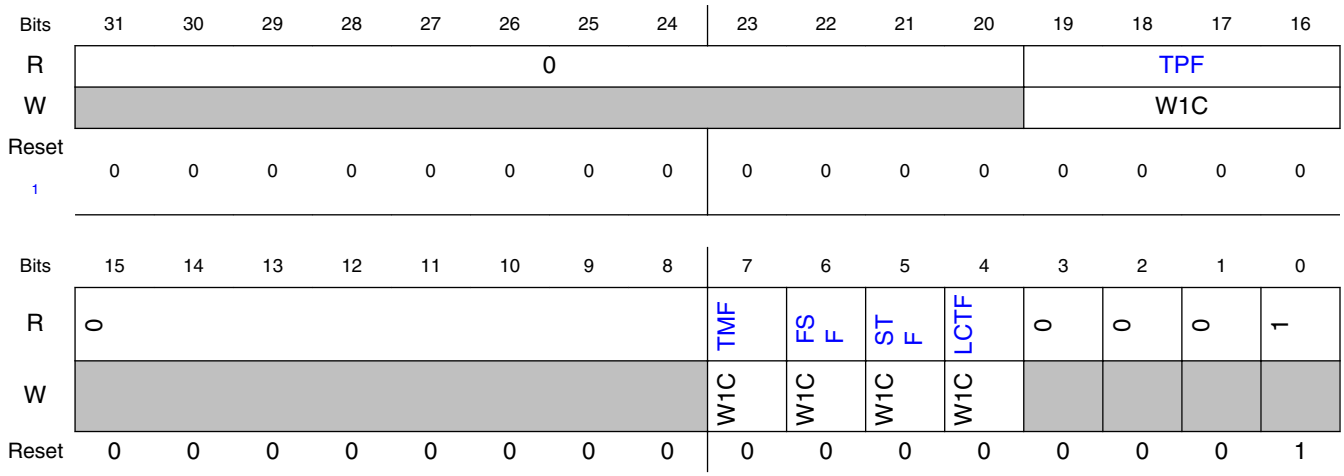
| Field | Function |
|---|---|
| 31-0<br>MCH | Monotonic Counter High<br>When the time invalid flag is set, the monotonic counter is held in reset. When the monotonic counter enable is clear, a write to this register will load the counter with the value written. When the monotonic counter enable is set, a write to this register will cause it to increment. A write to monotonic counter low that causes it to overflow will also increment monotonic counter high. |

### 54.3.1.14 RTC Tamper Detect Register (TDR)

#### 54.3.1.14.1 Offset

| Register | Offset |
|---|---|
| TDR | 34h |

## 54.3.1.14.2 Diagram




1. A software reset will clear all tamper flags, but flags can set again once software reset is negated.

## 54.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>TPF | Tamper Pin Flag<br>These flags are set whenever the corresponding tamper pin asserts. To clear, write logic one to the corresponding flag after that tamper pin negates.<br>0b - Tamper not detected.<br>1b - Tamper pin tamper detected. |
| 15-8<br>— | Reserved |
| 7<br>TMF | Test Mode Flag<br>This flag is set whenever any test mode is entered. To clear, write logic one to this flag after exiting from all test modes.<br>0b - Tamper not detected.<br>1b - Test mode tamper detected. |
| 6<br>FSF | Flash Security Flag<br>This flag is set whenever flash security is disabled. To clear, write logic one to this flag after flash security is enabled.<br>0b - Tamper not detected.<br>1b - Flash security tamper detected. |
| 5<br>STF | Security Tamper Flag<br>This flag is set whenever security module asserts its tamper detect. To clear, write logic one to this flag after security module has negated its tamper detect.<br>0b - Tamper not detected.<br>1b - Security module tamper detected. |
| 4<br>LCTF | Loss of Clock Tamper Flag |

*Table continues on the next page...*

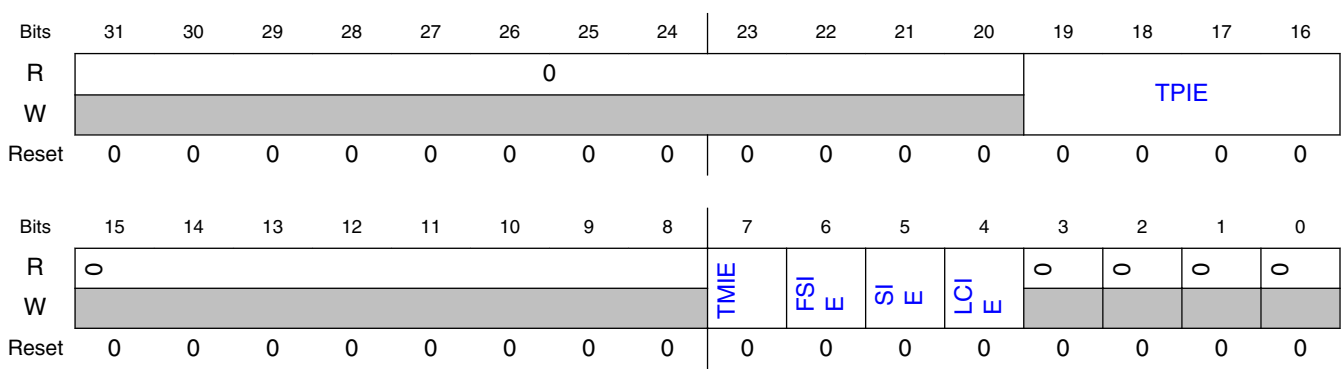
| Field | Function |
|---|---|
| | This flag is set whenever loss of clock is detected. To clear, write logic one to this flag after loss of clock negates.<br>0b - Tamper not detected.<br>1b - Loss of Clock tamper detected. |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 54.3.1.15 RTC Tamper Interrupt Register (TIR)

### 54.3.1.15.1 Offset

| Register | Offset |
|---|---|
| TIR | 3Ch |

### 54.3.1.15.2 Diagram




### 54.3.1.15.3 Fields

| Field | Function |
|---|---|
| 31-20 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 19-16<br>TPIE | Tamper Pin Interrupt Enable<br>0b - Interupt disabled.<br>1b - An interrupt is generated when the corresponding tamper pin flag is set. |
| 15-8<br>— | Reserved |
| 7<br>TMIE | Test Mode Interrupt Enable<br>0b - Interupt disabled.<br>1b - An interrupt is generated when the test mode flag is set. |
| 6<br>FSIE | Flash Security Interrupt Enable<br>0b - Interupt disabled.<br>1b - An interrupt is generated when the flash security flag is set. |
| 5<br>SIE | Security Module Interrupt Enable<br>0b - Interupt disabled.<br>1b - An interrupt is generated when the security module flag is set. |
| 4<br>LCIE | Loss of Clock Interrupt Enable<br>0b - Interupt disabled.<br>1b - An interrupt is generated when the loss of clock flag is set. |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

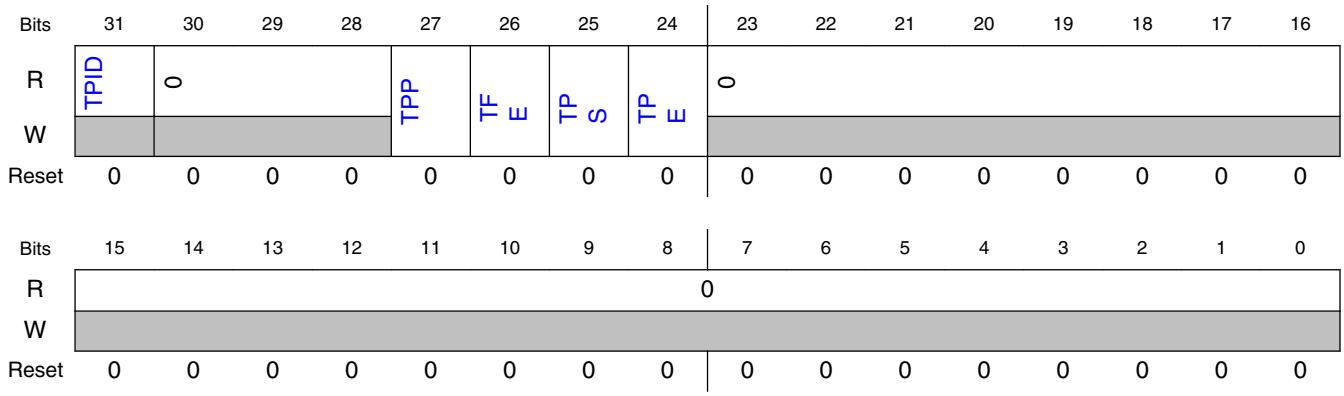
## 54.3.1.16 RTC Pin Configuration Register (PCR0 - PCR3)

### 54.3.1.16.1 Offset

For a = 0 to 3:

| Register | Offset |
|---|---|
| PCRa | 40h + (a × 4h) |

### 54.3.1.16.2 Diagram




### 54.3.1.16.3 Fields

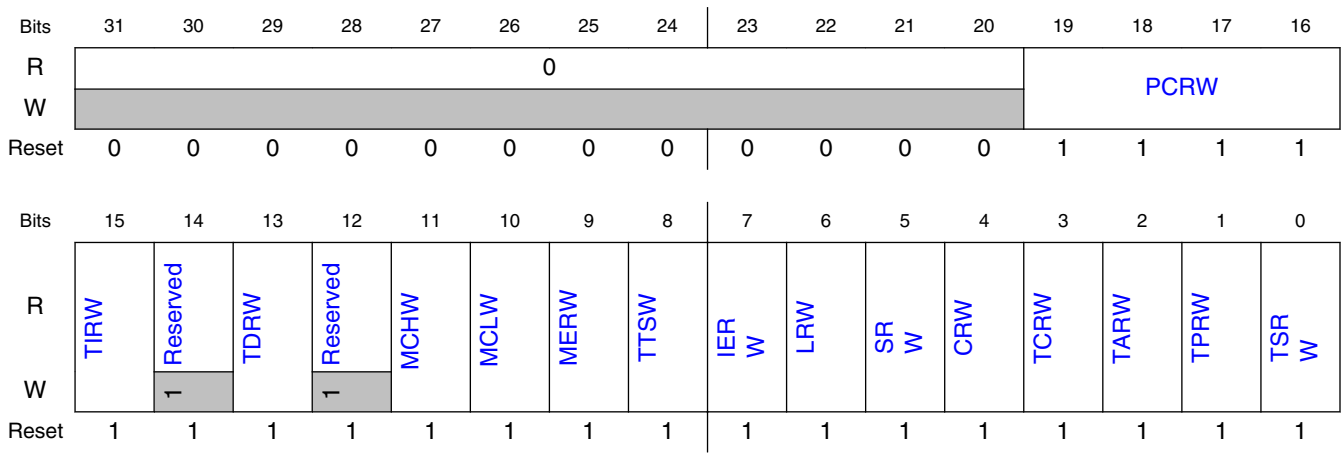
| Field | Function |
|---|---|
| 31<br>TPID | Tamper Pin Input Data<br>0b - Tamper pin input data is logic zero.<br>1b - Tamper pin input data is logic one. |
| 30-28<br>— | Reserved |
| 27<br>TPP | Tamper Pin Polarity<br>Configures the polarity of the tamper pin.<br>0b - Tamper pin is active high.<br>1b - Tamper pin is active low. |
| 26<br>TFE | Tamper Filter Enable<br>Enables the passive input filter on the tamper pin.<br>0b - Input filter is disabled on the tamper pin.<br>1b - Input filter is enabled on the tamper pin. |
| 25<br>TPS | Tamper Pull Select<br>Configures the direction of the tamper pin pull resistor.<br>0b - Tamper pin pull resistor direction will assert the tamper pin.<br>1b - Tamper pin pull resistor direction will negate the tamper pin. |
| 24<br>TPE | Tamper Pull Enable<br>Enables the pull resistor on the tamper pin.<br>0b - Pull resistor is disabled on tamper pin.<br>1b - Pull resistor is enabled on tamper pin. |
| 23-0<br>— | Reserved |

## 54.3.1.17 RTC Write Access Register (WAR)

## 54.3.1.17.1 Offset

| Register | Offset |
|---|---|
| WAR | 800h |

## 54.3.1.17.2 Diagram




## 54.3.1.17.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>PCRW | Pin Configuration Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Pin Configuration Register are ignored.<br>1b - Writes to the Pin Configuration Register complete as normal. |
| 15<br>TIRW | Tamper Interrupt Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Tamper Interrupt Register are ignored.<br>1b - Writes to the Tamper Interrupt Register complete as normal. |
| 14<br>— | Reserved |
| 13<br>TDRW | Tamper Detect Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Tamper Detect Register are ignored.<br>1b - Writes to the Tamper Detect Register complete as normal. |
| 12<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 11<br>MCHW | Monotonic Counter High Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Monotonic Counter High Register are ignored.<br>1b - Writes to the Monotonic Counter High Register complete as normal. |
| 10<br>MCLW | Monotonic Counter Low Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Monotonic Counter Low Register are ignored.<br>1b - Writes to the Monotonic Counter Low Register complete as normal. |
| 9<br>MERW | Monotonic Enable Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Monotonic Enable Register are ignored.<br>1b - Writes to the Monotonic Enable Register complete as normal. |
| 8<br>TTSW | Tamper Time Seconds Write<br>After being cleared, this bit is set onlyby system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Tamper Time Seconds Register are ignored.<br>1b - Writes to the Tamper Time Seconds Register complete as normal. |
| 7<br>IERW | Interrupt Enable Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Interupt Enable Register are ignored.<br>1b - Writes to the Interrupt Enable Register complete as normal. |
| 6<br>LRW | Lock Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Lock Register are ignored.<br>1b - Writes to the Lock Register complete as normal. |
| 5<br>SRW | Status Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Status Register are ignored.<br>1b - Writes to the Status Register complete as normal. |
| 4<br>CRW | Control Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Control Register are ignored.<br>1b - Writes to the Control Register complete as normal. |
| 3<br>TCRW | Time Compensation Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Time Compensation Register are ignored.<br>1b - Writes to the Time Compensation Register complete as normal. |
| 2<br>TARW | Time Alarm Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Time Alarm Register are ignored.<br>1b - Writes to the Time Alarm Register complete as normal. |
| 1<br>TPRW | Time Prescaler Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Time Prescaler Register are ignored.<br>1b - Writes to the Time Prescaler Register complete as normal. |
| 0<br>TSRW | Time Seconds Register Write<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Writes to the Time Seconds Register are ignored. |

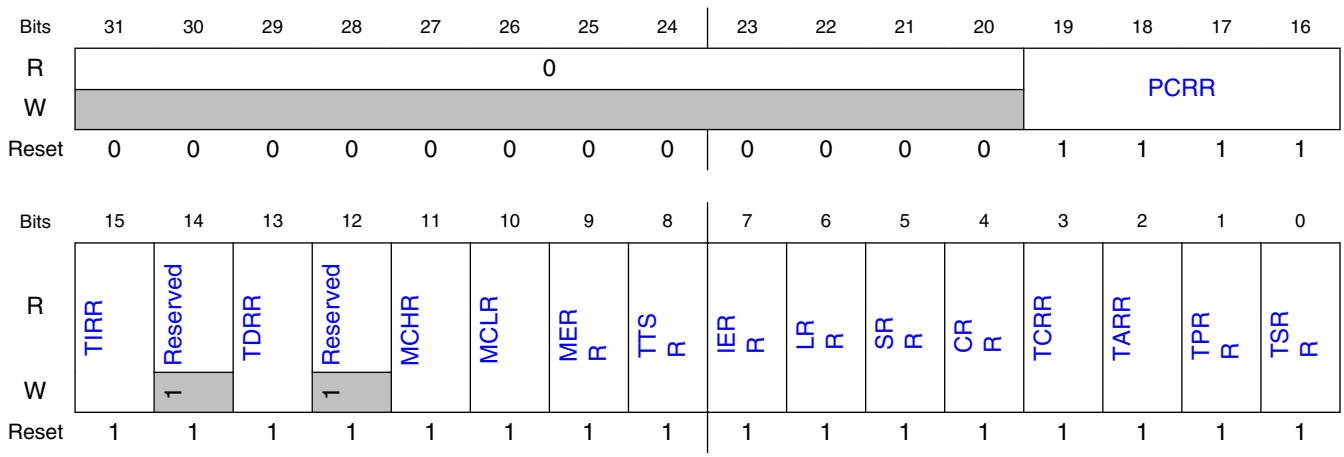
| Field | Function |
|---|---|
| | 1b - Writes to the Time Seconds Register complete as normal. |

## 54.3.1.18 RTC Read Access Register (RAR)

### 54.3.1.18.1 Offset

| Register | Offset |
|---|---|
| RAR | 804h |

### 54.3.1.18.2 Diagram




### 54.3.1.18.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>PCRR | Pin Configuration Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Pin Configuration Register are ignored.<br>1b - Reads to the Pin Configuration Register complete as normal. |
| 15<br>TIRR | Tamper Interrupt Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Tamper Interrupt Register are ignored.<br>1b - Reads to the Tamper Interrupt Register complete as normal. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14<br>— | Reserved |
| 13<br>TDRR | Tamper Detect Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Tamper Detect Register are ignored.<br>1b - Reads to the Tamper Detect Register complete as normal. |
| 12<br>— | Reserved |
| 11<br>MCHR | Monotonic Counter High Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Monotonic Counter High Register are ignored.<br>1b - Reads to the Monotonic Counter High Register complete as normal. |
| 10<br>MCLR | Monotonic Counter Low Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Monotonic Counter Low Register are ignored.<br>1b - Reads to the Monotonic Counter Low Register complete as normal. |
| 9<br>MERR | Monotonic Enable Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Monotonic Enable Register are ignored.<br>1b - Reads to the Monotonic Enable Register complete as normal. |
| 8<br>TTSR | Tamper Time Seconds Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Tamper Time Seconds Register are ignored.<br>1b - Reads to the Tamper Time Seconds Register complete as normal. |
| 7<br>IERR | Interrupt Enable Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Interrupt Enable Register are ignored.<br>1b - Reads to the Interrupt Enable Register complete as normal. |
| 6<br>LRR | Lock Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Lock Register are ignored.<br>1b - Reads to the Lock Register complete as normal. |
| 5<br>SRR | Status Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Status Register are ignored.<br>1b - Reads to the Status Register complete as normal. |
| 4<br>CRR | Control Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Control Register are ignored.<br>1b - Reads to the Control Register complete as normal. |
| 3<br>TCRR | Time Compensation Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Time Compensation Register are ignored.<br>1b - Reads to the Time Compensation Register complete as normal. |
| 2<br>TARR | Time Alarm Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Reads to the Time Alarm Register are ignored.<br>1b - Reads to the Time Alarm Register complete as normal. |
| 1<br>TPRR | Time Prescaler Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Time Pprescaler Register are ignored.<br>1b - Reads to the Time Prescaler Register complete as normal. |
| 0<br>TSRR | Time Seconds Register Read<br>After being cleared, this bit is set only by system reset. It is not affected by VBAT POR or software reset.<br>0b - Reads to the Time Seconds Register are ignored.<br>1b - Reads to the Time Seconds Register complete as normal. |

# 54.4 Functional description

## 54.4.1 Power, clocking, and reset

The RTC is an always powered block that remains active in all low power modes and is powered by the battery power supply (VBAT). The battery power supply ensures that the RTC registers retain their state during chip power-down and that the RTC time counter remains operational.

The time counter within the RTC is clocked by default from a 32.768 kHz clock and can supply this clock to other peripherals. The 32.768 kHz clock can be sourced from an external crystal using the oscillator that is part of the RTC module. Alternatively, the time counter can be clocked by a LPO 1 kHz clock and the prescaler will increment by 32 for each LPO clock.

The RTC includes its own analog POR block, which generates a VBAT power-on-reset signal whenever the RTC module is powered up and initializes all RTC registers to their default state. A software reset bit can also initialize all RTC registers. The RTC also monitors the chip power supply and electrically isolates itself when the rest of the chip is powered down.

**NOTE**

An attempt to access an RTC register, except the access control registers, results in a bus error when:

- VBAT is powered down,
- the RTC is electrically isolated, or
- VBAT POR is asserted.

#### 54.4.1.1 Oscillator control

The 32.768 kHz crystal oscillator is disabled at VBAT POR and must be enabled by software. After enabling the cystal oscillator, wait the oscillator startup time before setting SR[TCE] or using the oscillator clock external to the RTC.

The crystal oscillator includes tunable capacitors that can be configured by software. Do not change the capacitance unless the oscillator is disabled.

#### 54.4.1.2 Software reset

Writing 1 to CR[SWR] forces the equivalent of a VBAT POR to the rest of the RTC module. CR[SWR] is not affected by the software reset and must be cleared by software. The access control registers are not affected by either VBAT POR or the software reset; they are reset by the chip reset.

#### 54.4.1.3 Supervisor access

When the supervisor access control bit is clear, only supervisor mode software can write to the RTC registers or read the RTC tamper and monotonic registers, non-supervisor mode software will generate a bus error. Both supervisor and non-supervisor mode software can always read the other RTC registers.

### 54.4.2 Time counter

The time counter consists of a 32-bit seconds counter that increments once every second and a 16-bit prescaler register that increments once every 32.768 kHz clock cycle. There is also the option to clock the prescaler using a 1 kHz LPO that increments the prescaler by 32 on every clock cycle.

Reading the time counter (either seconds or prescaler) while it is incrementing may return invalid data due to synchronization of the read data bus. If it is necessary for software to read the prescaler or seconds counter when they could be incrementing, it is recommended that two read accesses are performed and that software verifies that the same data was returned for both reads.

The time seconds register and time prescaler register can be written only when SR[TCE] is clear. Always write to the prescaler register before writing to the seconds register, because the seconds register increments on the falling edge of bit 14 of the prescaler register.

The time prescaler register increments provided SR[TCE] is set, SR[TIF] is clear, SR[TOF] is clear, and the 32.768 kHz (or 1 kHz) clock source is present. After enabling the oscillator, wait the oscillator startup time before setting SR[TCE] to allow time for the oscillator clock output to stabilize.

If the time seconds register overflows then the SR[TOF] will set and the time prescaler register will stop incrementing. Clear SR[TOF] by initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TOF] is set.

SR[TIF] is set on VBAT POR, software reset, and when an enabled tamper source is detected. It is cleared by clearing the enabled tamper detect flag and then initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TIF] is set.

### 54.4.3 Compensation

The compensation logic provides an accurate and wide compensation range and can correct errors as high as 3906 ppm and as low as 0.12 ppm. The compensation factor must be calculated externally to the RTC and supplied by software to the compensation register. The RTC itself does not calculate the amount of compensation that is required, although the 1 Hz clock is output to an external pin in support of external calibration logic.

Crystal compensation can be supported by using firmware and crystal characteristics to determine the compensation amount. Temperature compensation can be supported by firmware that periodically measures the external temperature via ADC and updates the compensation register based on a look-up table that specifies the change in crystal frequency over temperature.

The compensation logic alters the number of 32.768 kHz clock cycles it takes for the prescaler register to overflow and increment the time seconds counter. The time compensation value is used to adjust the number of clock cycles between -127 and +128. Cycles are added or subtracted from the prescaler register when the prescaler register equals 0x3FFF and then increments. The compensation interval is used to adjust the frequency at which the time compensation value is used, that is, from once a second to once every 256 seconds.

Updates to the time compensation register will not take effect until the next time the time seconds register increments and provided the previous compensation interval has expired. When the compensation interval is set to other than once a second then the compensation is applied in the first second interval and the remaining second intervals receive no compensation.

Compensation is disabled by configuring the time compensation register to zero.

When the prescaler is configured to increment using the 1 kHz LPO, the effective compensation value is divided by 32 and can only adjust the number of clock cycles between -4 and +3.

### 54.4.4 Time alarm

The Time Alarm register (TAR), SR[TAF], and IER[TAIE] allow the RTC to generate an interrupt at a predefined time. The 32-bit TAR is compared with the 32-bit Time Seconds register (TSR) each time it increments. SR[TAF] will set when TAR equals TSR and TSR increments.

SR[TAF] is cleared by writing TAR. This will usually be the next alarm value, although writing a value that is less than TSR, such as 0, will prevent SR[TAF] from setting again. SR[TAF] cannot otherwise be disabled, although the interrupt it generates is enabled or disabled by IER[TAIE].

### 54.4.5 Update mode

The Update Mode field in the Control register (CR[UM]) configures software write access to the Time Counter Enable (SR[TCE]) field. When CR[UM] is clear, SR[TCE] can be written only when LR[SRL] is set. When CR[UM] is set, SR[TCE] can also be written when SR[TCE] is clear or when SR[TIF] or SR[TOF] are set. This allows the time seconds and prescaler registers to be initialized whenever time is invalidated, while preventing the time seconds and prescaler registers from being changed on the fly. When LR[SRL] is set, CR[UM] has no effect on SR[TCE].

CR[UM] also configures software write access to the Monotonic Counter Enable (MER[MCE]) bit. When CR[UM] is clear, MER[MCE] can be written only when LR[MEL] is set. When CR[UM] is set, MER[MCE] can also be written when MER[MCE] is clear or when SR[TIF] or SR[MOF] are set. This allows the monotonic counter register to be initialized whenever the monotonic counter is invalid, while preventing the monotonic counter from being changed on the fly. When LR[MEL] is set, CR[UM] has no effect on MCR[MCE].

### 54.4.6 Monotonic counter

The 64-bit Monotonic Counter is a counter that cannot be exhausted or return to any previous value, once it has been initialized. If the monotonic overflow flag is set, the monotonic counter returns zero and does not increment.

Depending on the value of the monotonic counter enable bit, writing to the monotonic counter either initializes the register with the value written, or increments the register by one (and the value written is ignored).

When the monotonic counter is enabled, the monotonic counter high increments on either a write to the monotonic counter high register or if the monotonic counter low register overflows (due to a write to the monotonic counter low register). The monotonic overflow flag sets when the monotonic counter high register overflows and is cleared by writing the monotonic counter high register when the monotonic counter is disabled.

The monotonic counter is held in reset whenever the time invalid flag is set. Always clear the time invalid flag before initializing the monotonic counter.

### 54.4.7 Tamper detect

Each external tamper pin and internal tamper source can be configured to generate a tamper interrupt. The contents of the seconds register are latched in the tamper time seconds register when the tamper interrupt asserts. Read the tamper time seconds register before clearing the flag in the tamper detect register.

Writing the tamper time seconds register at any time will set the time invalid flag. Since this is a software initiated tamper, there is no status flag to indicate this tamper source. To disable the software initiated tamper, lock the tamper time seconds register to prevent write accesses to that register.

### 54.4.8 Register lock

The Lock register (LR) can be used to block write accesses to certain registers until the next VBAT POR or software reset. Locking the Control register (CR) will disable the software reset. Locking LR will block future updates to LR.

Write accesses to a locked register are ignored and do not generate a bus error.

### 54.4.9 Access control

The read access and write access registers are implemented in the chip power domain and reset on the chip reset. They are not affected by the VBAT POR or the software reset. They are used to block read or write accesses to each register until the next chip system reset. When accesses are blocked, the bus access is not seen in the VBAT power supply and does not generate a bus error.

### 54.4.10 Interrupt

The RTC interrupt is asserted whenever a status flag and the corresponding interrupt enable bit are both set. It is always asserted on VBAT POR, and software reset, and when the VBAT power supply is powered down. The RTC interrupt is enabled at the chip level by enabling the chip-specific RTC clock gate control bit. The RTC interrupt can be used to wakeup the chip from any low-power mode. If the RTC wakeup pin is enabled and the chip is powered down, the RTC interrupt will cause the wakeup pin to assert.

The RTC interrupt is also asserted when a configured tamper flag asserts and the corresponding tamper interrupt enable bit is set. Both internal tamper events and external tamper pins can cause the tamper interrupt to assert. The contents of the seconds register are recorded in the tamper time seconds register when the tamper interrupt asserts.

The RTC seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector that is generated once a second and requires no software overhead (there is no corresponding status flag to clear). It is enabled in the RTC by the time seconds interrupt enable bit and enabled at the chip level by setting the chip-specific RTC clock gate control bit. The frequency of the seconds interrupt defaults to 1 Hz, but can instead be configured to trigger every 2, 4, 8, 16, 32, 64 or 128 Hz. The RTC seconds interrupt does not cause the RTC wakeup pin to assert.

# Chapter 55
# Low Power Inter-Integrated Circuit (LPI2C)

## 55.1 Chip-specific LPI2C information

### 55.1.1 LPI2C instantiations

The following list describes the LPI2C instantiations on this device:

- Instantiations: 4
- Configuration: Master and Slave
- Receive FIFO size: 4 entries
- Transmit FIFO size: 4 entries

## 55.2 Introduction

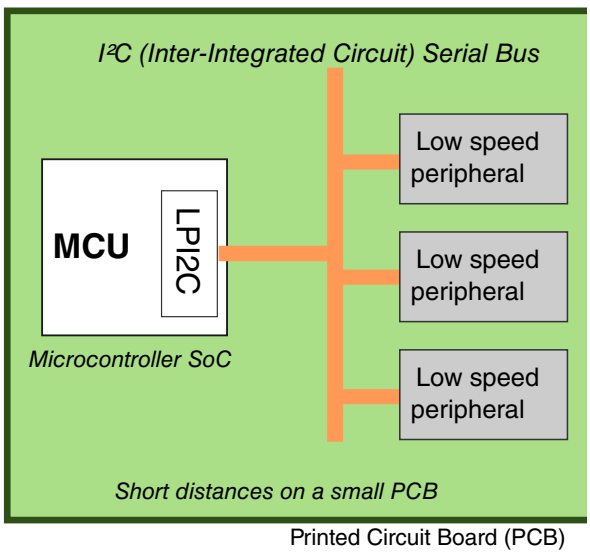



**Figure 55-1. I²C bus connects MCU to low speed peripherals on PCB**

The LPI2C is a low power Inter-Integrated Circuit (I2C) module that supports an efficient interface to an I2C bus as a master and/or as a slave.

- The LPI2C implements logic support for standard-mode, fast-mode, fast-mode plus and ultra-fast modes of operation.
- The LPI2C is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The LPI2C can continue operating in stop modes if an appropriate clock is available.

The LPI2C module also complies with the System Management Bus (SMBus) Specification, version 2. The SMBus is a single-ended simple two-wire bus, which is typically used for low bandwidth communications.

**NOTE**

The I²C (Inter-Integrated Circuit) serial bus is multi-master, multi-slave, packet-switched, and single-ended, and is often used to attach microcontroller ICs to lower-speed peripheral ICs.

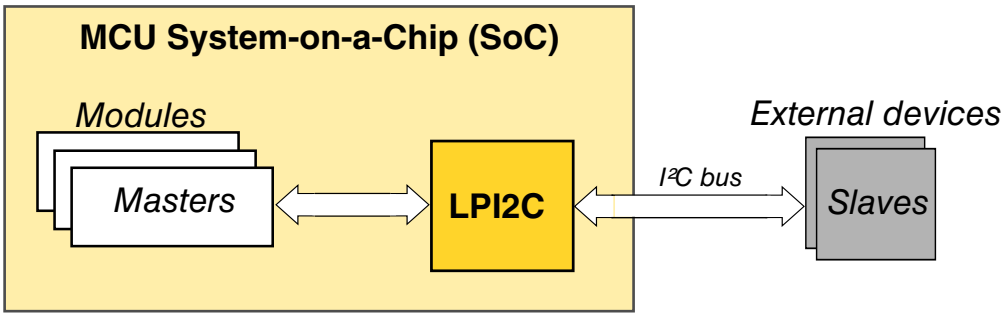



**Figure 55-2. LPI2C connects masters in MCU to external slaves on PCB**

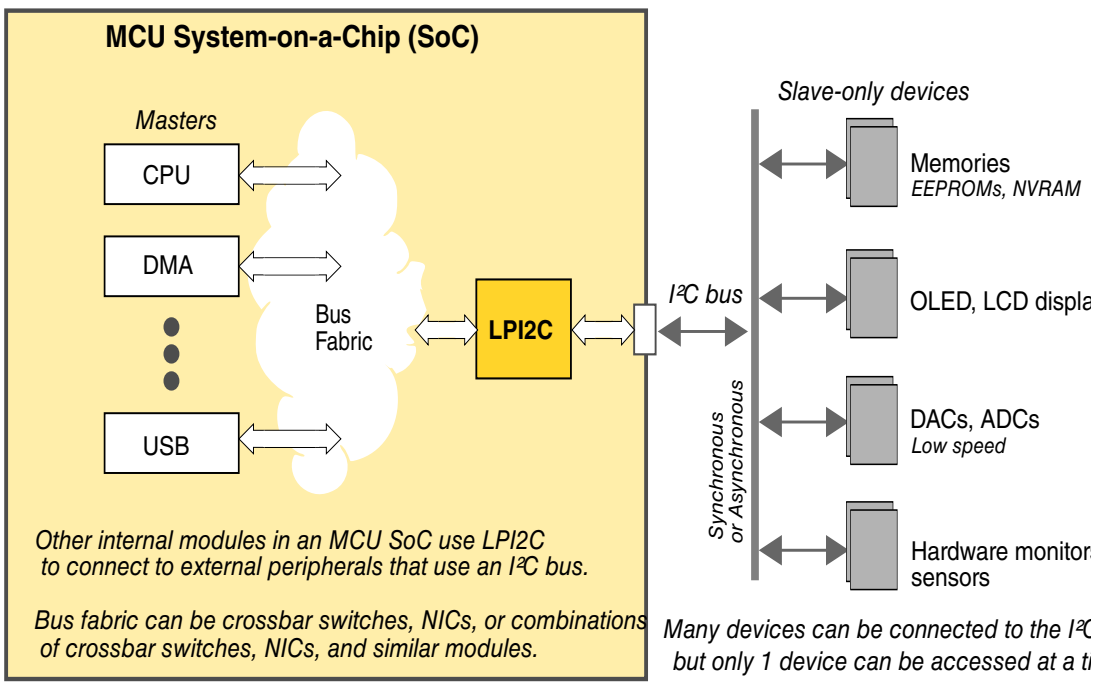



**Figure 55-3. Typical LPI2C connection scheme**

## 55.2.1 Features

The LPI2C supports:

- Standard, Fast, Fast+ and Ultra Fast modes are supported
- High speed mode (HS) in slave mode
- High speed mode (HS) in master mode, if SCL pin implements current source pull-up (device-specific)

- Multi-master support, including synchronization and arbitration. Multi-master means any number of master nodes can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent).
- Clock stretching: Sometimes multiple I2C nodes may be driving the lines at the same time. If any I2C node is driving a line low, then that line will be low. I2C nodes that are starting to transmit a logical one (by letting the line float high) can detect that the line is low, and thereby know that another I2C node is active at the same time.
  - When node detection is used on the SCL line, it is called *clock stretching*, and clock stretching is used as a I2C flow control mechanism for multiple laves.
  - When node detection is used on the SDA line, it is called *arbitration*, and arbitration ensures that there is only one I2C node transmitter at a time.
- General call, 7-bit and 10-bit addressing
- Software reset, START byte and Device ID (also require software support)

The LPI2C master supports:

- Command/transmit FIFO of 4words.
- Receive FIFO of 4words.
- Command FIFO will wait for idle I2C bus before initiating transfer
- Command FIFO can initiate (repeated) START and STOP conditions and one or more master-receiver transfers
- STOP condition can be generated from command FIFO, or generated automatically when the transmit FIFO is empty
- Host request input to control the start time of an I2C bus transfer
- Flexible receive data match can generate interrupt on data match and/or discard unwanted data
- Flag and optional interrupt to signal Repeated START condition, STOP condition, loss of arbitration, unexpected NACK, and command word errors
- Supports configurable bus idle timeout and pin-stuck-low timeout

The LPI2C slave supports:

- Separate I2C slave registers to minimize software overhead because of master/slave switching
- Support for 7-bit or 10-bit addressing, address range, SMBus alert and general call address
- Transmit data register that supports interrupt or DMA requests
- Receive data register that supports interrupt or DMA requests
- Software-controllable ACK or NACK, with optional clock stretching on ACK/NACK bit
- Configurable clock stretching, to avoid transmit FIFO underrun and receive FIFO overrun errors
- Flag and optional interrupt at end of packet, STOP condition, or bit error detection

## 55.2.2 Block Diagram

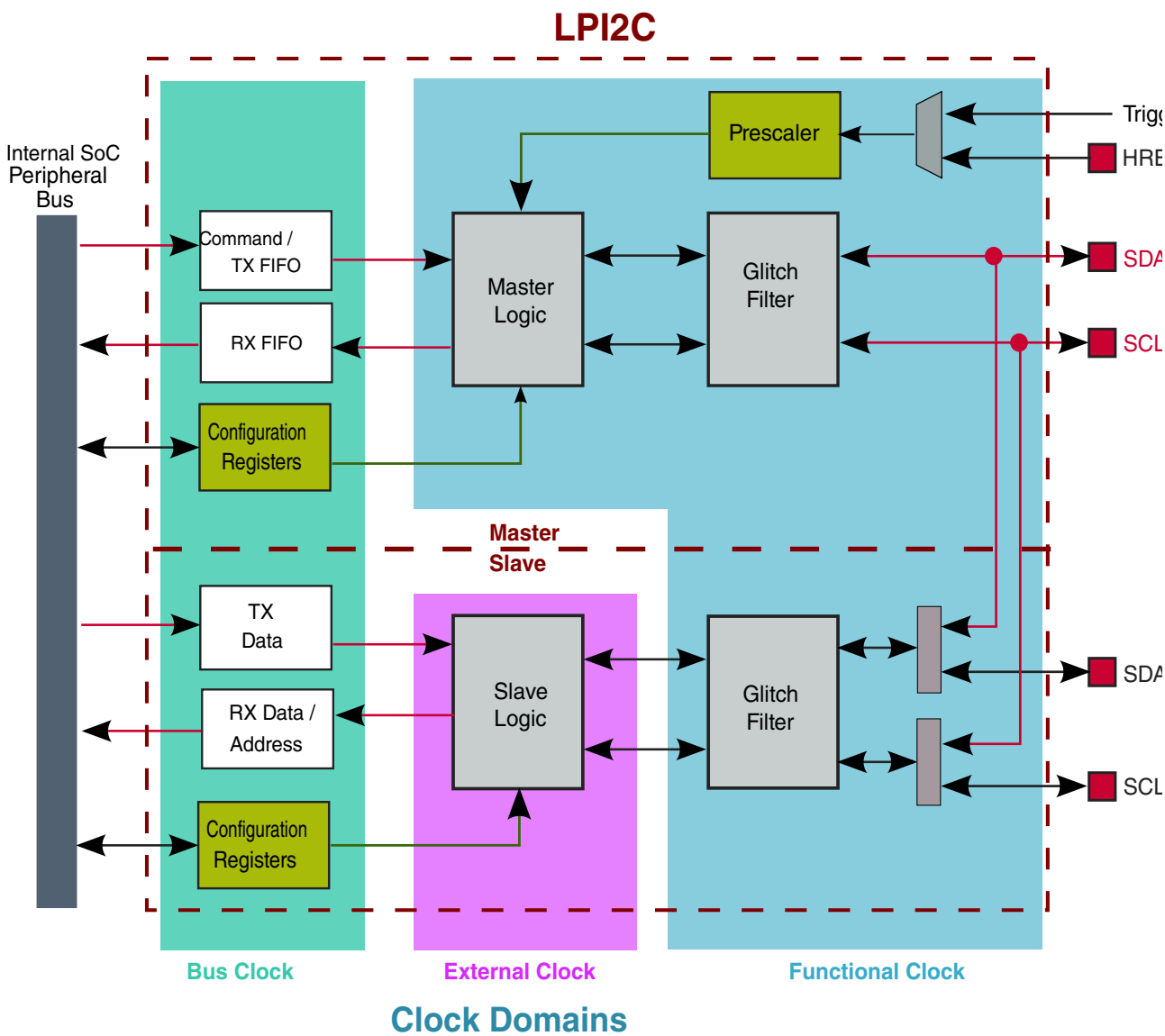



**Figure 55-4. LPI2C block diagram**

## 55.2.3 Modes of operation

**Table 55-1. Chip modes supported by the LPI2C module**

| Chip mode | LPI2C Operation |
|---|---|
| Run | Normal operations |

*Table continues on the next page...*

**Table 55-1. Chip modes supported by the LPI2C module (continued)**

| Chip mode | LPI2C Operation |
|---|---|
| Stop/Wait | Can continue operating in stop/wait modes if the Doze Enable bit (MCR[DOZEN]) is clear and the LPI2C is using an external or internal clock source that remains operating during stop/wait modes. |
| Low Power Stop (also called Low Leakage Stop or LLS) | Before entering low power stop mode, the LPI2C will wait for the current transfer to finish any pending operation, while temporarily ignoring The Doze Enable (MCR[DOZEN]) bit. |
| Debug | Can continue operating in debug mode if the Debug Enable bit (MCR[DBGE]) is set. |

## 55.2.4 Signal Descriptions

**Table 55-2. Signals**

| Signal | Name | 2-Wire Scheme | 4-Wire Scheme | I/O |
|---|---|---|---|---|
| SCL | LPI2C clock line | SCL | In 4-wire mode, this is the SCL input pin. | I/O |
| SDA | LPI2C data line | SDA | In 4-wire mode, this is the SDA input pin. | I/O |
| HREQ | Host request | If host request is asserted and the I2C bus is idle, then it will initiate an LPI2C master transfer. | | I |
| SCLS | Secondary I2C clock line | Not used | In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, then this the LPI2C slave SCL pin. | I/O |
| SDAS | Secondary I2C data line | Not used | In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, then this the LPI2C slave SDA pin. | I/O |

## 55.2.5 Wiring options

LPI2C can be used to implement 2-wire or 4-wire $I^2C$ serial busses.

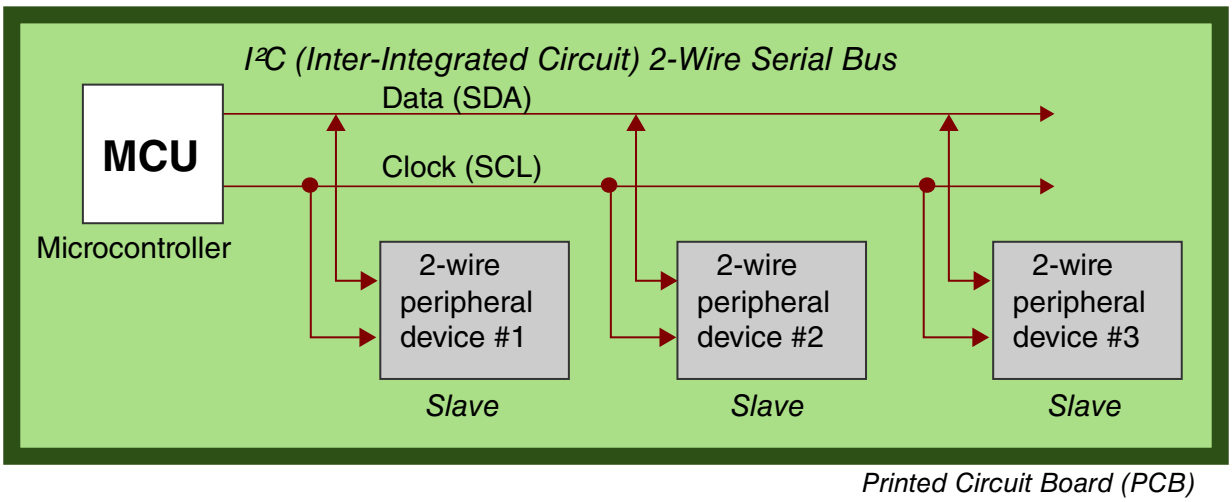



**Figure 55-5. 2-Wire scheme**

Some applications can provide a lot of load and noise on the $I^2C$ bus; to ensure robust $I^2C$ operations, a 4-wire interface with the MCU can be used, splitting the 2 lines into inputs and outputs. Using a few transistors, resistors and diodes, designers can make their own inexpensive line drivers.

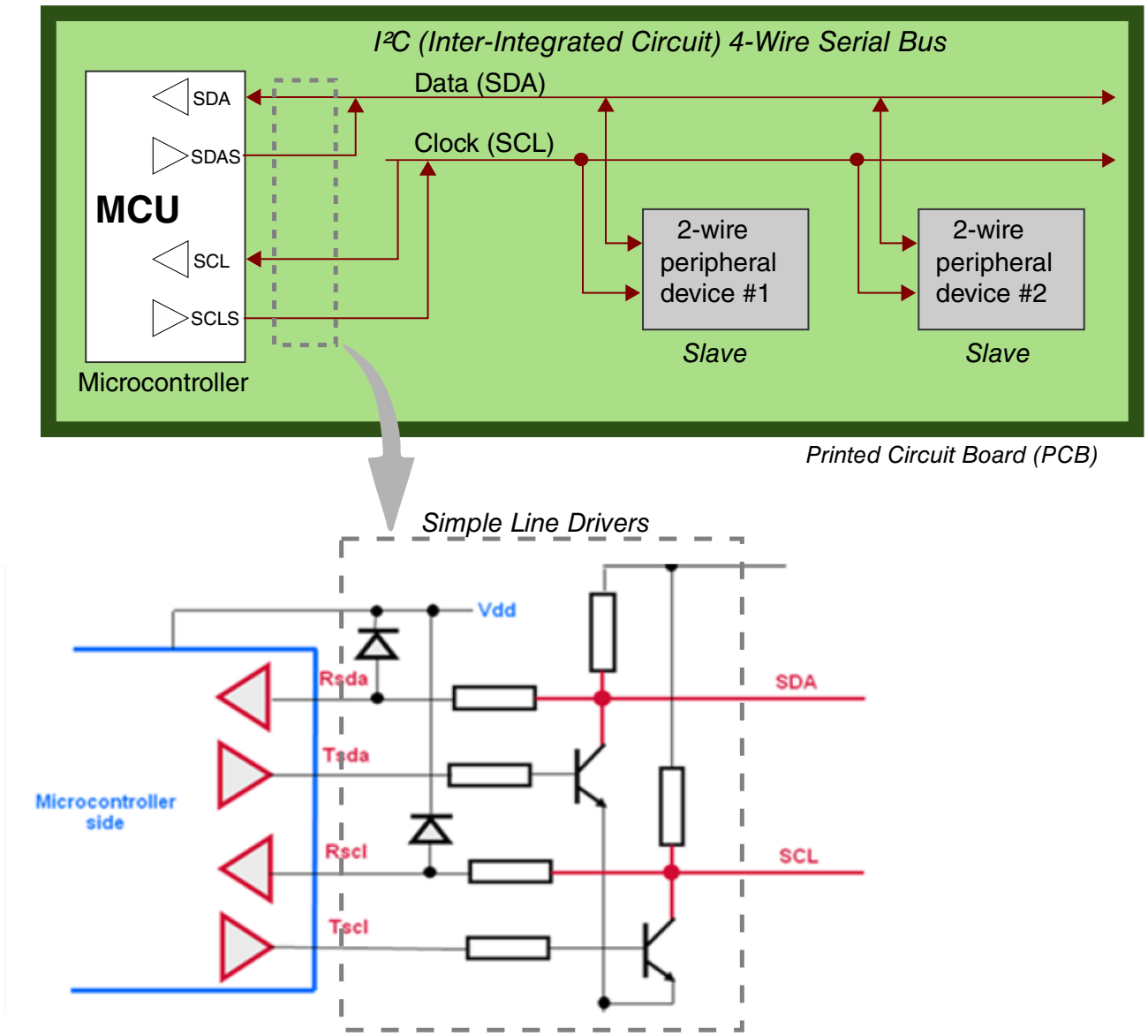



**Figure 55-6. 4-Wire scheme**

## 55.3 Memory Map and Registers

### 55.3.1 LPI2C0 register descriptions

### 55.3.1.1 LPI2C0 Memory map

LPI2C0 base address: 4003_A000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Master Control Register (MCR) | 32 | RW | 0000_0000h |
| 14h | Master Status Register (MSR) | 32 | W1C | 0000_0001h |
| 18h | Master Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |
| 1Ch | Master DMA Enable Register (MDER) | 32 | RW | 0000_0000h |
| 20h | Master Configuration Register 0 (MCFGR0) | 32 | RW | 0000_0000h |
| 24h | Master Configuration Register 1 (MCFGR1) | 32 | RW | 0000_0000h |
| 28h | Master Configuration Register 2 (MCFGR2) | 32 | RW | 0000_0000h |
| 2Ch | Master Configuration Register 3 (MCFGR3) | 32 | RW | 0000_0000h |
| 40h | Master Data Match Register (MDMR) | 32 | RW | 0000_0000h |
| 48h | Master Clock Configuration Register 0 (MCCR0) | 32 | RW | 0000_0000h |
| 50h | Master Clock Configuration Register 1 (MCCR1) | 32 | RW | 0000_0000h |
| 58h | Master FIFO Control Register (MFCR) | 32 | RW | 0000_0000h |
| 5Ch | Master FIFO Status Register (MFSR) | 32 | RO | 0000_0000h |
| 60h | Master Transmit Data Register (MTDR) | 32 | WO | 0000_0000h |
| 70h | Master Receive Data Register (MRDR) | 32 | RO | 0000_4000h |
| 110h | Slave Control Register (SCR) | 32 | RW | 0000_0000h |
| 114h | Slave Status Register (SSR) | 32 | W1C | 0000_0000h |
| 118h | Slave Interrupt Enable Register (SIER) | 32 | RW | 0000_0000h |
| 11Ch | Slave DMA Enable Register (SDER) | 32 | RW | 0000_0000h |
| 124h | Slave Configuration Register 1 (SCFGR1) | 32 | RW | 0000_0000h |
| 128h | Slave Configuration Register 2 (SCFGR2) | 32 | RW | 0000_0000h |
| 140h | Slave Address Match Register (SAMR) | 32 | RW | 0000_0000h |
| 150h | Slave Address Status Register (SASR) | 32 | RO | 0000_4000h |
| 154h | Slave Transmit ACK Register (STAR) | 32 | RW | 0000_0000h |
| 160h | Slave Transmit Data Register (STDR) | 32 | WO | 0000_0000h |
| 170h | Slave Receive Data Register (SRDR) | 32 | RO | 0000_4000h |

### 55.3.1.2 Version ID Register (VERID)

### 55.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 55.3.1.2.2 Function

.

### 55.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 55.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | Returns the major version number for the module design specification. Read-only field. |
| 23-16 | Minor Version Number |
| MINOR | Returns the minor version number for the module design specification. Read-only field. |
| 15-0 | Feature Specification Number |
| FEATURE | Returns the feature set number. Read-only field.<br>0000000000000010b - Master only, with standard feature set<br>0000000000000011b - Master and slave, with standard feature set |

## 55.3.1.3 Parameter Register (PARAM)

#### 55.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 55.3.1.3.2 Function

.

#### 55.3.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MRXFIFO | | | | 0 | | | | MTXFIFO | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 55.3.1.3.4 Fields

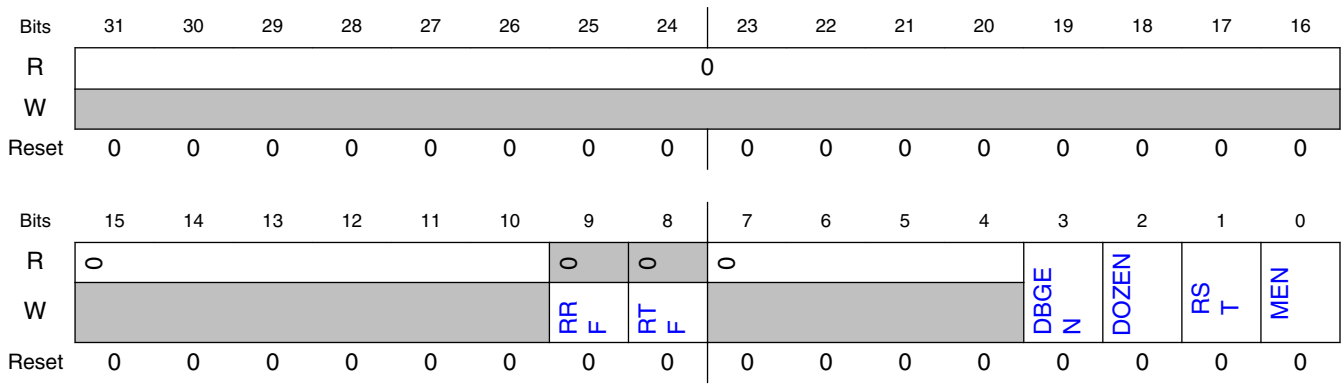
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>MRXFIFO | Master Receive FIFO Size<br>Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$ |
| 7-4<br>— | Reserved |
| 3-0<br>MTXFIFO | Master Transmit FIFO Size<br>Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$ |

### 55.3.1.4 Master Control Register (MCR)

### 55.3.1.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 10h |

### 55.3.1.4.2 Diagram




### 55.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Master is disabled in debug mode<br>1b - Master is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the master in Doze mode<br>0b - Master is enabled in Doze mode<br>1b - Master is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>MEN | Master Enable<br>0b - Master logic is disabled<br>1b - Master logic is enabled |

## 55.3.1.5 Master Status Register (MSR)

### 55.3.1.5.1 Offset

| Register | Offset |
|---|---|
| MSR | 14h |

### 55.3.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 55.3.1.5.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24<br>MBF | Master Busy Flag<br>0b - I2C Master is idle<br>1b - I2C Master is busy |
| 23-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set.<br>0b - Have not received matching data<br>1b - Have received matching data |
| 13<br>PLTF | Pin Low Timeout Flag<br>Will set when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR3[PINLOW]), even when the LPI2C master is idle.<br>• Software is responsible for resolving the pin low condition.<br>• Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues.<br>• Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared.<br>0b - Pin low timeout has not occurred or is disabled<br>1b - Pin low timeout has occurred |
| 12<br>FEF | FIFO Error Flag<br>Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared.<br>0b - No error<br>1b - Master sending or receiving data without a START condition |
| 11<br>ALF | Arbitration Lost Flag<br>Set:<br>• if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus<br>• or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data<br>When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag has been cleared.<br>0b - Master has not lost arbitration<br>1b - Master has lost arbitration |
| 10<br>NDF | NACK Detect Flag<br>Set if the LPI2C master detects a NACK when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated.<br>0b - Unexpected NACK was not detected<br>1b - Unexpected NACK was detected |
| 9<br>SDF | STOP Detect Flag<br>Set when the LPI2C master generates a STOP condition.<br>0b - Master has not generated a STOP condition<br>1b - Master has generated a STOP condition |
| 8<br>EPF | End Packet Flag<br>Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition.<br>0b - Master has not generated a STOP or Repeated START condition<br>1b - Master has generated a STOP or Repeated START condition |
| 7-2 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER.<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER.<br>0b - Transmit data is not requested<br>1b - Transmit data is requested |

## 55.3.1.6 Master Interrupt Enable Register (MIER)

### 55.3.1.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 18h |

### 55.3.1.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMIE | PLTIE | FEIE | ALIE | NDIE | SDIE | EPIE | 0 | | | | | | RDIE | TDIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14 | Data Match Interrupt Enable |

*Table continues on the next page...*

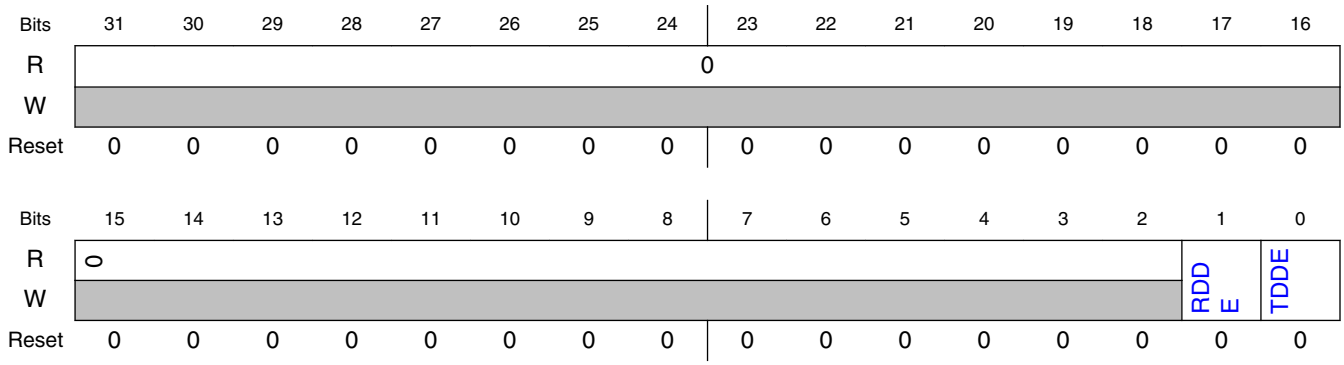
| Field | Function |
|---|---|
| DMIE | 0b - Disabled<br>1b - Enabled |
| 13<br>PLTIE | Pin Low Timeout Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>FEIE | FIFO Error Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>ALIE | Arbitration Lost Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>NDIE | NACK Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>EPIE | End Packet Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.1.7 Master DMA Enable Register (MDER)

### 55.3.1.7.1 Offset

| Register | Offset |
|---|---|
| MDER | 1Ch |

### 55.3.1.7.2 Diagram




### 55.3.1.7.3 Fields

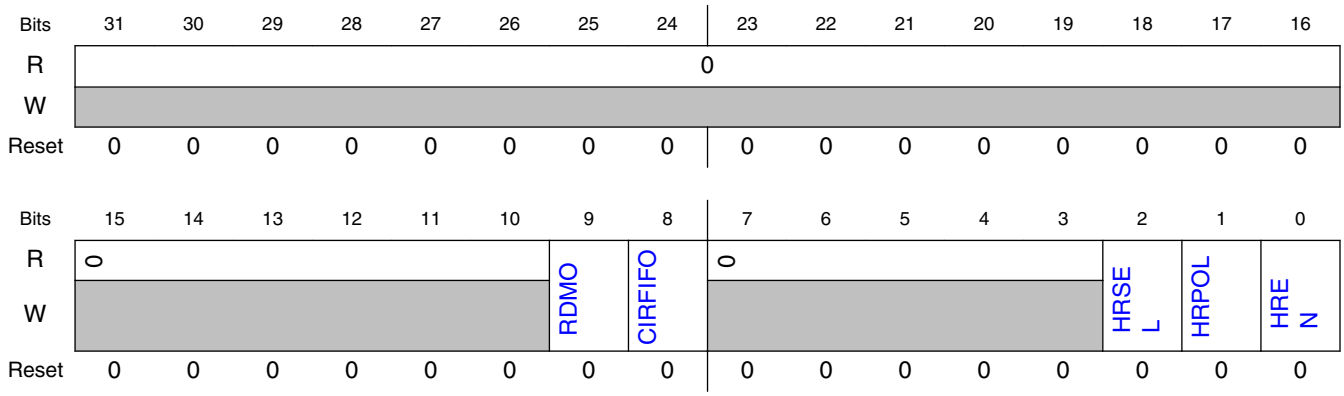
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.1.8 Master Configuration Register 0 (MCFGR0)

### 55.3.1.8.1 Offset

| Register | Offset |
|---|---|
| MCFGR0 | 20h |

## 55.3.1.8.2 Diagram




## 55.3.1.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change).<br>0b - Host request input is pin HREQ<br>1b - Host request input is input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field should remain static (Host Request Polarity should not change).<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable |

| Field | Function |
|---|---|
| | When enabled, the LPI2C master will only initiate a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request.<br>0b - Host request input is disabled<br>1b - Host request input is enabled |

## 55.3.1.9 Master Configuration Register 1 (MCFGR1)

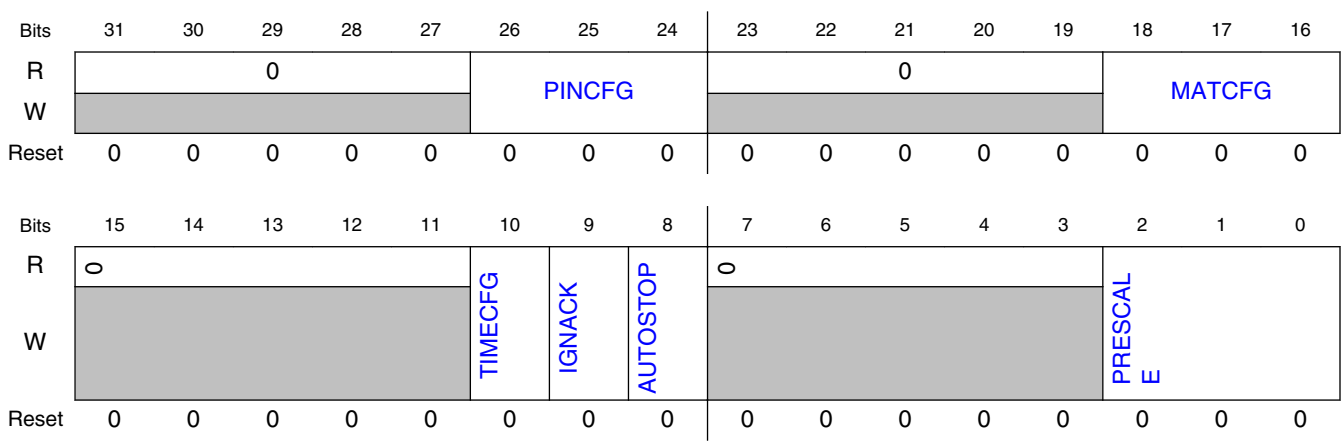
### 55.3.1.9.1 Offset

| Register | Offset |
|---|---|
| MCFGR1 | 24h |

### 55.3.1.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

### 55.3.1.9.3 Diagram




### 55.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>PINCFG | Pin Configuration |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Configures the pin mode for LPI2C.<br>Table 55-3. 2-pin / 4-pin pin configurations for masters and slaves
<table>
<tr><th>PINCFG</th><th>SCL / SDA pins</th><th>SCLS / SDAS pins</th></tr>
<tr><td>000</td><td>Bi-directional open drain for master and slave</td><td>Not used</td></tr>
<tr><td>001</td><td>Output-only (ultra-fast mode) open drain for master and slave</td><td>Not used</td></tr>
<tr><td>010</td><td>Bi-directional push-pull for master and slave</td><td>Not used</td></tr>
<tr><td>011</td><td>Input only for master and slave</td><td>Output-only push-pull for master and slave</td></tr>
<tr><td>100</td><td>Bi-directional open drain for master</td><td>Bi-directional open drain for slave</td></tr>
<tr><td>101</td><td>Output-only (ultra-fast mode) open drain for master</td><td>Output-only open drain for slave</td></tr>
<tr><td>110</td><td>Bi-directional push-pull for master</td><td>Bi-directional push-pull for slave</td></tr>
<tr><td>111</td><td>Input only for master and slave</td><td>Inverted output-only push-pull for master and slave</td></tr>
</table>
000b - 2-pin open drain mode<br>001b - 2-pin output only mode (ultra-fast mode)<br>010b - 2-pin push-pull mode<br>011b - 4-pin push-pull mode<br>100b - 2-pin open drain mode with separate LPI2C slave<br>101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave<br>110b - 2-pin push-pull mode with separate LPI2C slave<br>111b - 4-pin push-pull mode (inverted outputs)</td></tr>
<tr><td>23-19<br>—</td><td>Reserved</td></tr>
<tr><td>18-16<br>MATCFG</td><td>Match Configuration<br>Configures the condition that will cause the DMF to set.<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match is enabled (1st data word equals MATCH0 OR MATCH1)<br>011b - Match is enabled (any data word equals MATCH0 OR MATCH1)<br>100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1)<br>101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1)<br>110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1)<br>111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1)</td></tr>
<tr><td>15-11<br>—</td><td>Reserved</td></tr>
<tr><td>10<br>TIMECFG</td><td>Timeout Configuration<br>0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout<br>1b - Pin Low Timeout Flag will set if either SCL or SDA is low for longer than the configured timeout</td></tr>
<tr><td>9<br>IGNACK</td><td>IGNACK<br>When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode.<br>0b - LPI2C Master will receive ACK and NACK normally<br>1b - LPI2C Master will treat a received NACK as if it (NACK) was an ACK</td></tr>
</table>

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>AUTOSTOP | Automatic STOP Generation<br>When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command.<br>0b - No effect<br>1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy |
| 7-3<br>— | Reserved |
| 2-0<br>PRESCALE | Prescaler<br>Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 55.3.1.10 Master Configuration Register 2 (MCFGR2)

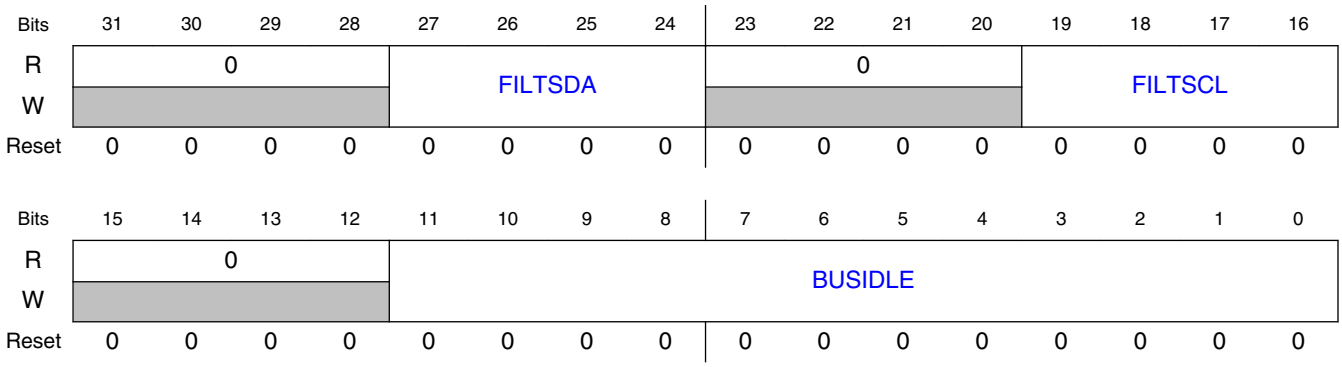
### 55.3.1.10.1 Offset

| Register | Offset |
|---|---|
| MCFGR2 | 28h |

### 55.3.1.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

## 55.3.1.10.3 Diagram




## 55.3.1.10.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C master digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C master digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 15-12<br>— | Reserved |
| 11-0<br>BUSIDLE | Bus Idle Timeout<br>Configures the bus idle timeout period in clock cycles.<br>• If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition<br>• When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled |

## 55.3.1.11 Master Configuration Register 3 (MCFGR3)

### 55.3.1.11.1 Offset

| Register | Offset |
|---|---|
| MCFGR3 | 2Ch |

### 55.3.1.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

### 55.3.1.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PINLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINLOW | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.11.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-8<br>PINLOW | Pin Low Timeout<br>Configures the pin low timeout flag in clock cycles.<br>• If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then PLTF is set<br>• When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled |
| 7-0<br>— | Reserved |

## 55.3.1.12 Master Data Match Register (MDMR)

### 55.3.1.12.1 Offset

| Register | Offset |
|---|---|
| MDMR | 40h |

### 55.3.1.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

### 55.3.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>MATCH1 | Match 1 Value<br>Compared against the received data when receive data match is enabled. |
| 15-8<br>— | Reserved |
| 7-0<br>MATCH0 | Match 0 Value<br>Compared against the received data when receive data match is enabled. |

## 55.3.1.13 Master Clock Configuration Register 0 (MCCR0)

### 55.3.1.13.1 Offset

| Register | Offset |
|---|---|
| MCCR0 | 48h |

### 55.3.1.13.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

### 55.3.1.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DATAVD | | | | | | 0 | | SETHOLD | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | CLKHI | | | | | | 0 | | CLKLO | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.13.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br><br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.1.14 Master Clock Configuration Register 1 (MCCR1)

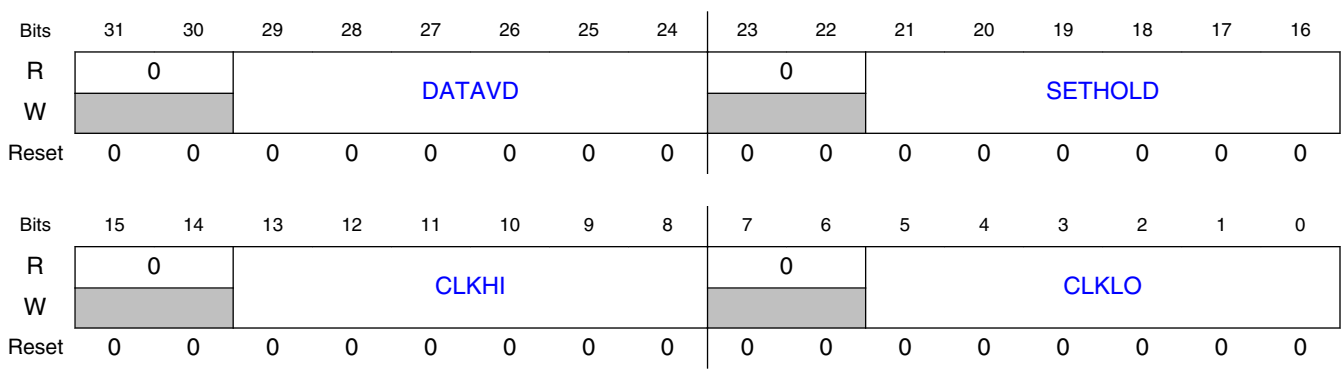
### 55.3.1.14.1 Offset

| Register | Offset |
|---|---|
| MCCR1 | 50h |

### 55.3.1.14.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0), before switching to high speed mode (with timing configured by MCCR1).

### 55.3.1.14.3 Diagram



#### 55.3.1.14.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. The Data Valid Delay must be configured to be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

### 55.3.1.15 Master FIFO Control Register (MFCR)

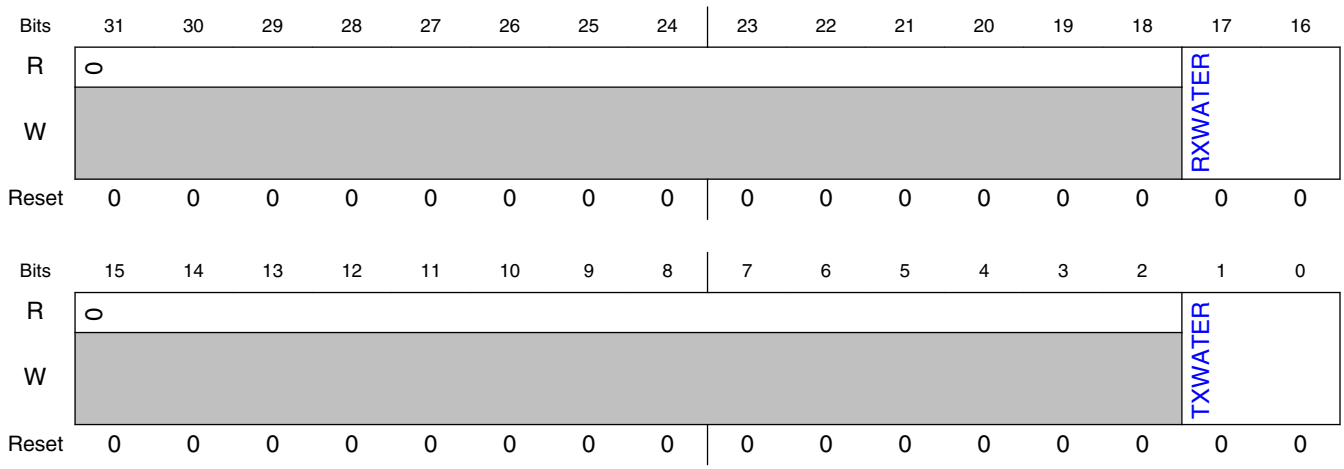
#### 55.3.1.15.1 Offset

| Register | Offset |
|---|---|
| MFCR | 58h |

### 55.3.1.15.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

### 55.3.1.15.3 Diagram




### 55.3.1.15.4 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |
| 15-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |

## 55.3.1.16 Master FIFO Status Register (MFSR)

### 55.3.1.16.1 Offset

| Register | Offset |
|---|---|
| MFSR | 5Ch |

### 55.3.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.16.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words in the receive FIFO. |
| 15-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words in the transmit FIFO. |

## 55.3.1.17 Master Transmit Data Register (MTDR)

### 55.3.1.17.1 Offset

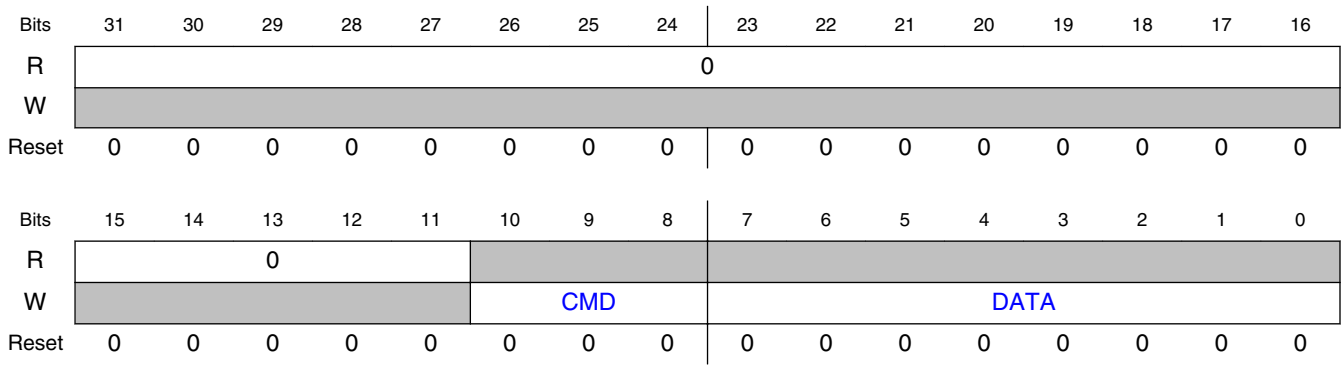
| Register | Offset |
|---|---|
| MTDR | 60h |

### 55.3.1.17.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.

- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit or 32-bit will write both the CMD and DATA fields and increment the FIFO.

### 55.3.1.17.3 Diagram




### 55.3.1.17.4 Fields

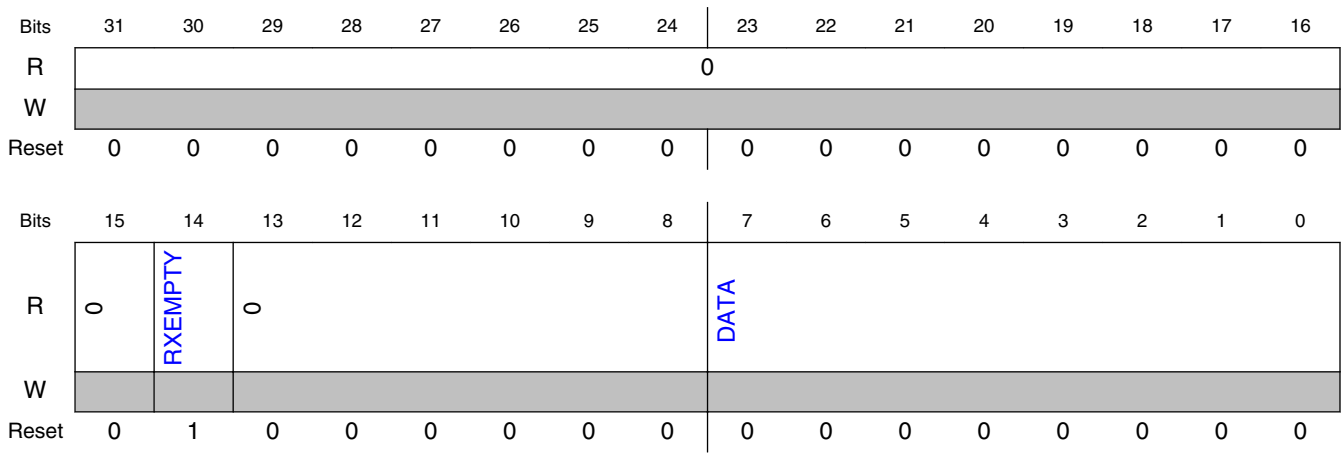
| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>CMD | Command Data<br>000b - Transmit DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes<br>010b - Generate STOP condition<br>011b - Receive and discard (DATA[7:0] + 1) bytes<br>100b - Generate (repeated) START and transmit address in DATA[7:0]<br>101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.<br>110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode<br>111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned. |
| 7-0<br>DATA | Transmit Data<br>Performing an 8-bit write to DATA will zero extend the CMD field. |

## 55.3.1.18 Master Receive Data Register (MRDR)

### 55.3.1.18.1 Offset

| Register | Offset |
|---|---|
| MRDR | 70h |

### 55.3.1.18.2 Diagram




### 55.3.1.18.3 Fields

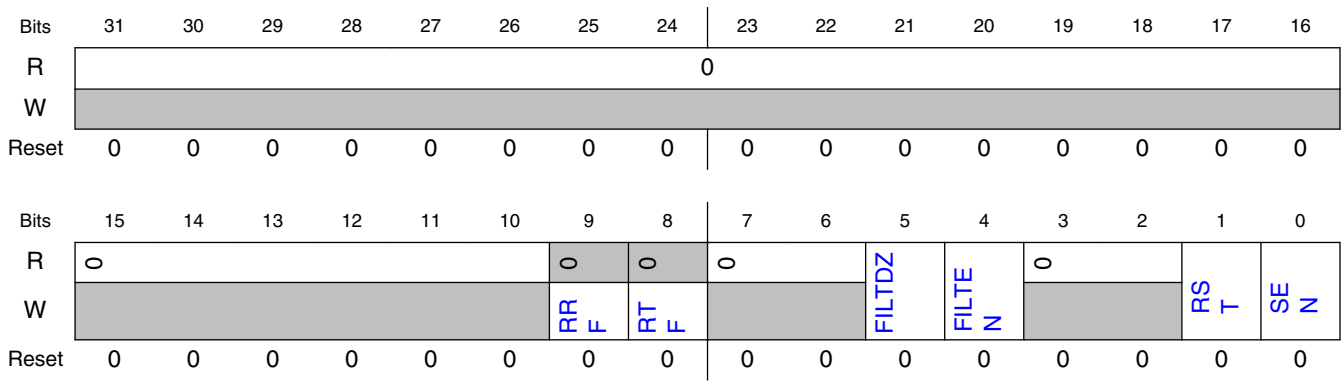
| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14<br>RXEMPTY | RX Empty<br>0b - Receive FIFO is not empty<br>1b - Receive FIFO is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data. |

## 55.3.1.19 Slave Control Register (SCR)

## 55.3.1.19.1 Offset

| Register | Offset |
|---|---|
| SCR | 110h |

## 55.3.1.19.2 Diagram




## 55.3.1.19.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive Data Register is now empty |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit Data Register is now empty |
| 7-6<br>— | Reserved |
| 5<br>FILTDZ | Filter Doze Enable<br>Filter Doze Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Filter remains enabled in Doze mode<br>1b - Filter is disabled in Doze mode |
| 4<br>FILTEN | Filter Enable<br>Filter Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Disable digital filter and output delay counter for slave mode<br>1b - Enable digital filter and output delay counter for slave mode |
| 3-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | 0b - Slave mode logic is not reset<br>1b - Slave mode logic is reset |
| 0<br>SEN | Slave Enable<br>0b - I2C Slave mode is disabled<br>1b - I2C Slave mode is enabled |

## 55.3.1.20 Slave Status Register (SSR)

### 55.3.1.20.1 Offset

| Register | Offset |
|---|---|
| SSR | 114h |

### 55.3.1.20.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | SBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARF | GCF | AM1F | AM0F | FEF | BEF | SDF | RSF | 0 | | | | TAF | AVF | RDF | TDF |
| W | | | | | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.20.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>Indicates if an I2C bus is idle or busy.<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24 | Slave Busy Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SBF | Indicates if an I2C slave is idle or busy.<br>0b - I2C Slave is idle<br>1b - I2C Slave is busy |
| 23-16<br>— | Reserved |
| 15<br>SARF | SMBus Alert Response Flag<br>• SMBus Alert Response Flag is cleared by reading the Address Status Register<br>• SMBus Alert Response Flag cannot generate an asynchronous wakeup<br><br>0b - SMBus Alert Response is disabled or not detected<br>1b - SMBus Alert Response is enabled and detected |
| 14<br>GCF | General Call Flag<br>Indicates whether a slave has detected the General Call Address.<br>• General Call Flag is cleared by reading the Address Status Register<br>• General Call Flag cannot generate an asynchronous wakeup<br><br>0b - Slave has not detected the General Call Address or the General Call Address is disabled<br>1b - Slave has detected the General Call Address |
| 13<br>AM1F | Address Match 1 Flag<br>Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by ADDRCFG.<br>• Address Match 1 Flag is cleared by reading the Address Status Register<br>• Address Match 1 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address<br>1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address |
| 12<br>AM0F | Address Match 0 Flag<br>Indicates that the received address has matched the ADDR0 field as configured by ADDRCFG.<br>• Address Match 0 Flag is cleared by reading the Address Status Register<br>• Address Match 0 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR0 matching address<br>1b - Have received an ADDR0 matching address |
| 11<br>FEF | FIFO Error Flag<br>FIFO error flag can only be set when clock stretching is disabled.<br>0b - FIFO underflow or overflow was not detected<br>1b - FIFO underflow or overflow was detected |
| 10<br>BEF | Bit Error Flag<br>Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore the rest of the transfer until the next (repeated) START condition.<br>0b - Slave has not detected a bit error<br>1b - Slave has detected a bit error |
| 9<br>SDF | STOP Detect Flag<br>STOP Detect Flag will set when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte.<br>0b - Slave has not detected a STOP condition<br>1b - Slave has detected a STOP condition |
| 8<br>RSF | Repeated Start Flag |

*Table continues on the next page...*

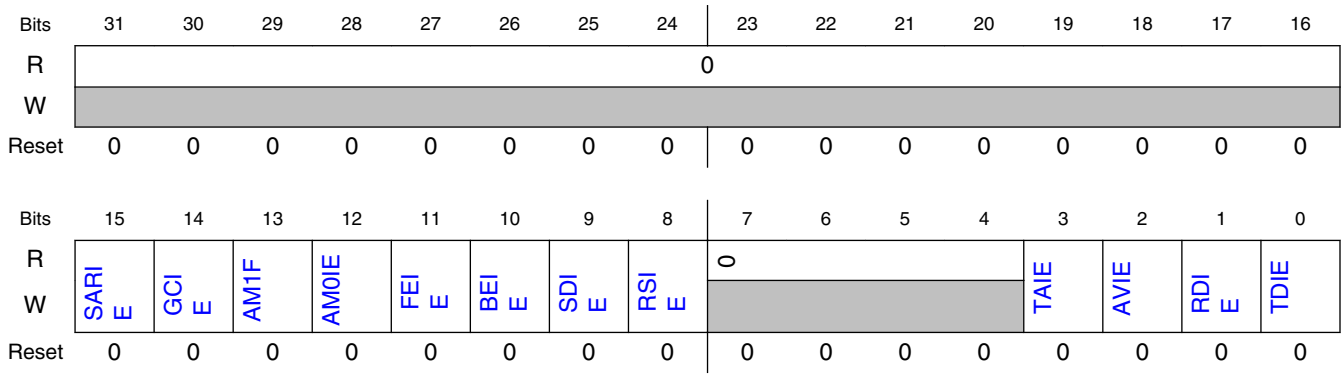
| Field | Function |
|---|---|
| | Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag does not set when the slave first detects a START condition.<br>0b - Slave has not detected a Repeated START condition<br>1b - Slave has detected a Repeated START condition |
| 7-4<br>— | Reserved |
| 3<br>TAF | Transmit ACK Flag<br>Transmit ACK Flag is cleared by writing the Transmit ACK register.<br>0b - Transmit ACK/NACK is not required<br>1b - Transmit ACK/NACK is required |
| 2<br>AVF | Address Valid Flag<br>Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register.<br>0b - Address Status Register is not valid<br>1b - Address Status Register is valid |
| 1<br>RDF | Receive Data Flag<br>Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set.<br>0b - Receive data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared.<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 55.3.1.21 Slave Interrupt Enable Register (SIER)

### 55.3.1.21.1 Offset

| Register | Offset |
|---|---|
| SIER | 118h |

## 55.3.1.21.2 Diagram




## 55.3.1.21.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SARIE | SMBus Alert Response Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 14<br>GCIE | General Call Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 13<br>AM1F | Address Match 1 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>AM0IE | Address Match 0 Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>FEIE | FIFO Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>BEIE | Bit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>RSIE | Repeated Start Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-4<br>— | Reserved |
| 3<br>TAIE | Transmit ACK Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>AVIE | Address Valid Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.1.22 Slave DMA Enable Register (SDER)

### 55.3.1.22.1 Offset

| Register | Offset |
|---|---|
| SDER | 11Ch |

### 55.3.1.22.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | AVDE | RDDE | TDDE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.22.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>AVDE | Address Valid DMA Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set Receive Data Configuration (SCFGR1[RXCFG]), to allow the DMA to read the address from the Receive Data Register.<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.1.23 Slave Configuration Register 1 (SCFGR1)

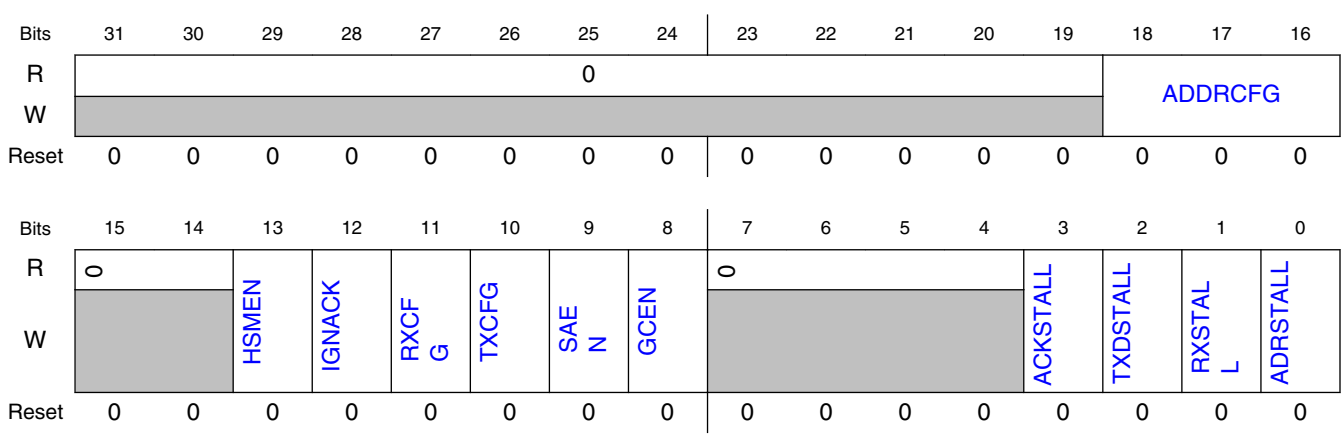
### 55.3.1.23.1 Offset

| Register | Offset |
|---|---|
| SCFGR1 | 124h |

### 55.3.1.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

### 55.3.1.23.3 Diagram



### 55.3.1.23.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>ADDRCFG | Address Configuration<br>Configures the condition that will cause an address to match.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or Address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or Address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or Address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or Address match 1 (7-bit)<br>110b - From Address match 0 (7-bit) to Address match 1 (7-bit)<br>111b - From Address match 0 (10-bit) to Address match 1 (10-bit) |
| 15-14<br>— | Reserved |
| 13<br>HSMEN | High Speed Mode Enable<br>Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, the FILTEN and ACKSTALL bits are ignored until the next STOP condition is detected.<br>0b - Disables detection of HS-mode master code<br>1b - Enables detection of HS-mode master code |
| 12<br>IGNACK | Ignore NACK<br>When Ignore NACK is set, the LPI2C slave will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode.<br>0b - Slave will end transfer when NACK is detected<br>1b - Slave will not end transfer when NACK detected |
| 11<br>RXCFG | Receive Data Configuration<br>0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]).<br>1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]). |
| 10<br>TXCFG | Transmit Flag Configuration<br>The transmit data flag will always assert before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO.<br>• When TXCFG=0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer.<br>• When TXCFG=1, the Transmit Data flag will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.<br>0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty<br>1b - Transmit Data Flag will assert whenever the Transmit Data register is empty |
| 9 | SMBus Alert Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SAEN | 0b - Disables match on SMBus Alert<br>1b - Enables match on SMBus Alert |
| 8<br>GCEN | General Call Enable<br>0b - General Call address is disabled<br>1b - General Call address is enabled |
| 7-4<br>— | Reserved |
| 3<br>ACKSTALL | ACK SCL Stall<br>Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Transmit ACK Register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode.<br>When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]).<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 2<br>TXDSTALL | TX Data SCL Stall<br>Enables SCL clock stretching when the Transmit Data flag (SSR[TDF]) is set during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 1<br>RXSTALL | RX SCL Stall<br>Enables SCL clock stretching when the Receive Data flag (SSR[RDF]) is set during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 0<br>ADRSTALL | Address SCL Stall<br>Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |

## 55.3.1.24 Slave Configuration Register 2 (SCFGR2)

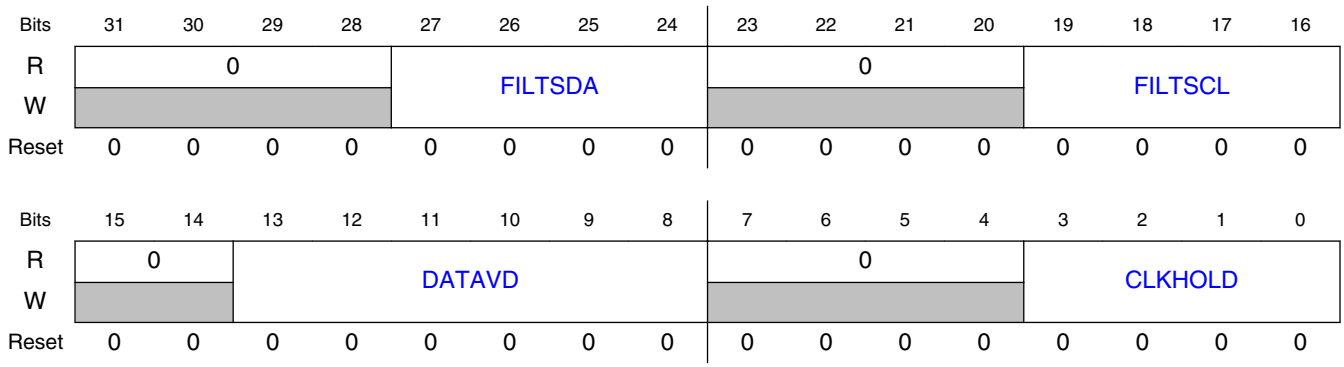
### 55.3.1.24.1 Offset

| Register | Offset |
|---|---|
| SCFGR2 | 128h |

### 55.3.1.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

### 55.3.1.24.3 Diagram




### 55.3.1.24.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C slave digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C slave digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 15-14<br>— | Reserved |
| 13-8<br>DATAVD | Data Valid Delay<br>Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles.<br>• The data valid delay must be configured to be less than the minimum SCL low period<br>• The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode |
| 7-4<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>CLKHOLD | Clock Hold Time<br>Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled.<br>• The minimum hold time is equal to CLKHOLD+3 cycles<br>• The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode |

## 55.3.1.25 Slave Address Match Register (SAMR)

### 55.3.1.25.1 Offset

| Register | Offset |
|---|---|
| SAMR | 140h |

### 55.3.1.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

### 55.3.1.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR1 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR0 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.25.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-17<br>ADDR1 | Address 1 Value<br>Compared against the received address to detect the Slave Address. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17]<br>• In 7-bit mode, the address is compared to ADDR1[23:17] |
| 16-11<br>— | Reserved |
| 10-1<br>ADDR0 | Address 0 Value<br>Compared against the received address to detect the Slave Address.<br>• In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1]<br>• In 7-bit mode, the address is compared to ADDR0[7:1]<br>• |
| 0<br>— | Reserved |

## 55.3.1.26 Slave Address Status Register (SASR)

### 55.3.1.26.1 Offset

| Register | Offset |
|---|---|
| SASR | 150h |

### 55.3.1.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ANV | 0 | | | RADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.26.3 Fields

| Field | Function |
|---|---|
| 31-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>ANV | Address Not Valid<br>0b - Received Address (RADDR) is valid<br>1b - Received Address (RADDR) is not valid |
| 13-11<br>— | Reserved |
| 10-0<br>RADDR | Received Address<br>The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register.<br>• In 7-bit mode, the address byte is store in RADDR[7:0]<br>• In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0] |

## 55.3.1.27 Slave Transmit ACK Register (STAR)

### 55.3.1.27.1 Offset

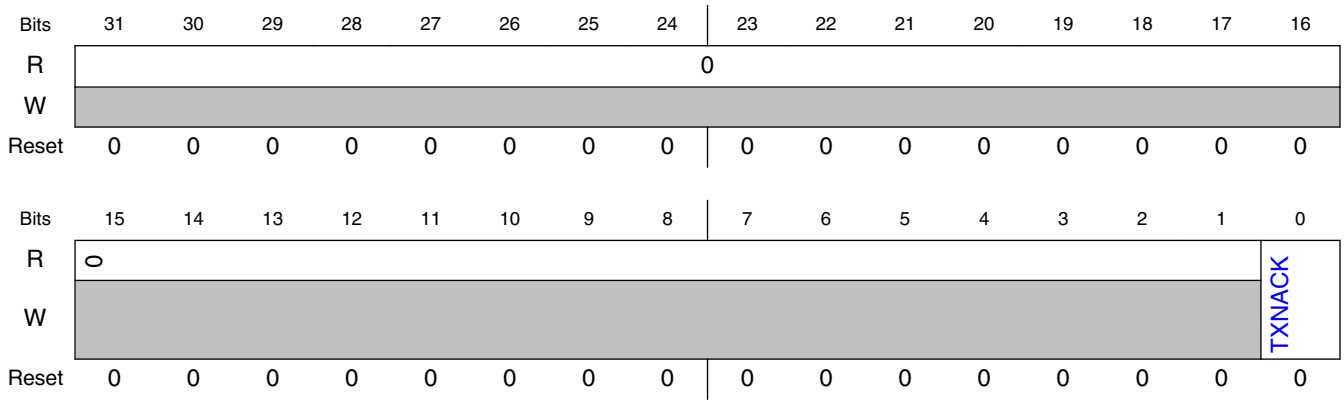
| Register | Offset |
|---|---|
| STAR | 154h |

### 55.3.1.27.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR1[ACKSTALL].

- The ACKSTALL bit will enable clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).

### 55.3.1.27.3 Diagram




### 55.3.1.27.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TXNACK | Transmit NACK<br>After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK.<br>• When ACKSTALL is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that will stall the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK).<br>• To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle.<br><br>0b - Write a Transmit ACK for each received word<br>1b - Write a Transmit NACK for each received word |

## 55.3.1.28 Slave Transmit Data Register (STDR)

### 55.3.1.28.1 Offset

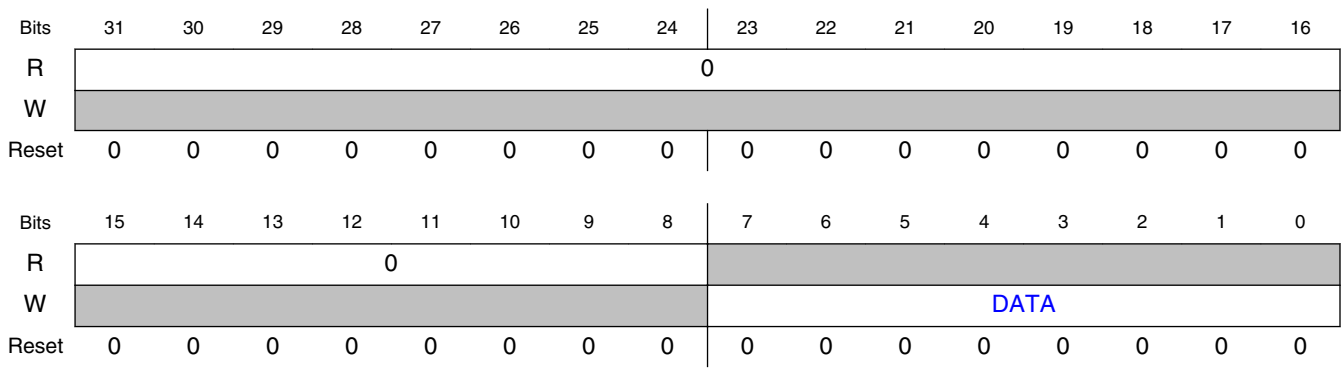
| Register | Offset |
|---|---|
| STDR | 160h |

### 55.3.1.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

### 55.3.1.28.3 Diagram




### 55.3.1.28.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>DATA | Transmit Data<br>Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register |

## 55.3.1.29 Slave Receive Data Register (SRDR)

### 55.3.1.29.1 Offset

| Register | Offset |
|---|---|
| SRDR | 170h |

### 55.3.1.29.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SOF | RXEMPTY | 0 | | | | | | DATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.1.29.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SOF | Start Of Frame<br>0b - Indicates this is not the first data word since a (repeated) START or STOP condition<br>1b - Indicates this is the first data word since a (repeated) START or STOP condition |
| 14<br>RXEMPTY | RX Empty<br>0b - The Receive Data Register is not empty<br>1b - The Receive Data Register is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Slave Receive Data Register returns the data received by the I2C slave |

## 55.3.2 LPI2C1 register descriptions

### 55.3.2.1 LPI2C1 Memory map

LPI2C1 base address: 4003_B000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Master Control Register (MCR) | 32 | RW | 0000_0000h |
| 14h | Master Status Register (MSR) | 32 | W1C | 0000_0001h |
| 18h | Master Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |
| 1Ch | Master DMA Enable Register (MDER) | 32 | RW | 0000_0000h |
| 20h | Master Configuration Register 0 (MCFGR0) | 32 | RW | 0000_0000h |
| 24h | Master Configuration Register 1 (MCFGR1) | 32 | RW | 0000_0000h |
| 28h | Master Configuration Register 2 (MCFGR2) | 32 | RW | 0000_0000h |
| 2Ch | Master Configuration Register 3 (MCFGR3) | 32 | RW | 0000_0000h |
| 40h | Master Data Match Register (MDMR) | 32 | RW | 0000_0000h |
| 48h | Master Clock Configuration Register 0 (MCCR0) | 32 | RW | 0000_0000h |
| 50h | Master Clock Configuration Register 1 (MCCR1) | 32 | RW | 0000_0000h |
| 58h | Master FIFO Control Register (MFCR) | 32 | RW | 0000_0000h |
| 5Ch | Master FIFO Status Register (MFSR) | 32 | RO | 0000_0000h |
| 60h | Master Transmit Data Register (MTDR) | 32 | WO | 0000_0000h |
| 70h | Master Receive Data Register (MRDR) | 32 | RO | 0000_4000h |
| 110h | Slave Control Register (SCR) | 32 | RW | 0000_0000h |
| 114h | Slave Status Register (SSR) | 32 | W1C | 0000_0000h |
| 118h | Slave Interrupt Enable Register (SIER) | 32 | RW | 0000_0000h |
| 11Ch | Slave DMA Enable Register (SDER) | 32 | RW | 0000_0000h |
| 124h | Slave Configuration Register 1 (SCFGR1) | 32 | RW | 0000_0000h |
| 128h | Slave Configuration Register 2 (SCFGR2) | 32 | RW | 0000_0000h |
| 140h | Slave Address Match Register (SAMR) | 32 | RW | 0000_0000h |
| 150h | Slave Address Status Register (SASR) | 32 | RO | 0000_4000h |
| 154h | Slave Transmit ACK Register (STAR) | 32 | RW | 0000_0000h |
| 160h | Slave Transmit Data Register (STDR) | 32 | WO | 0000_0000h |
| 170h | Slave Receive Data Register (SRDR) | 32 | RO | 0000_4000h |

### 55.3.2.2 Version ID Register (VERID)

#### 55.3.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 55.3.2.2.2 Function

.

#### 55.3.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 55.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | Returns the major version number for the module design specification. Read-only field. |
| 23-16 | Minor Version Number |
| MINOR | Returns the minor version number for the module design specification. Read-only field. |
| 15-0 | Feature Specification Number |
| FEATURE | Returns the feature set number. Read-only field.<br>0000000000000010b - Master only, with standard feature set<br>0000000000000011b - Master and slave, with standard feature set |

### 55.3.2.3 Parameter Register (PARAM)

#### 55.3.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 55.3.2.3.2 Function

.

#### 55.3.2.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MRXFIFO | | | | 0 | | | | MTXFIFO | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 55.3.2.3.4 Fields

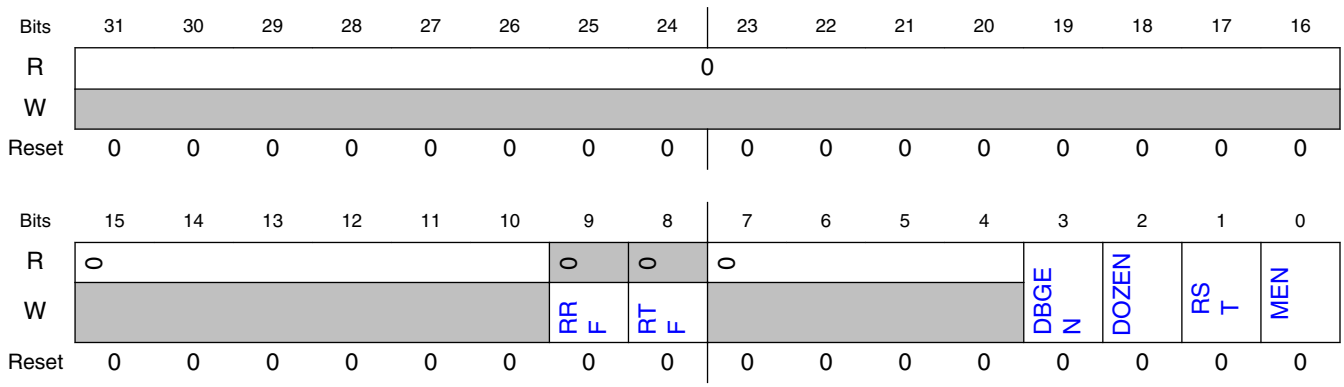
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>MRXFIFO | Master Receive FIFO Size<br>Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$ |
| 7-4<br>— | Reserved |
| 3-0<br>MTXFIFO | Master Transmit FIFO Size<br>Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$ |

### 55.3.2.4 Master Control Register (MCR)

### 55.3.2.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 10h |

### 55.3.2.4.2 Diagram




### 55.3.2.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Master is disabled in debug mode<br>1b - Master is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the master in Doze mode<br>0b - Master is enabled in Doze mode<br>1b - Master is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>MEN | Master Enable<br>0b - Master logic is disabled<br>1b - Master logic is enabled |

## 55.3.2.5 Master Status Register (MSR)

### 55.3.2.5.1 Offset

| Register | Offset |
|---|---|
| MSR | 14h |

### 55.3.2.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 55.3.2.5.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24<br>MBF | Master Busy Flag<br>0b - I2C Master is idle<br>1b - I2C Master is busy |
| 23-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set.<br>0b - Have not received matching data<br>1b - Have received matching data |
| 13<br>PLTF | Pin Low Timeout Flag<br>Will set when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR3[PINLOW]), even when the LPI2C master is idle.<br>• Software is responsible for resolving the pin low condition.<br>• Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues.<br>• Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared.<br>0b - Pin low timeout has not occurred or is disabled<br>1b - Pin low timeout has occurred |
| 12<br>FEF | FIFO Error Flag<br>Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared.<br>0b - No error<br>1b - Master sending or receiving data without a START condition |
| 11<br>ALF | Arbitration Lost Flag<br>Set:<br>• if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus<br>• or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data<br>When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag has been cleared.<br>0b - Master has not lost arbitration<br>1b - Master has lost arbitration |
| 10<br>NDF | NACK Detect Flag<br>Set if the LPI2C master detects a NACK when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated.<br>0b - Unexpected NACK was not detected<br>1b - Unexpected NACK was detected |
| 9<br>SDF | STOP Detect Flag<br>Set when the LPI2C master generates a STOP condition.<br>0b - Master has not generated a STOP condition<br>1b - Master has generated a STOP condition |
| 8<br>EPF | End Packet Flag<br>Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition.<br>0b - Master has not generated a STOP or Repeated START condition<br>1b - Master has generated a STOP or Repeated START condition |
| 7-2 | Reserved |

*Table continues on the next page...*

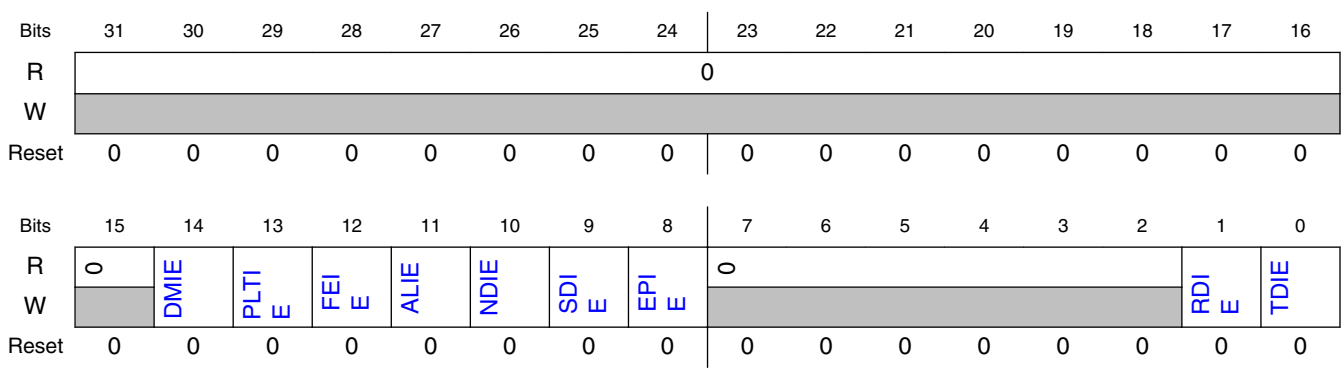
| Field | Function |
|---|---|
| — | |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER.<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER.<br>0b - Transmit data is not requested<br>1b - Transmit data is requested |

## 55.3.2.6 Master Interrupt Enable Register (MIER)

### 55.3.2.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 18h |

### 55.3.2.6.2 Diagram




### 55.3.2.6.3 Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14 | Data Match Interrupt Enable |

*Table continues on the next page...*

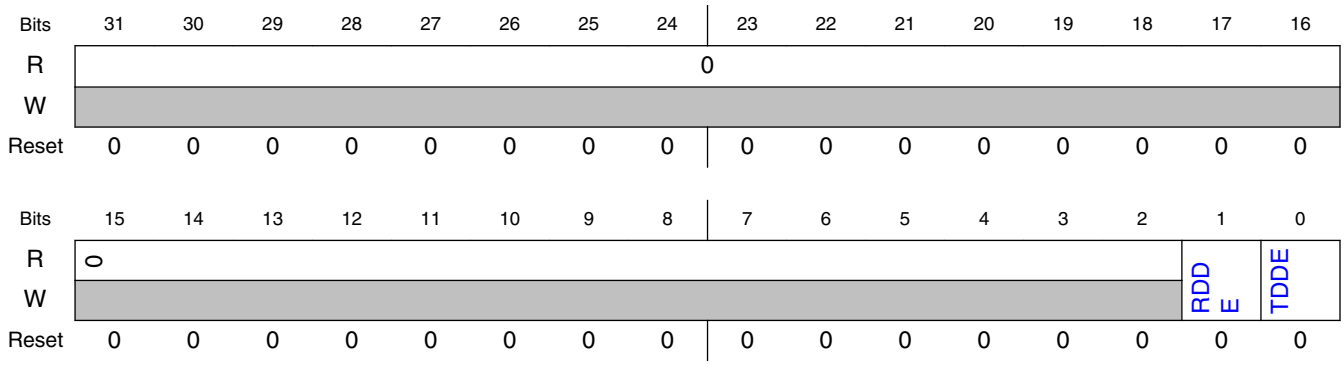
| Field | Function |
|---|---|
| DMIE | 0b - Disabled<br>1b - Enabled |
| 13<br>PLTIE | Pin Low Timeout Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>FEIE | FIFO Error Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>ALIE | Arbitration Lost Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>NDIE | NACK Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>EPIE | End Packet Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.2.7 Master DMA Enable Register (MDER)

### 55.3.2.7.1 Offset

| Register | Offset |
|---|---|
| MDER | 1Ch |

### 55.3.2.7.2 Diagram




### 55.3.2.7.3 Fields

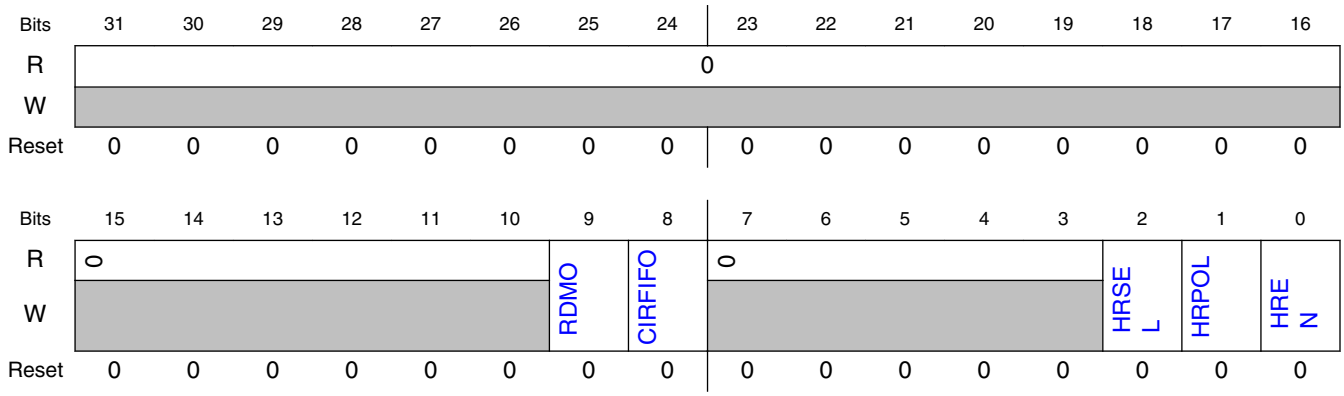
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.2.8 Master Configuration Register 0 (MCFGR0)

### 55.3.2.8.1 Offset

| Register | Offset |
|---|---|
| MCFGR0 | 20h |

### 55.3.2.8.2 Diagram




### 55.3.2.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change).<br>0b - Host request input is pin HREQ<br>1b - Host request input is input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field should remain static (Host Request Polarity should not change).<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable |

| Field | Function |
|---|---|
| | When enabled, the LPI2C master will only initiate a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request.<br>0b - Host request input is disabled<br>1b - Host request input is enabled |

## 55.3.2.9 Master Configuration Register 1 (MCFGR1)

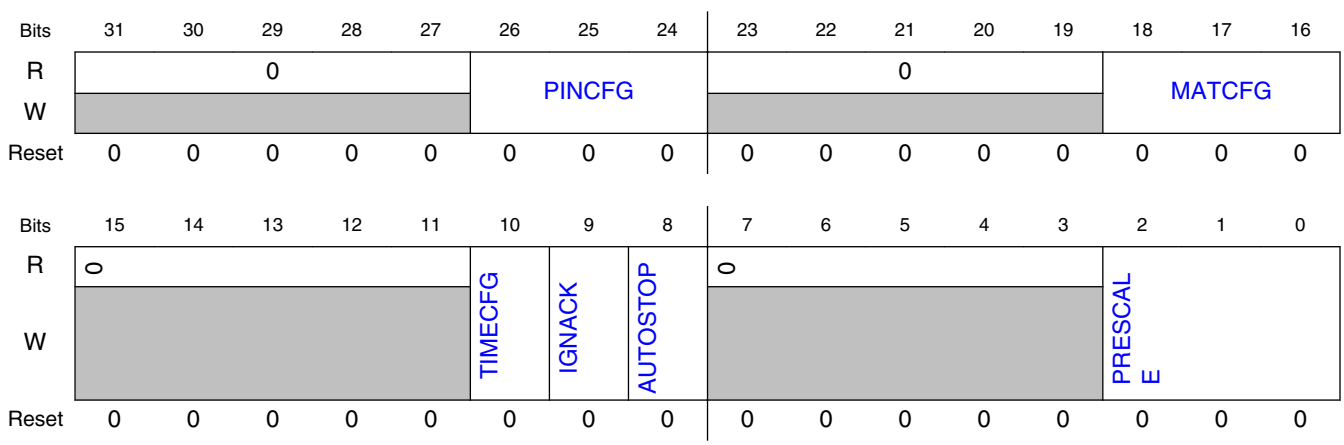
### 55.3.2.9.1 Offset

| Register | Offset |
|---|---|
| MCFGR1 | 24h |

### 55.3.2.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

### 55.3.2.9.3 Diagram




### 55.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>PINCFG | Pin Configuration |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Configures the pin mode for LPI2C.<br>Table 55-4. 2-pin / 4-pin pin configurations for masters and slaves
<table>
<tr><th>PINCFG</th><th>SCL / SDA pins</th><th>SCLS / SDAS pins</th></tr>
<tr><td>000</td><td>Bi-directional open drain for master and slave</td><td>Not used</td></tr>
<tr><td>001</td><td>Output-only (ultra-fast mode) open drain for master and slave</td><td>Not used</td></tr>
<tr><td>010</td><td>Bi-directional push-pull for master and slave</td><td>Not used</td></tr>
<tr><td>011</td><td>Input only for master and slave</td><td>Output-only push-pull for master and slave</td></tr>
<tr><td>100</td><td>Bi-directional open drain for master</td><td>Bi-directional open drain for slave</td></tr>
<tr><td>101</td><td>Output-only (ultra-fast mode) open drain for master</td><td>Output-only open drain for slave</td></tr>
<tr><td>110</td><td>Bi-directional push-pull for master</td><td>Bi-directional push-pull for slave</td></tr>
<tr><td>111</td><td>Input only for master and slave</td><td>Inverted output-only push-pull for master and slave</td></tr>
</table>
000b - 2-pin open drain mode<br>001b - 2-pin output only mode (ultra-fast mode)<br>010b - 2-pin push-pull mode<br>011b - 4-pin push-pull mode<br>100b - 2-pin open drain mode with separate LPI2C slave<br>101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave<br>110b - 2-pin push-pull mode with separate LPI2C slave<br>111b - 4-pin push-pull mode (inverted outputs)</td></tr>
<tr><td>23-19<br>—</td><td>Reserved</td></tr>
<tr><td>18-16<br>MATCFG</td><td>Match Configuration<br>Configures the condition that will cause the DMF to set.<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match is enabled (1st data word equals MATCH0 OR MATCH1)<br>011b - Match is enabled (any data word equals MATCH0 OR MATCH1)<br>100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1)<br>101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1)<br>110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1)<br>111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1)</td></tr>
<tr><td>15-11<br>—</td><td>Reserved</td></tr>
<tr><td>10<br>TIMECFG</td><td>Timeout Configuration<br>0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout<br>1b - Pin Low Timeout Flag will set if either SCL or SDA is low for longer than the configured timeout</td></tr>
<tr><td>9<br>IGNACK</td><td>IGNACK<br>When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode.<br>0b - LPI2C Master will receive ACK and NACK normally<br>1b - LPI2C Master will treat a received NACK as if it (NACK) was an ACK</td></tr>
</table>

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>AUTOSTOP | Automatic STOP Generation<br>When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command.<br>0b - No effect<br>1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy |
| 7-3<br>— | Reserved |
| 2-0<br>PRESCALE | Prescaler<br>Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 55.3.2.10 Master Configuration Register 2 (MCFGR2)

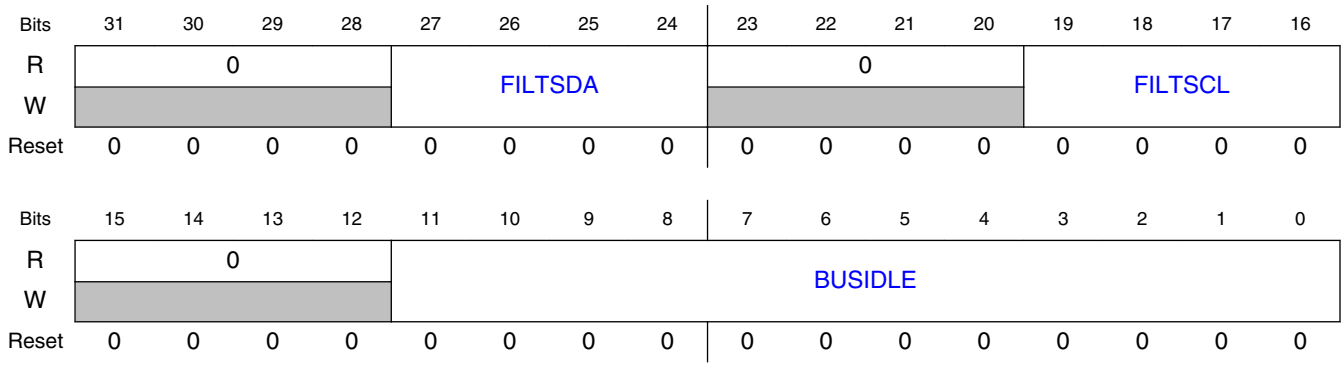
### 55.3.2.10.1 Offset

| Register | Offset |
|---|---|
| MCFGR2 | 28h |

### 55.3.2.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

### 55.3.2.10.3 Diagram




### 55.3.2.10.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C master digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C master digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 15-12<br>— | Reserved |
| 11-0<br>BUSIDLE | Bus Idle Timeout<br>Configures the bus idle timeout period in clock cycles.<br>• If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition<br>• When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled |

## 55.3.2.11 Master Configuration Register 3 (MCFGR3)

### 55.3.2.11.1 Offset

| Register | Offset |
|---|---|
| MCFGR3 | 2Ch |

### 55.3.2.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

### 55.3.2.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PINLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINLOW | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.11.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-8<br>PINLOW | Pin Low Timeout<br>Configures the pin low timeout flag in clock cycles.<br>• If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then PLTF is set<br>• When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled |
| 7-0<br>— | Reserved |

## 55.3.2.12 Master Data Match Register (MDMR)

### 55.3.2.12.1 Offset

| Register | Offset |
|---|---|
| MDMR | 40h |

### 55.3.2.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

### 55.3.2.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>MATCH1 | Match 1 Value<br>Compared against the received data when receive data match is enabled. |
| 15-8<br>— | Reserved |
| 7-0<br>MATCH0 | Match 0 Value<br>Compared against the received data when receive data match is enabled. |

## 55.3.2.13 Master Clock Configuration Register 0 (MCCR0)

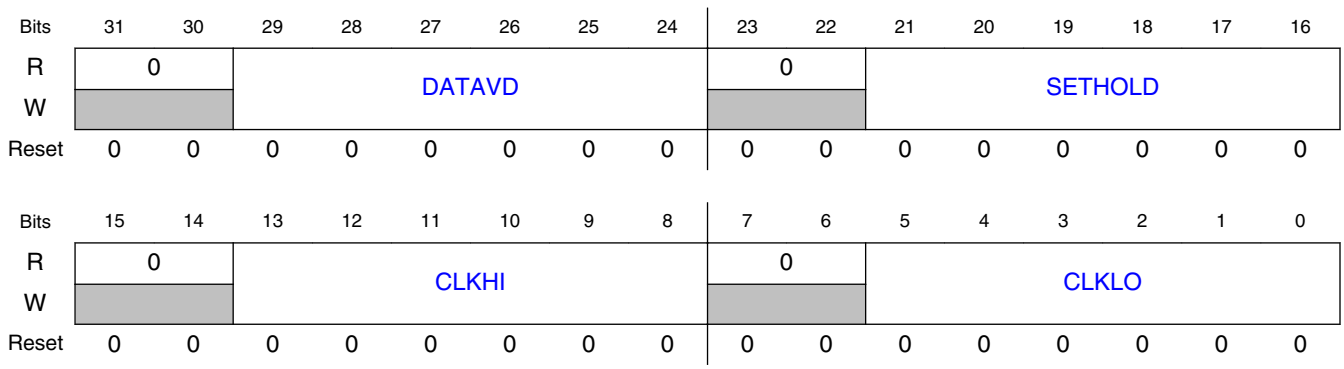
### 55.3.2.13.1 Offset

| Register | Offset |
|---|---|
| MCCR0 | 48h |

### 55.3.2.13.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

### 55.3.2.13.3 Diagram




### 55.3.2.13.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br><br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.2.14 Master Clock Configuration Register 1 (MCCR1)

### 55.3.2.14.1 Offset

| Register | Offset |
|---|---|
| MCCR1 | 50h |

### 55.3.2.14.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0), before switching to high speed mode (with timing configured by MCCR1).

### 55.3.2.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DATAVD | | | | | | 0 | | SETHOLD | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | CLKHI | | | | | | 0 | | CLKLO | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.14.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. The Data Valid Delay must be configured to be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.2.15 Master FIFO Control Register (MFCR)

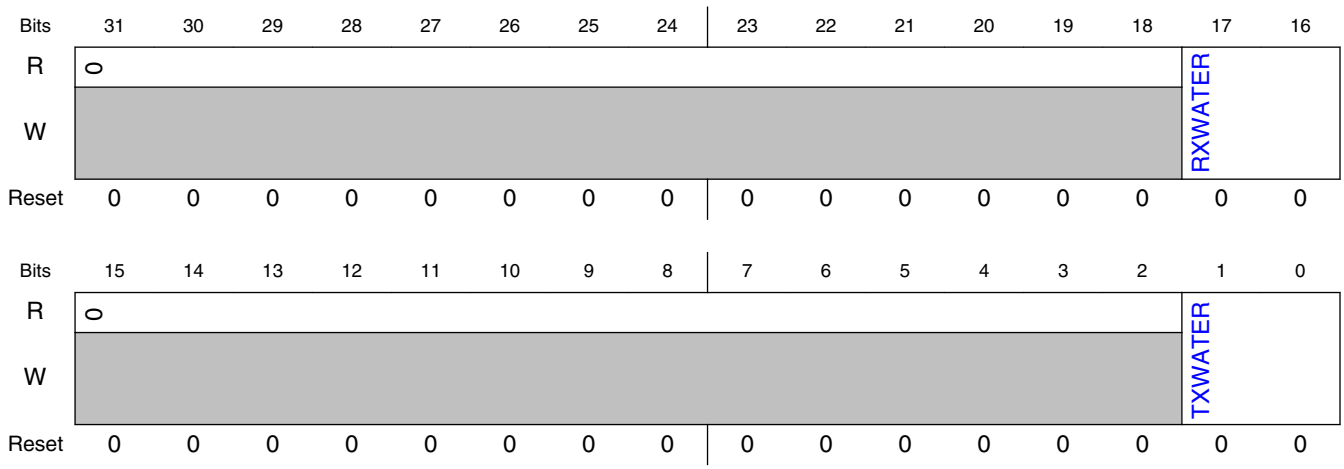
### 55.3.2.15.1 Offset

| Register | Offset |
|---|---|
| MFCR | 58h |

### 55.3.2.15.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

### 55.3.2.15.3 Diagram




### 55.3.2.15.4 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |
| 15-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |

## 55.3.2.16 Master FIFO Status Register (MFSR)

### 55.3.2.16.1 Offset

| Register | Offset |
|---|---|
| MFSR | 5Ch |

### 55.3.2.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.16.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words in the receive FIFO. |
| 15-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words in the transmit FIFO. |

## 55.3.2.17 Master Transmit Data Register (MTDR)

### 55.3.2.17.1 Offset

| Register | Offset |
|---|---|
| MTDR | 60h |

### 55.3.2.17.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.

- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit or 32-bit will write both the CMD and DATA fields and increment the FIFO.

### 55.3.2.17.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | CMD | | | DATA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.17.4 Fields

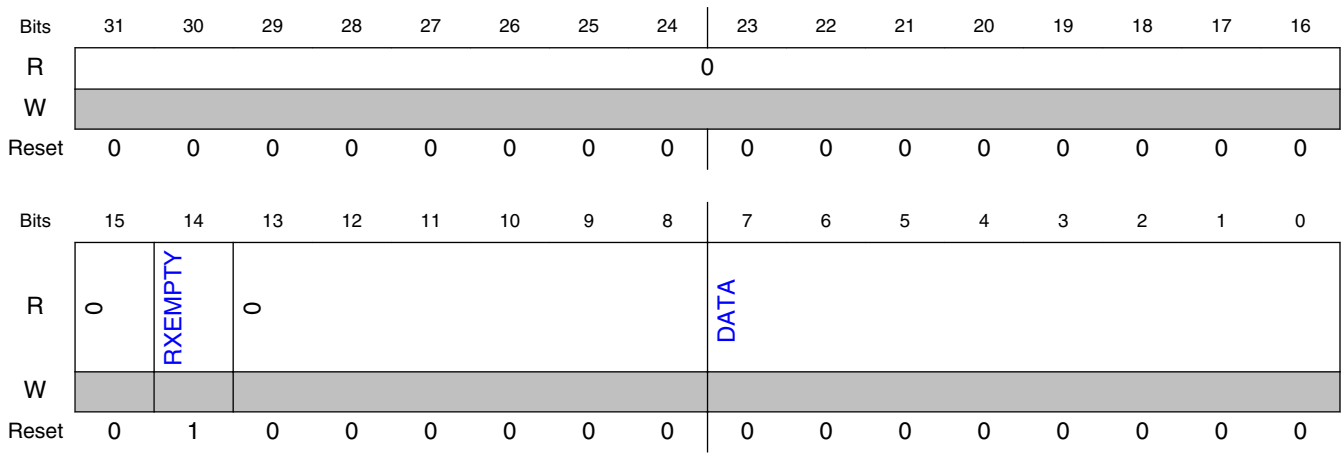
| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>CMD | Command Data<br>000b - Transmit DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes<br>010b - Generate STOP condition<br>011b - Receive and discard (DATA[7:0] + 1) bytes<br>100b - Generate (repeated) START and transmit address in DATA[7:0]<br>101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.<br>110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode<br>111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned. |
| 7-0<br>DATA | Transmit Data<br>Performing an 8-bit write to DATA will zero extend the CMD field. |

## 55.3.2.18 Master Receive Data Register (MRDR)

### 55.3.2.18.1 Offset

| Register | Offset |
|---|---|
| MRDR | 70h |

### 55.3.2.18.2 Diagram




### 55.3.2.18.3 Fields

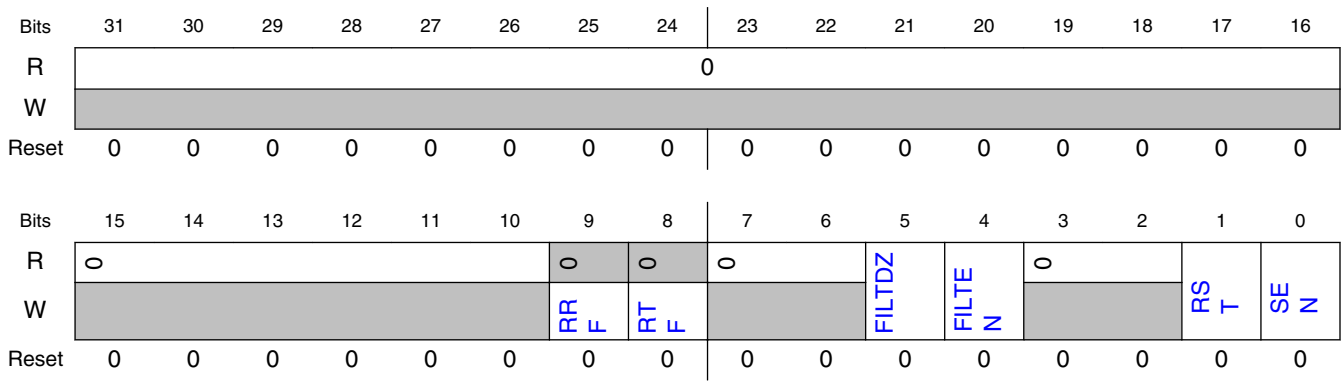
| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14<br>RXEMPTY | RX Empty<br>0b - Receive FIFO is not empty<br>1b - Receive FIFO is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data. |

## 55.3.2.19 Slave Control Register (SCR)

### 55.3.2.19.1 Offset

| Register | Offset |
|---|---|
| SCR | 110h |

### 55.3.2.19.2 Diagram




### 55.3.2.19.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive Data Register is now empty |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit Data Register is now empty |
| 7-6<br>— | Reserved |
| 5<br>FILTDZ | Filter Doze Enable<br>Filter Doze Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Filter remains enabled in Doze mode<br>1b - Filter is disabled in Doze mode |
| 4<br>FILTEN | Filter Enable<br>Filter Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Disable digital filter and output delay counter for slave mode<br>1b - Enable digital filter and output delay counter for slave mode |
| 3-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | 0b - Slave mode logic is not reset<br>1b - Slave mode logic is reset |
| 0<br>SEN | Slave Enable<br>0b - I2C Slave mode is disabled<br>1b - I2C Slave mode is enabled |

## 55.3.2.20 Slave Status Register (SSR)

### 55.3.2.20.1 Offset

| Register | Offset |
|---|---|
| SSR | 114h |

### 55.3.2.20.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | SBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARF | GCF | AM1F | AM0F | FEF | BEF | SDF | RSF | 0 | | | | TAF | AVF | RDF | TDF |
| W | | | | | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.20.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>Indicates if an I2C bus is idle or busy.<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24 | Slave Busy Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SBF | Indicates if an I2C slave is idle or busy.<br>0b - I2C Slave is idle<br>1b - I2C Slave is busy |
| 23-16<br>— | Reserved |
| 15<br>SARF | SMBus Alert Response Flag<br>• SMBus Alert Response Flag is cleared by reading the Address Status Register<br>• SMBus Alert Response Flag cannot generate an asynchronous wakeup<br><br>0b - SMBus Alert Response is disabled or not detected<br>1b - SMBus Alert Response is enabled and detected |
| 14<br>GCF | General Call Flag<br>Indicates whether a slave has detected the General Call Address.<br>• General Call Flag is cleared by reading the Address Status Register<br>• General Call Flag cannot generate an asynchronous wakeup<br><br>0b - Slave has not detected the General Call Address or the General Call Address is disabled<br>1b - Slave has detected the General Call Address |
| 13<br>AM1F | Address Match 1 Flag<br>Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by ADDRCFG.<br>• Address Match 1 Flag is cleared by reading the Address Status Register<br>• Address Match 1 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address<br>1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address |
| 12<br>AM0F | Address Match 0 Flag<br>Indicates that the received address has matched the ADDR0 field as configured by ADDRCFG.<br>• Address Match 0 Flag is cleared by reading the Address Status Register<br>• Address Match 0 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR0 matching address<br>1b - Have received an ADDR0 matching address |
| 11<br>FEF | FIFO Error Flag<br>FIFO error flag can only be set when clock stretching is disabled.<br>0b - FIFO underflow or overflow was not detected<br>1b - FIFO underflow or overflow was detected |
| 10<br>BEF | Bit Error Flag<br>Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore the rest of the transfer until the next (repeated) START condition.<br>0b - Slave has not detected a bit error<br>1b - Slave has detected a bit error |
| 9<br>SDF | STOP Detect Flag<br>STOP Detect Flag will set when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte.<br>0b - Slave has not detected a STOP condition<br>1b - Slave has detected a STOP condition |
| 8<br>RSF | Repeated Start Flag |

*Table continues on the next page...*

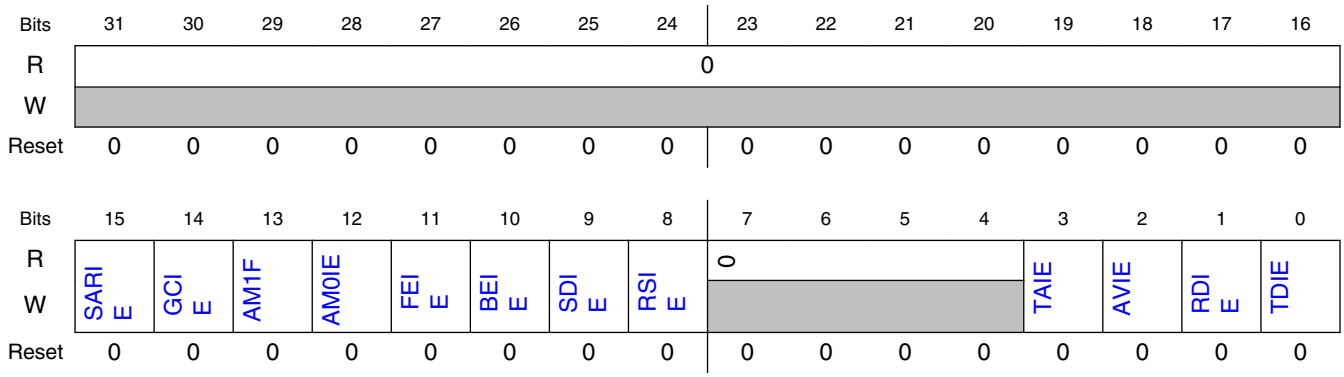
| Field | Function |
|---|---|
| | Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag does not set when the slave first detects a START condition.<br>0b - Slave has not detected a Repeated START condition<br>1b - Slave has detected a Repeated START condition |
| 7-4<br>— | Reserved |
| 3<br>TAF | Transmit ACK Flag<br>Transmit ACK Flag is cleared by writing the Transmit ACK register.<br>0b - Transmit ACK/NACK is not required<br>1b - Transmit ACK/NACK is required |
| 2<br>AVF | Address Valid Flag<br>Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register.<br>0b - Address Status Register is not valid<br>1b - Address Status Register is valid |
| 1<br>RDF | Receive Data Flag<br>Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set.<br>0b - Receive data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared.<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 55.3.2.21 Slave Interrupt Enable Register (SIER)

### 55.3.2.21.1 Offset

| Register | Offset |
|---|---|
| SIER | 118h |

## 55.3.2.21.2 Diagram




## 55.3.2.21.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SARIE | SMBus Alert Response Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 14<br>GCIE | General Call Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 13<br>AM1F | Address Match 1 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>AM0IE | Address Match 0 Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>FEIE | FIFO Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>BEIE | Bit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>RSIE | Repeated Start Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-4<br>— | Reserved |
| 3<br>TAIE | Transmit ACK Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

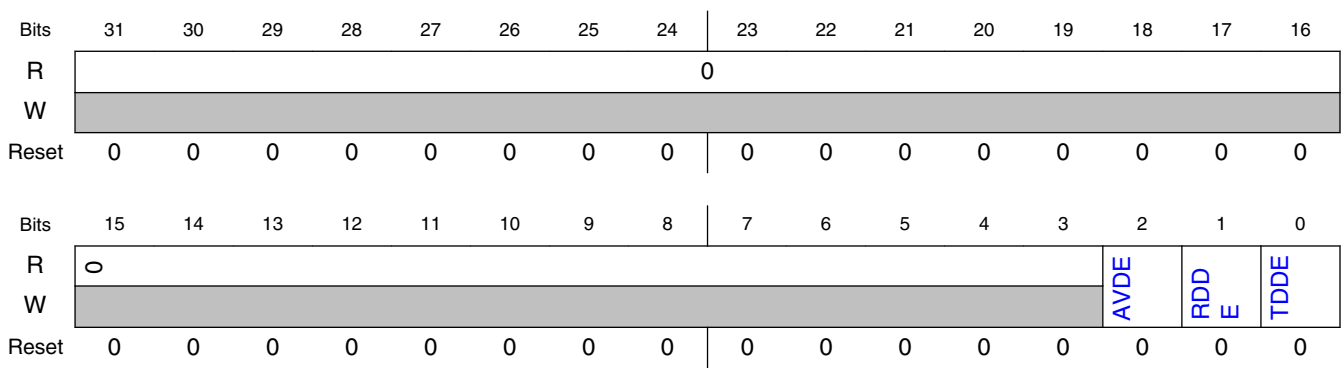
| Field | Function |
|---|---|
| 2<br>AVIE | Address Valid Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.2.22 Slave DMA Enable Register (SDER)

### 55.3.2.22.1 Offset

| Register | Offset |
|---|---|
| SDER | 11Ch |

### 55.3.2.22.2 Diagram




### 55.3.2.22.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>AVDE | Address Valid DMA Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set Receive Data Configuration (SCFGR1[RXCFG]), to allow the DMA to read the address from the Receive Data Register.<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.2.23 Slave Configuration Register 1 (SCFGR1)

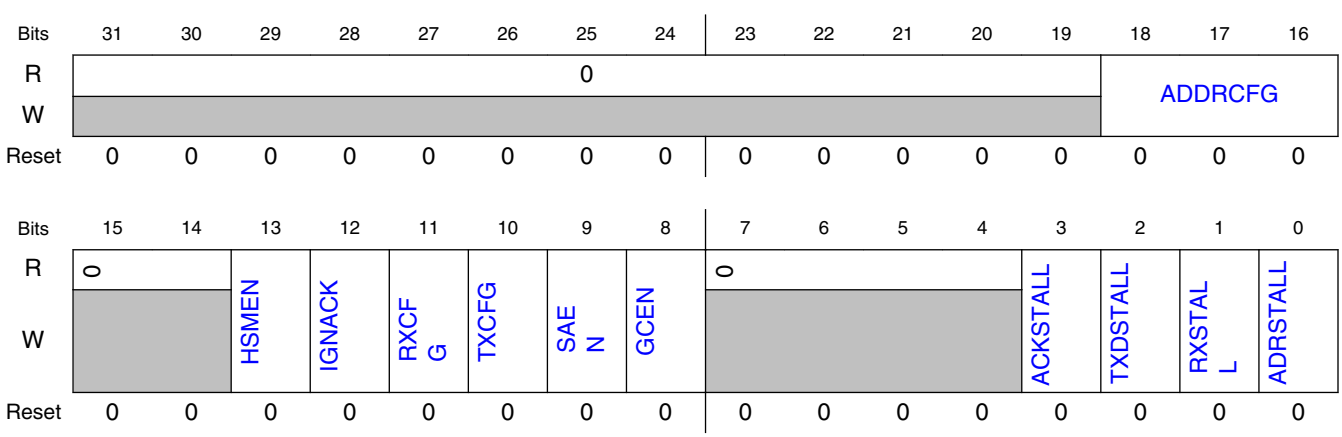
### 55.3.2.23.1 Offset

| Register | Offset |
|---|---|
| SCFGR1 | 124h |

### 55.3.2.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

### 55.3.2.23.3 Diagram



## 55.3.2.23.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>ADDRCFG | Address Configuration<br>Configures the condition that will cause an address to match.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or Address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or Address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or Address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or Address match 1 (7-bit)<br>110b - From Address match 0 (7-bit) to Address match 1 (7-bit)<br>111b - From Address match 0 (10-bit) to Address match 1 (10-bit) |
| 15-14<br>— | Reserved |
| 13<br>HSMEN | High Speed Mode Enable<br>Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, the FILTEN and ACKSTALL bits are ignored until the next STOP condition is detected.<br>0b - Disables detection of HS-mode master code<br>1b - Enables detection of HS-mode master code |
| 12<br>IGNACK | Ignore NACK<br>When Ignore NACK is set, the LPI2C slave will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode.<br>0b - Slave will end transfer when NACK is detected<br>1b - Slave will not end transfer when NACK detected |
| 11<br>RXCFG | Receive Data Configuration<br>0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]).<br>1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]). |
| 10<br>TXCFG | Transmit Flag Configuration<br>The transmit data flag will always assert before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO.<br>• When TXCFG=0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer.<br>• When TXCFG=1, the Transmit Data flag will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.<br>0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty<br>1b - Transmit Data Flag will assert whenever the Transmit Data register is empty |
| 9 | SMBus Alert Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SAEN | 0b - Disables match on SMBus Alert<br>1b - Enables match on SMBus Alert |
| 8<br>GCEN | General Call Enable<br>0b - General Call address is disabled<br>1b - General Call address is enabled |
| 7-4<br>— | Reserved |
| 3<br>ACKSTALL | ACK SCL Stall<br>Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Transmit ACK Register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode.<br>When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]).<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 2<br>TXDSTALL | TX Data SCL Stall<br>Enables SCL clock stretching when the Transmit Data flag (SSR[TDF]) is set during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 1<br>RXSTALL | RX SCL Stall<br>Enables SCL clock stretching when the Receive Data flag (SSR[RDF]) is set during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 0<br>ADRSTALL | Address SCL Stall<br>Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |

## 55.3.2.24 Slave Configuration Register 2 (SCFGR2)

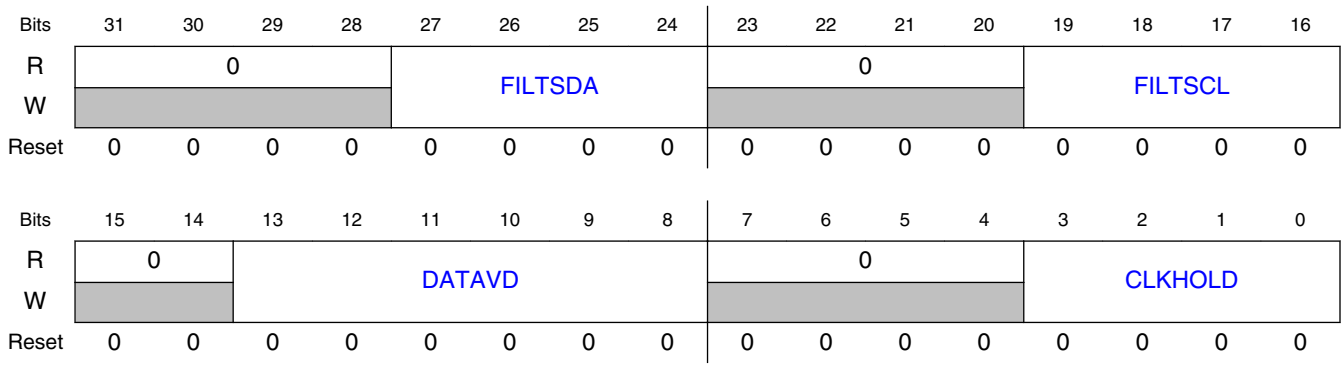
### 55.3.2.24.1 Offset

| Register | Offset |
|---|---|
| SCFGR2 | 128h |

### 55.3.2.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

## 55.3.2.24.3 Diagram




## 55.3.2.24.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C slave digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C slave digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 15-14<br>— | Reserved |
| 13-8<br>DATAVD | Data Valid Delay<br>Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles.<br>• The data valid delay must be configured to be less than the minimum SCL low period<br>• The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode |
| 7-4<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>CLKHOLD | Clock Hold Time<br>Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled.<br>• The minimum hold time is equal to CLKHOLD+3 cycles<br>• The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode |

## 55.3.2.25 Slave Address Match Register (SAMR)

### 55.3.2.25.1 Offset

| Register | Offset |
|---|---|
| SAMR | 140h |

### 55.3.2.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

### 55.3.2.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR1 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR0 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.25.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-17<br>ADDR1 | Address 1 Value<br>Compared against the received address to detect the Slave Address. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17]<br>• In 7-bit mode, the address is compared to ADDR1[23:17] |
| 16-11<br>— | Reserved |
| 10-1<br>ADDR0 | Address 0 Value<br>Compared against the received address to detect the Slave Address.<br>• In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1]<br>• In 7-bit mode, the address is compared to ADDR0[7:1]<br>• |
| 0<br>— | Reserved |

## 55.3.2.26 Slave Address Status Register (SASR)

### 55.3.2.26.1 Offset

| Register | Offset |
|---|---|
| SASR | 150h |

### 55.3.2.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ANV | 0 | | | RADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.2.26.3 Fields

| Field | Function |
|---|---|
| 31-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>ANV | Address Not Valid<br>0b - Received Address (RADDR) is valid<br>1b - Received Address (RADDR) is not valid |
| 13-11<br>— | Reserved |
| 10-0<br>RADDR | Received Address<br>The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register.<br>• In 7-bit mode, the address byte is store in RADDR[7:0]<br>• In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0] |

## 55.3.2.27 Slave Transmit ACK Register (STAR)

### 55.3.2.27.1 Offset

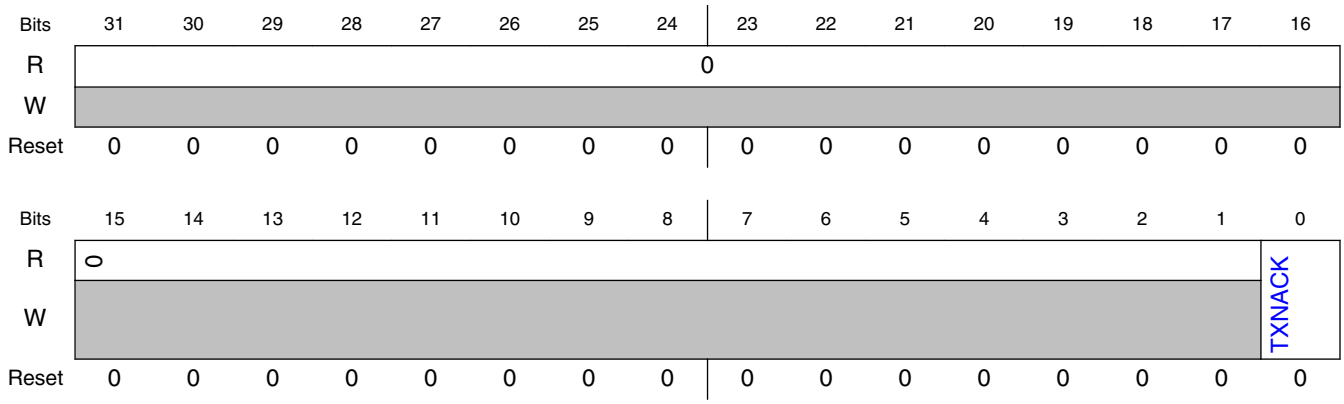
| Register | Offset |
|---|---|
| STAR | 154h |

### 55.3.2.27.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR1[ACKSTALL].

- The ACKSTALL bit will enable clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).

### 55.3.2.27.3 Diagram




### 55.3.2.27.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TXNACK | Transmit NACK<br>After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK.<br>• When ACKSTALL is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that will stall the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK).<br>• To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle.<br><br>0b - Write a Transmit ACK for each received word<br>1b - Write a Transmit NACK for each received word |

## 55.3.2.28 Slave Transmit Data Register (STDR)

### 55.3.2.28.1 Offset

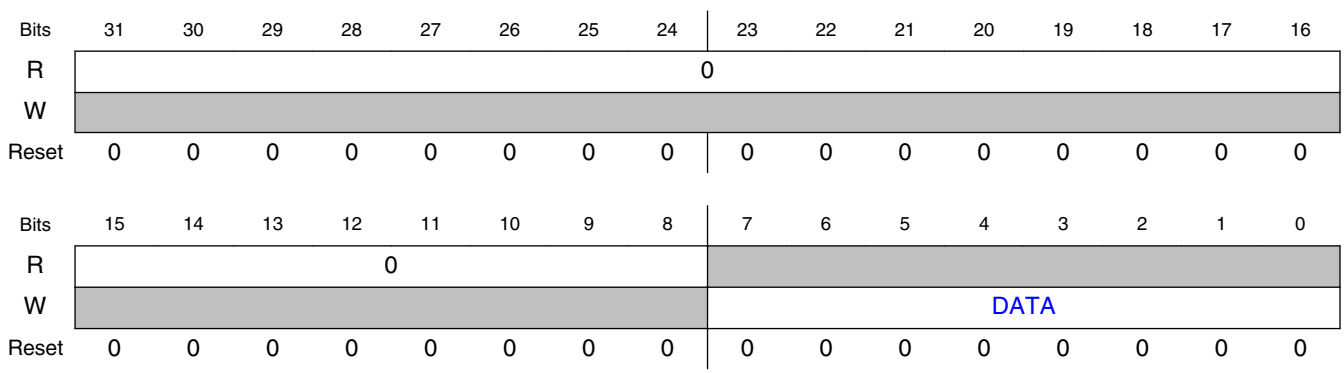
| Register | Offset |
|---|---|
| STDR | 160h |

#### 55.3.2.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

#### 55.3.2.28.3 Diagram




#### 55.3.2.28.4 Fields

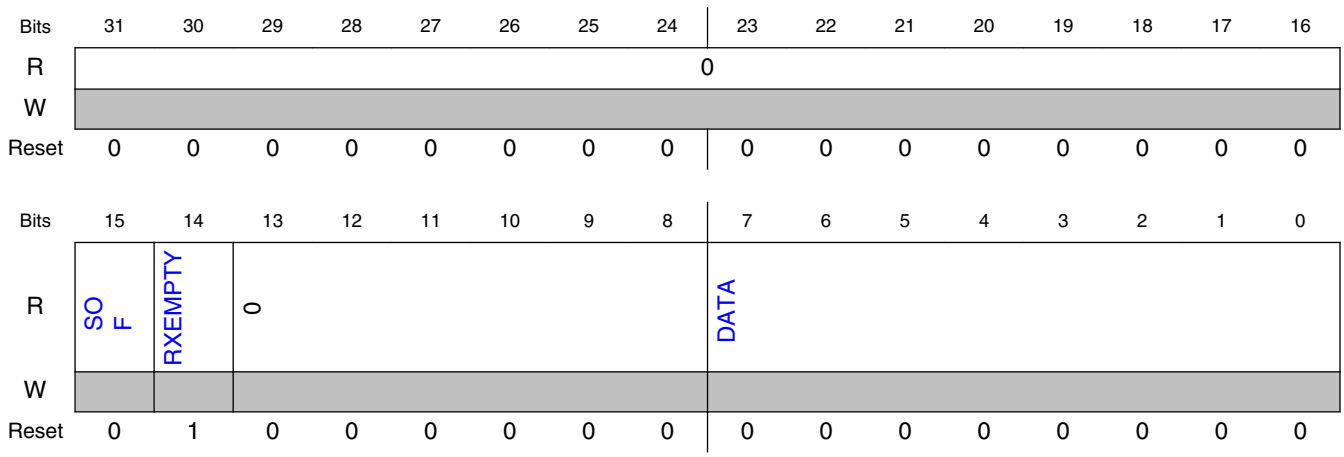
| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>DATA | Transmit Data<br>Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register |

### 55.3.2.29 Slave Receive Data Register (SRDR)

### 55.3.2.29.1 Offset

| Register | Offset |
|---|---|
| SRDR | 170h |

### 55.3.2.29.2 Diagram




### 55.3.2.29.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SOF | Start Of Frame<br>0b - Indicates this is not the first data word since a (repeated) START or STOP condition<br>1b - Indicates this is the first data word since a (repeated) START or STOP condition |
| 14<br>RXEMPTY | RX Empty<br>0b - The Receive Data Register is not empty<br>1b - The Receive Data Register is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Slave Receive Data Register returns the data received by the I2C slave |

## 55.3.3 LPI2C2 register descriptions

### 55.3.3.1 LPI2C2 Memory map

LPI2C2 base address: 4003_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Master Control Register (MCR) | 32 | RW | 0000_0000h |
| 14h | Master Status Register (MSR) | 32 | W1C | 0000_0001h |
| 18h | Master Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |
| 1Ch | Master DMA Enable Register (MDER) | 32 | RW | 0000_0000h |
| 20h | Master Configuration Register 0 (MCFGR0) | 32 | RW | 0000_0000h |
| 24h | Master Configuration Register 1 (MCFGR1) | 32 | RW | 0000_0000h |
| 28h | Master Configuration Register 2 (MCFGR2) | 32 | RW | 0000_0000h |
| 2Ch | Master Configuration Register 3 (MCFGR3) | 32 | RW | 0000_0000h |
| 40h | Master Data Match Register (MDMR) | 32 | RW | 0000_0000h |
| 48h | Master Clock Configuration Register 0 (MCCR0) | 32 | RW | 0000_0000h |
| 50h | Master Clock Configuration Register 1 (MCCR1) | 32 | RW | 0000_0000h |
| 58h | Master FIFO Control Register (MFCR) | 32 | RW | 0000_0000h |
| 5Ch | Master FIFO Status Register (MFSR) | 32 | RO | 0000_0000h |
| 60h | Master Transmit Data Register (MTDR) | 32 | WO | 0000_0000h |
| 70h | Master Receive Data Register (MRDR) | 32 | RO | 0000_4000h |
| 110h | Slave Control Register (SCR) | 32 | RW | 0000_0000h |
| 114h | Slave Status Register (SSR) | 32 | W1C | 0000_0000h |
| 118h | Slave Interrupt Enable Register (SIER) | 32 | RW | 0000_0000h |
| 11Ch | Slave DMA Enable Register (SDER) | 32 | RW | 0000_0000h |
| 124h | Slave Configuration Register 1 (SCFGR1) | 32 | RW | 0000_0000h |
| 128h | Slave Configuration Register 2 (SCFGR2) | 32 | RW | 0000_0000h |
| 140h | Slave Address Match Register (SAMR) | 32 | RW | 0000_0000h |
| 150h | Slave Address Status Register (SASR) | 32 | RO | 0000_4000h |
| 154h | Slave Transmit ACK Register (STAR) | 32 | RW | 0000_0000h |
| 160h | Slave Transmit Data Register (STDR) | 32 | WO | 0000_0000h |
| 170h | Slave Receive Data Register (SRDR) | 32 | RO | 0000_4000h |

### 55.3.3.2 Version ID Register (VERID)

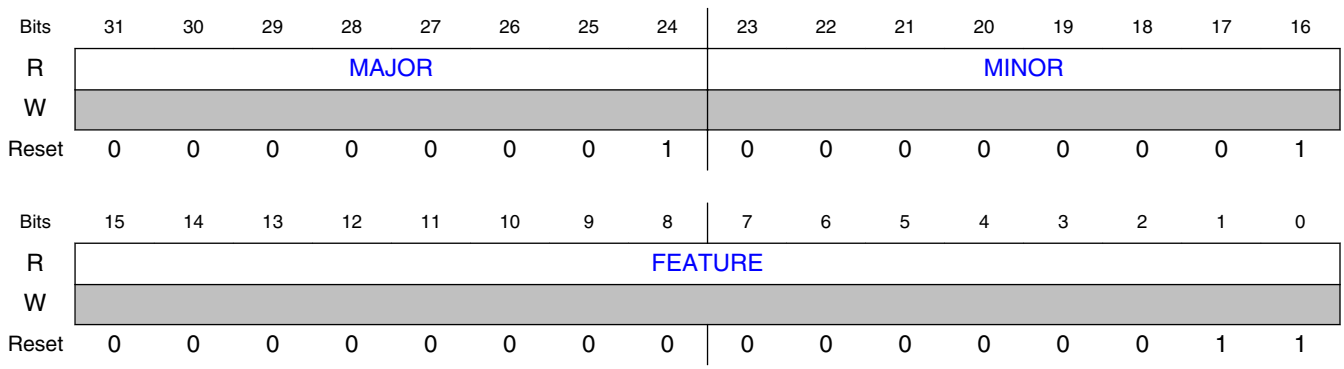
#### 55.3.3.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 55.3.3.2.2 Function

.

#### 55.3.3.2.3 Diagram




#### 55.3.3.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module design specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module design specification. Read-only field. |
| 15-0<br>FEATURE | Feature Specification Number<br>Returns the feature set number. Read-only field.<br>0000000000000010b - Master only, with standard feature set<br>0000000000000011b - Master and slave, with standard feature set |

### 55.3.3.3 Parameter Register (PARAM)

#### 55.3.3.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 55.3.3.3.2 Function

.

#### 55.3.3.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MRXFIFO | | | | 0 | | | | MTXFIFO | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 55.3.3.3.4 Fields

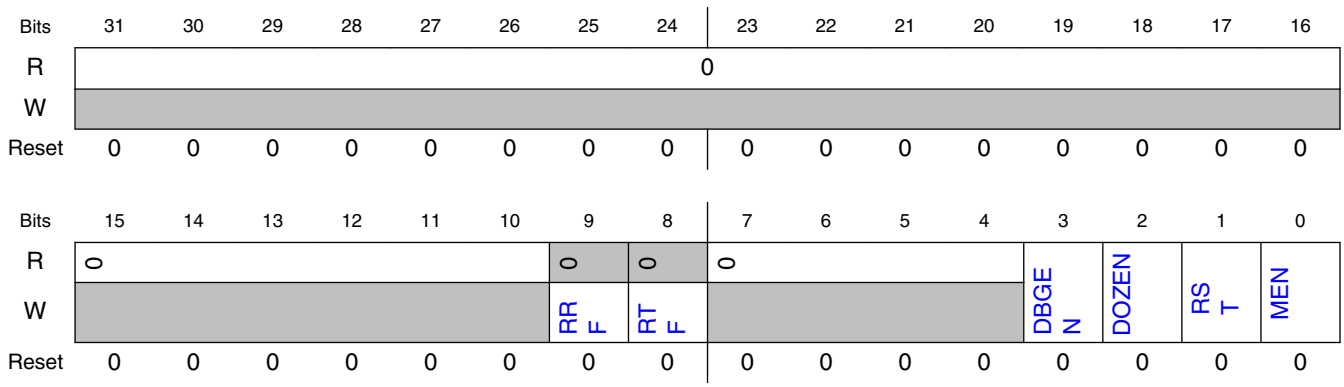
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>MRXFIFO | Master Receive FIFO Size<br>Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$ |
| 7-4<br>— | Reserved |
| 3-0<br>MTXFIFO | Master Transmit FIFO Size<br>Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$ |

### 55.3.3.4 Master Control Register (MCR)

### 55.3.3.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 10h |

### 55.3.3.4.2 Diagram




### 55.3.3.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Master is disabled in debug mode<br>1b - Master is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the master in Doze mode<br>0b - Master is enabled in Doze mode<br>1b - Master is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>MEN | Master Enable<br>0b - Master logic is disabled<br>1b - Master logic is enabled |

## 55.3.3.5 Master Status Register (MSR)

### 55.3.3.5.1 Offset

| Register | Offset |
|---|---|
| MSR | 14h |

### 55.3.3.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 55.3.3.5.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24<br>MBF | Master Busy Flag<br>0b - I2C Master is idle<br>1b - I2C Master is busy |
| 23-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set.<br>0b - Have not received matching data<br>1b - Have received matching data |
| 13<br>PLTF | Pin Low Timeout Flag<br>Will set when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR3[PINLOW]), even when the LPI2C master is idle.<br>• Software is responsible for resolving the pin low condition.<br>• Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues.<br>• Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared.<br>0b - Pin low timeout has not occurred or is disabled<br>1b - Pin low timeout has occurred |
| 12<br>FEF | FIFO Error Flag<br>Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared.<br>0b - No error<br>1b - Master sending or receiving data without a START condition |
| 11<br>ALF | Arbitration Lost Flag<br>Set:<br>• if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus<br>• or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data<br>When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag has been cleared.<br>0b - Master has not lost arbitration<br>1b - Master has lost arbitration |
| 10<br>NDF | NACK Detect Flag<br>Set if the LPI2C master detects a NACK when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated.<br>0b - Unexpected NACK was not detected<br>1b - Unexpected NACK was detected |
| 9<br>SDF | STOP Detect Flag<br>Set when the LPI2C master generates a STOP condition.<br>0b - Master has not generated a STOP condition<br>1b - Master has generated a STOP condition |
| 8<br>EPF | End Packet Flag<br>Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition.<br>0b - Master has not generated a STOP or Repeated START condition<br>1b - Master has generated a STOP or Repeated START condition |
| 7-2 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER.<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER.<br>0b - Transmit data is not requested<br>1b - Transmit data is requested |

## 55.3.3.6 Master Interrupt Enable Register (MIER)

### 55.3.3.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 18h |

### 55.3.3.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMIE | PLTIE | FEIE | ALIE | NDIE | SDIE | EPIE | 0 | | | | | | RDIE | TDIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.6.3 Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14 | Data Match Interrupt Enable |

*Table continues on the next page...*

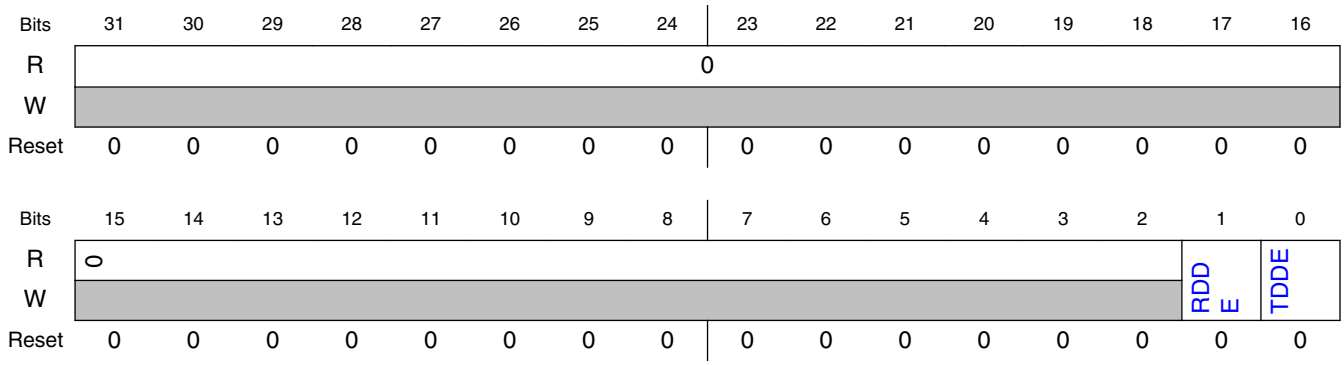
| Field | Function |
|---|---|
| DMIE | 0b - Disabled<br>1b - Enabled |
| 13<br>PLTIE | Pin Low Timeout Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>FEIE | FIFO Error Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>ALIE | Arbitration Lost Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>NDIE | NACK Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>EPIE | End Packet Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.3.7 Master DMA Enable Register (MDER)

### 55.3.3.7.1 Offset

| Register | Offset |
|---|---|
| MDER | 1Ch |

### 55.3.3.7.2 Diagram




### 55.3.3.7.3 Fields

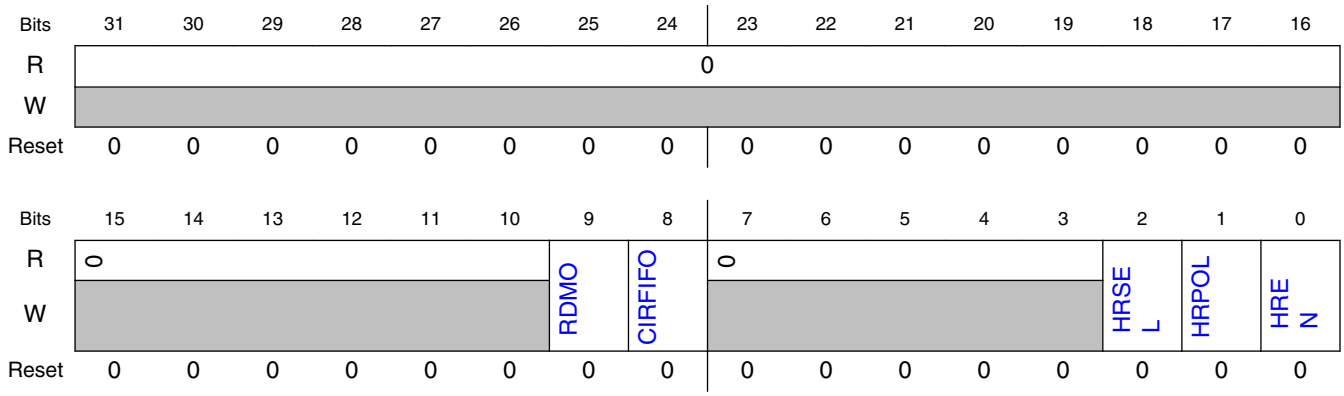
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.3.8 Master Configuration Register 0 (MCFGR0)

### 55.3.3.8.1 Offset

| Register | Offset |
|---|---|
| MCFGR0 | 20h |

## 55.3.3.8.2 Diagram




## 55.3.3.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change).<br>0b - Host request input is pin HREQ<br>1b - Host request input is input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field should remain static (Host Request Polarity should not change).<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable |

| Field | Function |
|---|---|
| | When enabled, the LPI2C master will only initiate a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request.<br>0b - Host request input is disabled<br>1b - Host request input is enabled |

## 55.3.3.9 Master Configuration Register 1 (MCFGR1)

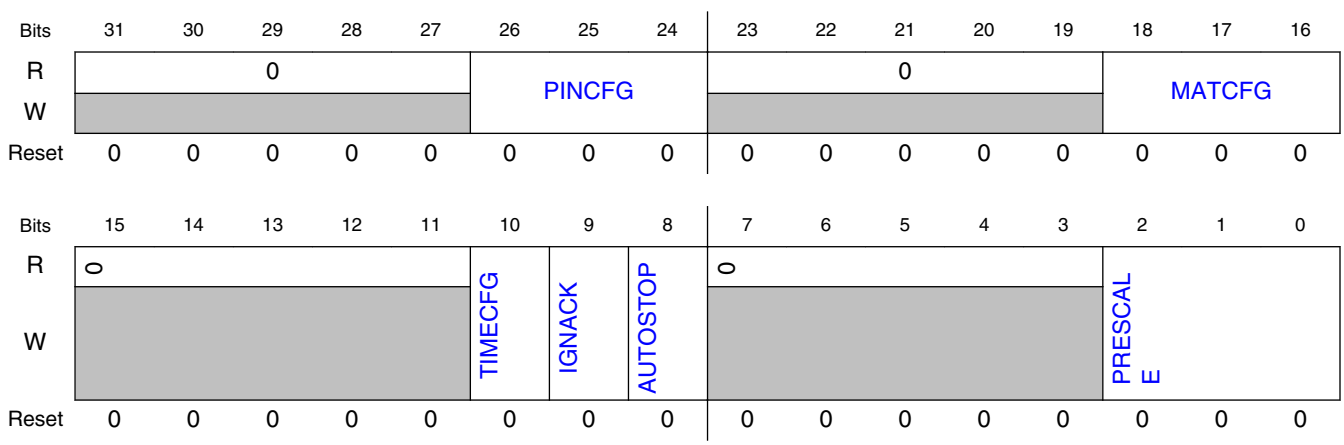
### 55.3.3.9.1 Offset

| Register | Offset |
|---|---|
| MCFGR1 | 24h |

### 55.3.3.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

### 55.3.3.9.3 Diagram




### 55.3.3.9.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>PINCFG | Pin Configuration |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Configures the pin mode for LPI2C.<br>**Table 55-5. 2-pin / 4-pin pin configurations for masters and slaves**<br>PINCFG / SCL / SDA pins / SCLS / SDAS pins<br>000 / Bi-directional open drain for master and slave / Not used<br>001 / Output-only (ultra-fast mode) open drain for master and slave / Not used<br>010 / Bi-directional push-pull for master and slave / Not used<br>011 / Input only for master and slave / Output-only push-pull for master and slave<br>100 / Bi-directional open drain for master / Bi-directional open drain for slave<br>101 / Output-only (ultra-fast mode) open drain for master / Output-only open drain for slave<br>110 / Bi-directional push-pull for master / Bi-directional push-pull for slave<br>111 / Input only for master and slave / Inverted output-only push-pull for master and slave<br>000b - 2-pin open drain mode<br>001b - 2-pin output only mode (ultra-fast mode)<br>010b - 2-pin push-pull mode<br>011b - 4-pin push-pull mode<br>100b - 2-pin open drain mode with separate LPI2C slave<br>101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave<br>110b - 2-pin push-pull mode with separate LPI2C slave<br>111b - 4-pin push-pull mode (inverted outputs) |
| 23-19<br>— | Reserved |
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that will cause the DMF to set.<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match is enabled (1st data word equals MATCH0 OR MATCH1)<br>011b - Match is enabled (any data word equals MATCH0 OR MATCH1)<br>100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1)<br>101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1)<br>110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1)<br>111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1) |
| 15-11<br>— | Reserved |
| 10<br>TIMECFG | Timeout Configuration<br>0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout<br>1b - Pin Low Timeout Flag will set if either SCL or SDA is low for longer than the configured timeout |
| 9<br>IGNACK | IGNACK<br>When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode.<br>0b - LPI2C Master will receive ACK and NACK normally<br>1b - LPI2C Master will treat a received NACK as if it (NACK) was an ACK |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>AUTOSTOP | Automatic STOP Generation<br>When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command.<br>0b - No effect<br>1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy |
| 7-3<br>— | Reserved |
| 2-0<br>PRESCALE | Prescaler<br>Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 55.3.3.10 Master Configuration Register 2 (MCFGR2)

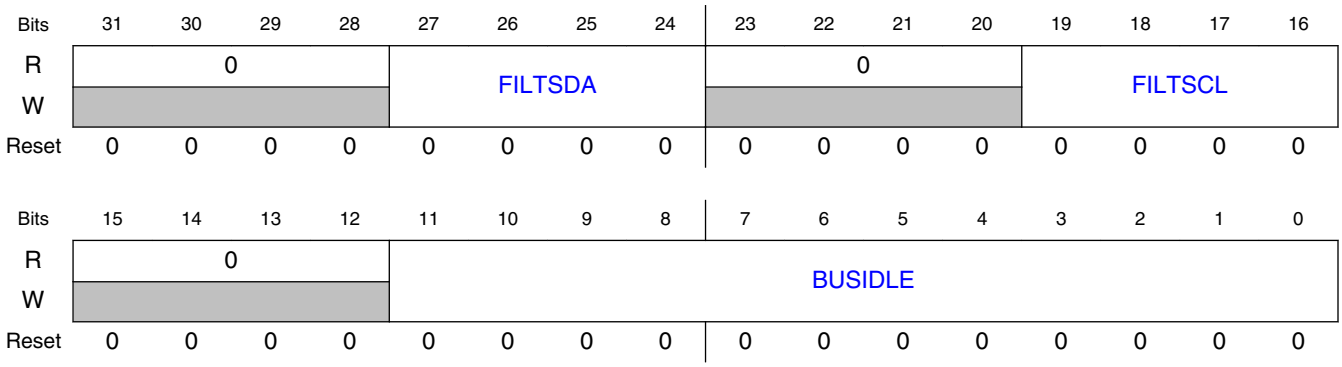
### 55.3.3.10.1 Offset

| Register | Offset |
|---|---|
| MCFGR2 | 28h |

### 55.3.3.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

## 55.3.3.10.3 Diagram




## 55.3.3.10.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C master digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C master digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 15-12<br>— | Reserved |
| 11-0<br>BUSIDLE | Bus Idle Timeout<br>Configures the bus idle timeout period in clock cycles.<br>• If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition<br>• When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled |

## 55.3.3.11 Master Configuration Register 3 (MCFGR3)

### 55.3.3.11.1 Offset

| Register | Offset |
|---|---|
| MCFGR3 | 2Ch |

### 55.3.3.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

### 55.3.3.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PINLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINLOW | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.11.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-8<br>PINLOW | Pin Low Timeout<br>Configures the pin low timeout flag in clock cycles.<br>• If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then PLTF is set<br>• When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled |
| 7-0<br>— | Reserved |

## 55.3.3.12 Master Data Match Register (MDMR)

### 55.3.3.12.1 Offset

| Register | Offset |
|---|---|
| MDMR | 40h |

### 55.3.3.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

### 55.3.3.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>MATCH1 | Match 1 Value<br>Compared against the received data when receive data match is enabled. |
| 15-8<br>— | Reserved |
| 7-0<br>MATCH0 | Match 0 Value<br>Compared against the received data when receive data match is enabled. |

## 55.3.3.13 Master Clock Configuration Register 0 (MCCR0)

### 55.3.3.13.1 Offset

| Register | Offset |
|---|---|
| MCCR0 | 48h |

### 55.3.3.13.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

### 55.3.3.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DATAVD | | | | | | 0 | | SETHOLD | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | CLKHI | | | | | | 0 | | CLKLO | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.13.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br><br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.3.14 Master Clock Configuration Register 1 (MCCR1)

### 55.3.3.14.1 Offset

| Register | Offset |
|---|---|
| MCCR1 | 50h |

### 55.3.3.14.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0), before switching to high speed mode (with timing configured by MCCR1).

### 55.3.3.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DATAVD | | | | | | 0 | | SETHOLD | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | CLKHI | | | | | | 0 | | CLKLO | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.14.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. The Data Valid Delay must be configured to be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.3.15 Master FIFO Control Register (MFCR)

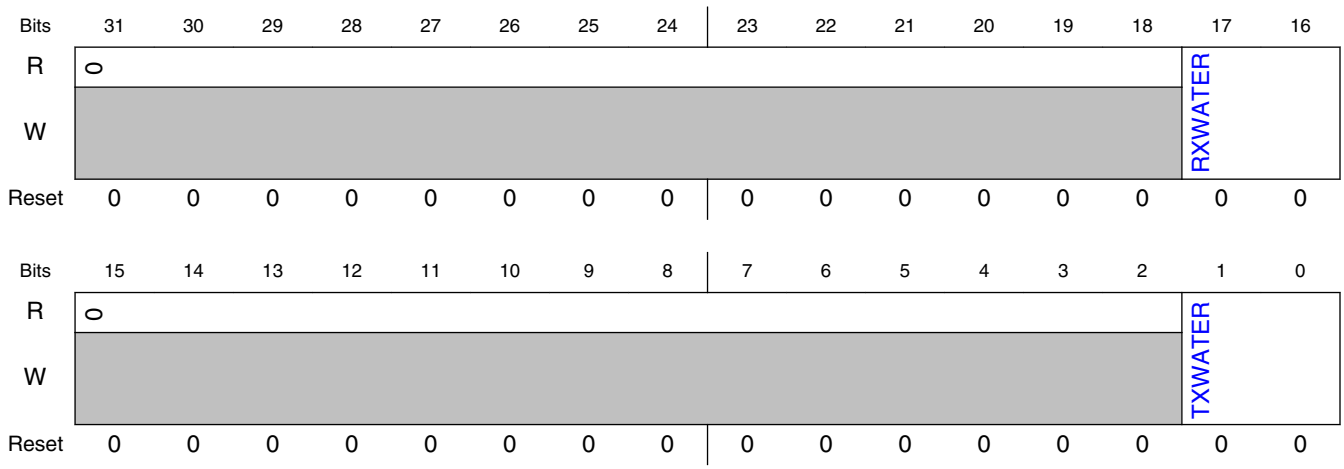
### 55.3.3.15.1 Offset

| Register | Offset |
|---|---|
| MFCR | 58h |

### 55.3.3.15.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

### 55.3.3.15.3 Diagram




### 55.3.3.15.4 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |
| 15-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |

## 55.3.3.16 Master FIFO Status Register (MFSR)

### 55.3.3.16.1 Offset

| Register | Offset |
|---|---|
| MFSR | 5Ch |

### 55.3.3.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.16.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words in the receive FIFO. |
| 15-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words in the transmit FIFO. |

## 55.3.3.17 Master Transmit Data Register (MTDR)

### 55.3.3.17.1 Offset

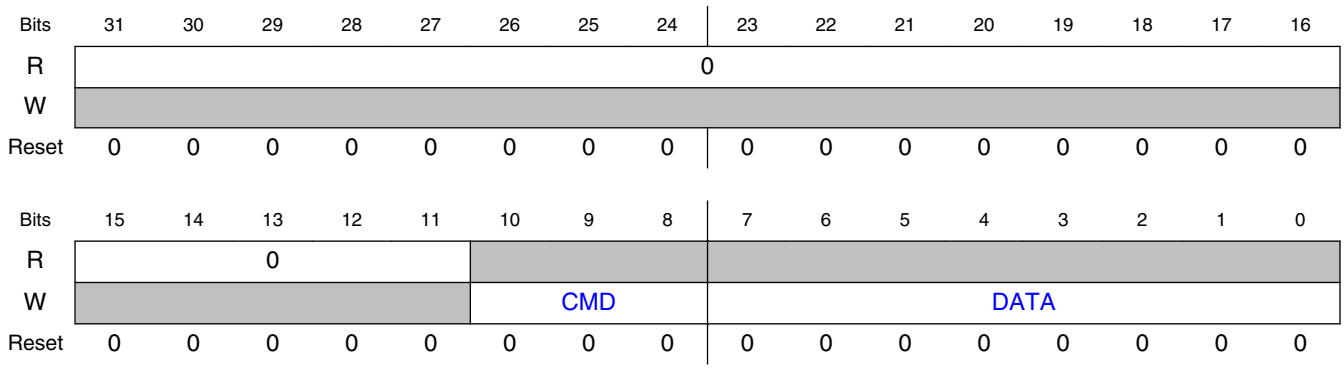
| Register | Offset |
|---|---|
| MTDR | 60h |

### 55.3.3.17.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.

- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit or 32-bit will write both the CMD and DATA fields and increment the FIFO.

### 55.3.3.17.3 Diagram




### 55.3.3.17.4 Fields

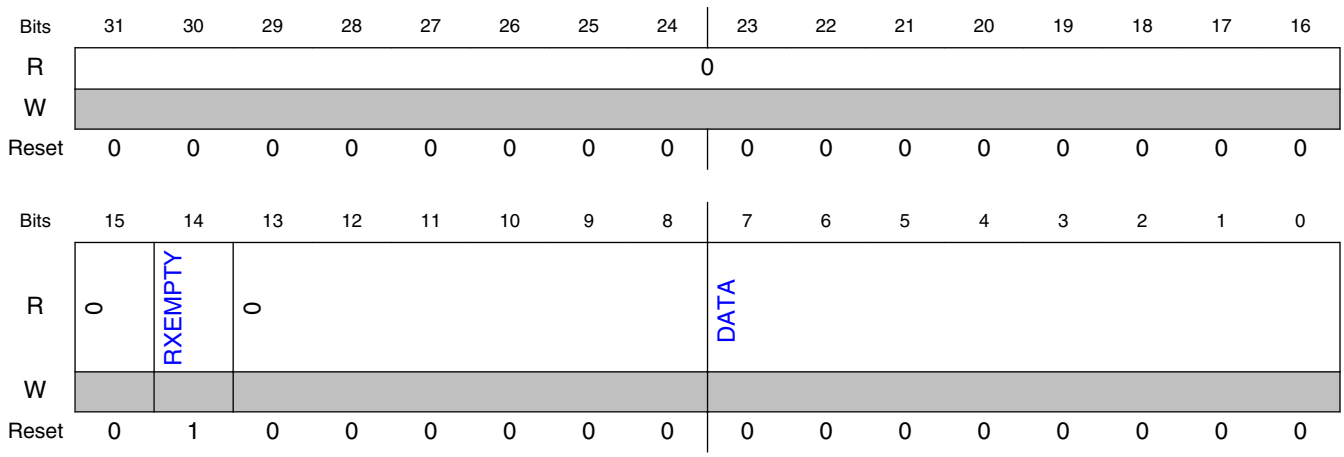
| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>CMD | Command Data<br>000b - Transmit DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes<br>010b - Generate STOP condition<br>011b - Receive and discard (DATA[7:0] + 1) bytes<br>100b - Generate (repeated) START and transmit address in DATA[7:0]<br>101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.<br>110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode<br>111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned. |
| 7-0<br>DATA | Transmit Data<br>Performing an 8-bit write to DATA will zero extend the CMD field. |

## 55.3.3.18 Master Receive Data Register (MRDR)

### 55.3.3.18.1 Offset

| Register | Offset |
|---|---|
| MRDR | 70h |

### 55.3.3.18.2 Diagram




### 55.3.3.18.3 Fields

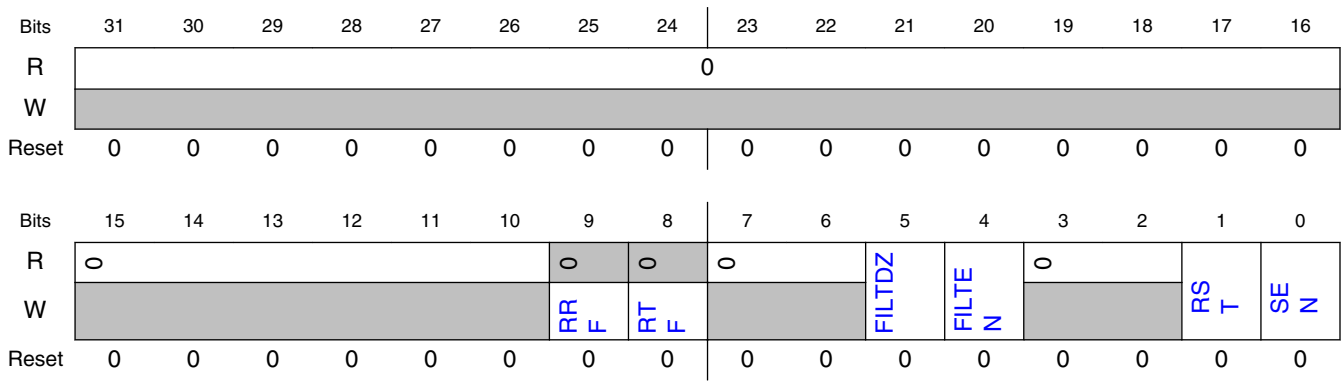
| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14<br>RXEMPTY | RX Empty<br>0b - Receive FIFO is not empty<br>1b - Receive FIFO is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data. |

## 55.3.3.19 Slave Control Register (SCR)

## 55.3.3.19.1 Offset

| Register | Offset |
|---|---|
| SCR | 110h |

## 55.3.3.19.2 Diagram




## 55.3.3.19.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive Data Register is now empty |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit Data Register is now empty |
| 7-6<br>— | Reserved |
| 5<br>FILTDZ | Filter Doze Enable<br>Filter Doze Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Filter remains enabled in Doze mode<br>1b - Filter is disabled in Doze mode |
| 4<br>FILTEN | Filter Enable<br>Filter Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Disable digital filter and output delay counter for slave mode<br>1b - Enable digital filter and output delay counter for slave mode |
| 3-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | 0b - Slave mode logic is not reset<br>1b - Slave mode logic is reset |
| 0<br>SEN | Slave Enable<br>0b - I2C Slave mode is disabled<br>1b - I2C Slave mode is enabled |

## 55.3.3.20 Slave Status Register (SSR)

### 55.3.3.20.1 Offset

| Register | Offset |
|---|---|
| SSR | 114h |

### 55.3.3.20.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | SBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARF | GCF | AM1F | AM0F | FEF | BEF | SDF | RSF | 0 | | | | TAF | AVF | RDF | TDF |
| W | | | | | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.20.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>Indicates if an I2C bus is idle or busy.<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24 | Slave Busy Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SBF | Indicates if an I2C slave is idle or busy.<br>0b - I2C Slave is idle<br>1b - I2C Slave is busy |
| 23-16<br>— | Reserved |
| 15<br>SARF | SMBus Alert Response Flag<br>• SMBus Alert Response Flag is cleared by reading the Address Status Register<br>• SMBus Alert Response Flag cannot generate an asynchronous wakeup<br><br>0b - SMBus Alert Response is disabled or not detected<br>1b - SMBus Alert Response is enabled and detected |
| 14<br>GCF | General Call Flag<br>Indicates whether a slave has detected the General Call Address.<br>• General Call Flag is cleared by reading the Address Status Register<br>• General Call Flag cannot generate an asynchronous wakeup<br><br>0b - Slave has not detected the General Call Address or the General Call Address is disabled<br>1b - Slave has detected the General Call Address |
| 13<br>AM1F | Address Match 1 Flag<br>Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by ADDRCFG.<br>• Address Match 1 Flag is cleared by reading the Address Status Register<br>• Address Match 1 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address<br>1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address |
| 12<br>AM0F | Address Match 0 Flag<br>Indicates that the received address has matched the ADDR0 field as configured by ADDRCFG.<br>• Address Match 0 Flag is cleared by reading the Address Status Register<br>• Address Match 0 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR0 matching address<br>1b - Have received an ADDR0 matching address |
| 11<br>FEF | FIFO Error Flag<br>FIFO error flag can only be set when clock stretching is disabled.<br>0b - FIFO underflow or overflow was not detected<br>1b - FIFO underflow or overflow was detected |
| 10<br>BEF | Bit Error Flag<br>Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore the rest of the transfer until the next (repeated) START condition.<br>0b - Slave has not detected a bit error<br>1b - Slave has detected a bit error |
| 9<br>SDF | STOP Detect Flag<br>STOP Detect Flag will set when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte.<br>0b - Slave has not detected a STOP condition<br>1b - Slave has detected a STOP condition |
| 8<br>RSF | Repeated Start Flag |

*Table continues on the next page...*

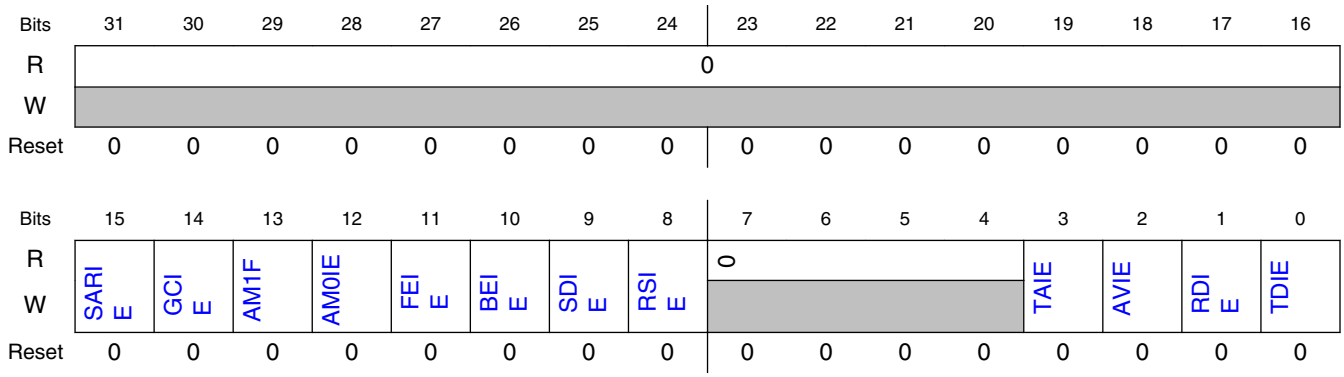
| Field | Function |
|---|---|
| | Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag does not set when the slave first detects a START condition.<br>0b - Slave has not detected a Repeated START condition<br>1b - Slave has detected a Repeated START condition |
| 7-4<br>— | Reserved |
| 3<br>TAF | Transmit ACK Flag<br>Transmit ACK Flag is cleared by writing the Transmit ACK register.<br>0b - Transmit ACK/NACK is not required<br>1b - Transmit ACK/NACK is required |
| 2<br>AVF | Address Valid Flag<br>Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register.<br>0b - Address Status Register is not valid<br>1b - Address Status Register is valid |
| 1<br>RDF | Receive Data Flag<br>Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set.<br>0b - Receive data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared.<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 55.3.3.21 Slave Interrupt Enable Register (SIER)

### 55.3.3.21.1 Offset

| Register | Offset |
|---|---|
| SIER | 118h |

## 55.3.3.21.2 Diagram




## 55.3.3.21.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SARIE | SMBus Alert Response Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 14<br>GCIE | General Call Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 13<br>AM1F | Address Match 1 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>AM0IE | Address Match 0 Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>FEIE | FIFO Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>BEIE | Bit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>RSIE | Repeated Start Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-4<br>— | Reserved |
| 3<br>TAIE | Transmit ACK Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>AVIE | Address Valid Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.3.22 Slave DMA Enable Register (SDER)

### 55.3.3.22.1 Offset

| Register | Offset |
|---|---|
| SDER | 11Ch |

### 55.3.3.22.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | AVDE | RDDE | TDDE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.22.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>AVDE | Address Valid DMA Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set Receive Data Configuration (SCFGR1[RXCFG]), to allow the DMA to read the address from the Receive Data Register.<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.3.23 Slave Configuration Register 1 (SCFGR1)

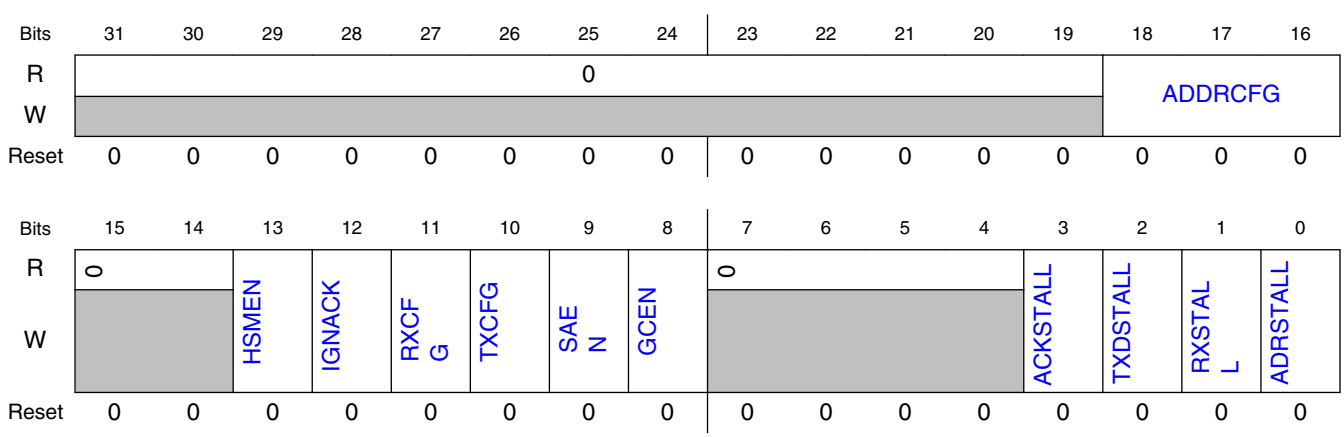
### 55.3.3.23.1 Offset

| Register | Offset |
|---|---|
| SCFGR1 | 124h |

### 55.3.3.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

### 55.3.3.23.3 Diagram



### 55.3.3.23.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>ADDRCFG | Address Configuration<br>Configures the condition that will cause an address to match.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or Address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or Address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or Address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or Address match 1 (7-bit)<br>110b - From Address match 0 (7-bit) to Address match 1 (7-bit)<br>111b - From Address match 0 (10-bit) to Address match 1 (10-bit) |
| 15-14<br>— | Reserved |
| 13<br>HSMEN | High Speed Mode Enable<br>Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, the FILTEN and ACKSTALL bits are ignored until the next STOP condition is detected.<br>0b - Disables detection of HS-mode master code<br>1b - Enables detection of HS-mode master code |
| 12<br>IGNACK | Ignore NACK<br>When Ignore NACK is set, the LPI2C slave will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode.<br>0b - Slave will end transfer when NACK is detected<br>1b - Slave will not end transfer when NACK detected |
| 11<br>RXCFG | Receive Data Configuration<br>0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]).<br>1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]). |
| 10<br>TXCFG | Transmit Flag Configuration<br>The transmit data flag will always assert before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO.<br>• When TXCFG=0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer.<br>• When TXCFG=1, the Transmit Data flag will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.<br>0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty<br>1b - Transmit Data Flag will assert whenever the Transmit Data register is empty |
| 9 | SMBus Alert Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SAEN | 0b - Disables match on SMBus Alert<br>1b - Enables match on SMBus Alert |
| 8<br>GCEN | General Call Enable<br>0b - General Call address is disabled<br>1b - General Call address is enabled |
| 7-4<br>— | Reserved |
| 3<br>ACKSTALL | ACK SCL Stall<br>Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Transmit ACK Register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode.<br>When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]).<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 2<br>TXDSTALL | TX Data SCL Stall<br>Enables SCL clock stretching when the Transmit Data flag (SSR[TDF]) is set during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 1<br>RXSTALL | RX SCL Stall<br>Enables SCL clock stretching when the Receive Data flag (SSR[RDF]) is set during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 0<br>ADRSTALL | Address SCL Stall<br>Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |

## 55.3.3.24 Slave Configuration Register 2 (SCFGR2)

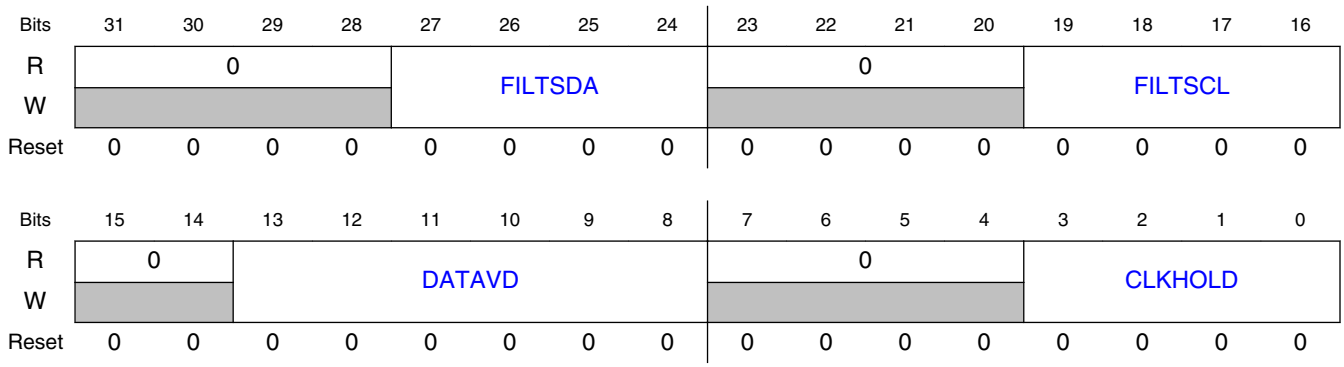
### 55.3.3.24.1 Offset

| Register | Offset |
|---|---|
| SCFGR2 | 128h |

### 55.3.3.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

### 55.3.3.24.3 Diagram




### 55.3.3.24.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C slave digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C slave digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 15-14<br>— | Reserved |
| 13-8<br>DATAVD | Data Valid Delay<br>Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles.<br>• The data valid delay must be configured to be less than the minimum SCL low period<br>• The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode |
| 7-4<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>CLKHOLD | Clock Hold Time<br>Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled.<br>• The minimum hold time is equal to CLKHOLD+3 cycles<br>• The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode |

## 55.3.3.25 Slave Address Match Register (SAMR)

### 55.3.3.25.1 Offset

| Register | Offset |
|---|---|
| SAMR | 140h |

### 55.3.3.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

### 55.3.3.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR1 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR0 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.25.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-17<br>ADDR1 | Address 1 Value<br>Compared against the received address to detect the Slave Address. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17]<br>• In 7-bit mode, the address is compared to ADDR1[23:17] |
| 16-11<br>— | Reserved |
| 10-1<br>ADDR0 | Address 0 Value<br>Compared against the received address to detect the Slave Address.<br>• In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1]<br>• In 7-bit mode, the address is compared to ADDR0[7:1]<br>• |
| 0<br>— | Reserved |

## 55.3.3.26 Slave Address Status Register (SASR)

### 55.3.3.26.1 Offset

| Register | Offset |
|---|---|
| SASR | 150h |

### 55.3.3.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ANV | 0 | | | RADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.26.3 Fields

| Field | Function |
|---|---|
| 31-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>ANV | Address Not Valid<br>0b - Received Address (RADDR) is valid<br>1b - Received Address (RADDR) is not valid |
| 13-11<br>— | Reserved |
| 10-0<br>RADDR | Received Address<br>The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register.<br>• In 7-bit mode, the address byte is store in RADDR[7:0]<br>• In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0] |

## 55.3.3.27 Slave Transmit ACK Register (STAR)

### 55.3.3.27.1 Offset

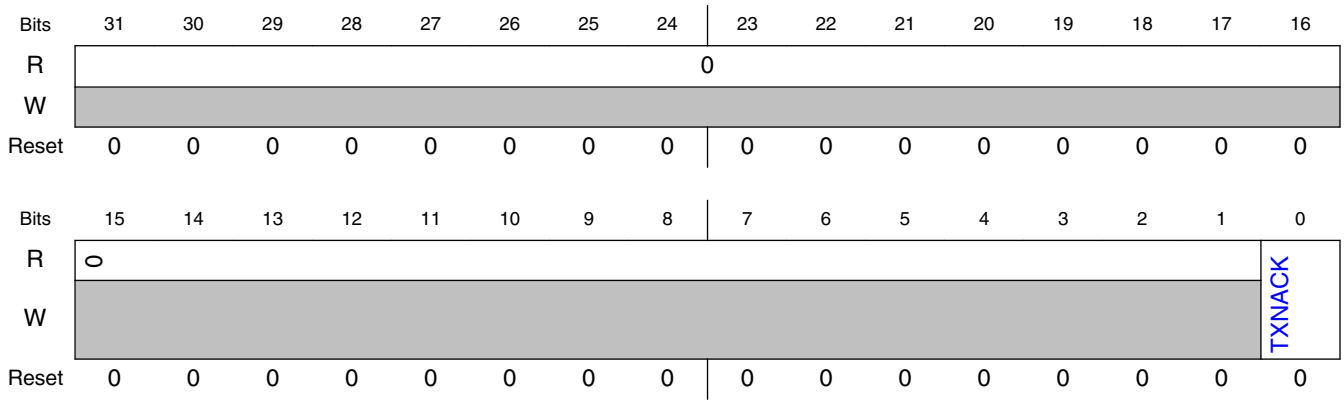
| Register | Offset |
|---|---|
| STAR | 154h |

### 55.3.3.27.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR1[ACKSTALL].

- The ACKSTALL bit will enable clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).

### 55.3.3.27.3 Diagram




### 55.3.3.27.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TXNACK | Transmit NACK<br>After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK.<br>• When ACKSTALL is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that will stall the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK).<br>• To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle.<br><br>0b - Write a Transmit ACK for each received word<br>1b - Write a Transmit NACK for each received word |

## 55.3.3.28 Slave Transmit Data Register (STDR)

### 55.3.3.28.1 Offset

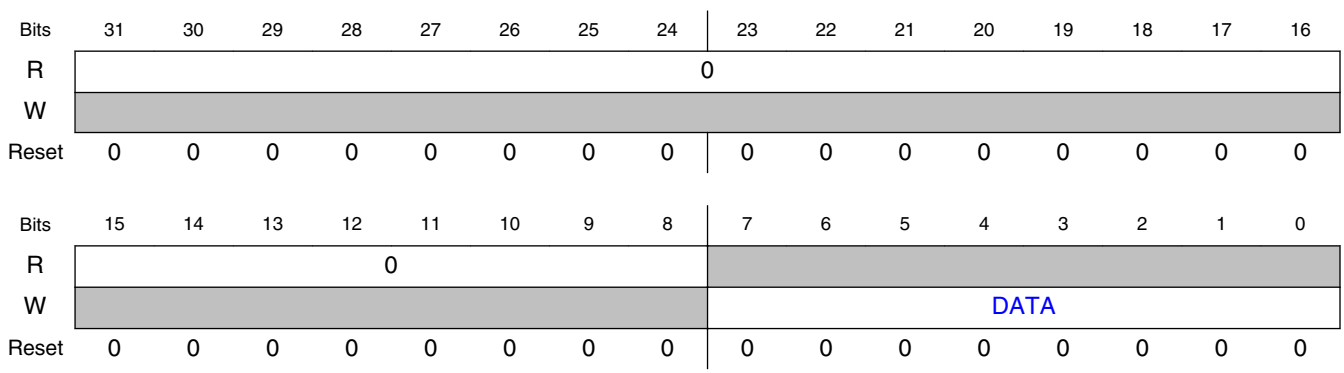
| Register | Offset |
|---|---|
| STDR | 160h |

### 55.3.3.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

### 55.3.3.28.3 Diagram




### 55.3.3.28.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>DATA | Transmit Data<br>Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register |

## 55.3.3.29 Slave Receive Data Register (SRDR)

### 55.3.3.29.1 Offset

| Register | Offset |
|---|---|
| SRDR | 170h |

### 55.3.3.29.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SOF | RXEMPTY | 0 | | | | | | DATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.3.29.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SOF | Start Of Frame<br>0b - Indicates this is not the first data word since a (repeated) START or STOP condition<br>1b - Indicates this is the first data word since a (repeated) START or STOP condition |
| 14<br>RXEMPTY | RX Empty<br>0b - The Receive Data Register is not empty<br>1b - The Receive Data Register is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Slave Receive Data Register returns the data received by the I2C slave |

## 55.3.4 LPI2C3 register descriptions

### 55.3.4.1 LPI2C3 Memory map

LPI2C3 base address: 4102_E000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Master Control Register (MCR) | 32 | RW | 0000_0000h |
| 14h | Master Status Register (MSR) | 32 | W1C | 0000_0001h |
| 18h | Master Interrupt Enable Register (MIER) | 32 | RW | 0000_0000h |
| 1Ch | Master DMA Enable Register (MDER) | 32 | RW | 0000_0000h |
| 20h | Master Configuration Register 0 (MCFGR0) | 32 | RW | 0000_0000h |
| 24h | Master Configuration Register 1 (MCFGR1) | 32 | RW | 0000_0000h |
| 28h | Master Configuration Register 2 (MCFGR2) | 32 | RW | 0000_0000h |
| 2Ch | Master Configuration Register 3 (MCFGR3) | 32 | RW | 0000_0000h |
| 40h | Master Data Match Register (MDMR) | 32 | RW | 0000_0000h |
| 48h | Master Clock Configuration Register 0 (MCCR0) | 32 | RW | 0000_0000h |
| 50h | Master Clock Configuration Register 1 (MCCR1) | 32 | RW | 0000_0000h |
| 58h | Master FIFO Control Register (MFCR) | 32 | RW | 0000_0000h |
| 5Ch | Master FIFO Status Register (MFSR) | 32 | RO | 0000_0000h |
| 60h | Master Transmit Data Register (MTDR) | 32 | WO | 0000_0000h |
| 70h | Master Receive Data Register (MRDR) | 32 | RO | 0000_4000h |
| 110h | Slave Control Register (SCR) | 32 | RW | 0000_0000h |
| 114h | Slave Status Register (SSR) | 32 | W1C | 0000_0000h |
| 118h | Slave Interrupt Enable Register (SIER) | 32 | RW | 0000_0000h |
| 11Ch | Slave DMA Enable Register (SDER) | 32 | RW | 0000_0000h |
| 124h | Slave Configuration Register 1 (SCFGR1) | 32 | RW | 0000_0000h |
| 128h | Slave Configuration Register 2 (SCFGR2) | 32 | RW | 0000_0000h |
| 140h | Slave Address Match Register (SAMR) | 32 | RW | 0000_0000h |
| 150h | Slave Address Status Register (SASR) | 32 | RO | 0000_4000h |
| 154h | Slave Transmit ACK Register (STAR) | 32 | RW | 0000_0000h |
| 160h | Slave Transmit Data Register (STDR) | 32 | WO | 0000_0000h |
| 170h | Slave Receive Data Register (SRDR) | 32 | RO | 0000_4000h |

### 55.3.4.2 Version ID Register (VERID)

#### 55.3.4.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 55.3.4.2.2 Function

.

#### 55.3.4.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 55.3.4.2.4 Fields

| Field | Function |
|---|---|
| 31-24 | Major Version Number |
| MAJOR | Returns the major version number for the module design specification. Read-only field. |
| 23-16 | Minor Version Number |
| MINOR | Returns the minor version number for the module design specification. Read-only field. |
| 15-0 | Feature Specification Number |
| FEATURE | Returns the feature set number. Read-only field.<br>0000000000000010b - Master only, with standard feature set<br>0000000000000011b - Master and slave, with standard feature set |

### 55.3.4.3 Parameter Register (PARAM)

### 55.3.4.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 55.3.4.3.2 Function

.

### 55.3.4.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MRXFIFO | | | | 0 | | | | MTXFIFO | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

### 55.3.4.3.4 Fields

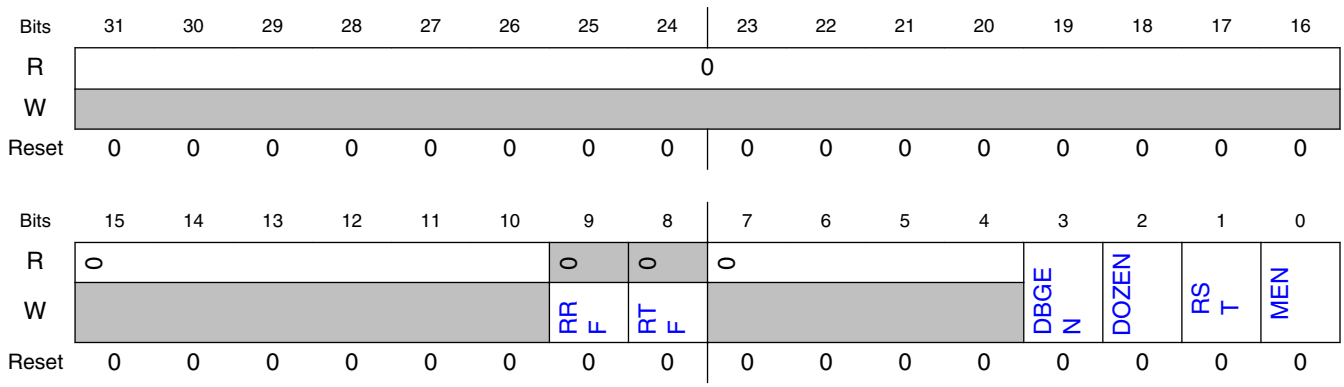
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>MRXFIFO | Master Receive FIFO Size<br>Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$ |
| 7-4<br>— | Reserved |
| 3-0<br>MTXFIFO | Master Transmit FIFO Size<br>Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$ |

## 55.3.4.4 Master Control Register (MCR)

### 55.3.4.4.1 Offset

| Register | Offset |
|---|---|
| MCR | 10h |

### 55.3.4.4.2 Diagram




### 55.3.4.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Master is disabled in debug mode<br>1b - Master is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the master in Doze mode<br>0b - Master is enabled in Doze mode<br>1b - Master is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 0<br>MEN | Master Enable<br>0b - Master logic is disabled<br>1b - Master logic is enabled |

## 55.3.4.5 Master Status Register (MSR)

### 55.3.4.5.1 Offset

| Register | Offset |
|---|---|
| MSR | 14h |

### 55.3.4.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 55.3.4.5.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24<br>MBF | Master Busy Flag<br>0b - I2C Master is idle<br>1b - I2C Master is busy |
| 23-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set.<br>0b - Have not received matching data<br>1b - Have received matching data |
| 13<br>PLTF | Pin Low Timeout Flag<br>Will set when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR3[PINLOW]), even when the LPI2C master is idle.<br>• Software is responsible for resolving the pin low condition.<br>• Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues.<br>• Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared.<br>0b - Pin low timeout has not occurred or is disabled<br>1b - Pin low timeout has occurred |
| 12<br>FEF | FIFO Error Flag<br>Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared.<br>0b - No error<br>1b - Master sending or receiving data without a START condition |
| 11<br>ALF | Arbitration Lost Flag<br>Set:<br>• if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus<br>• or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data<br>When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag has been cleared.<br>0b - Master has not lost arbitration<br>1b - Master has lost arbitration |
| 10<br>NDF | NACK Detect Flag<br>Set if the LPI2C master detects a NACK when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated.<br>0b - Unexpected NACK was not detected<br>1b - Unexpected NACK was detected |
| 9<br>SDF | STOP Detect Flag<br>Set when the LPI2C master generates a STOP condition.<br>0b - Master has not generated a STOP condition<br>1b - Master has generated a STOP condition |
| 8<br>EPF | End Packet Flag<br>Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition.<br>0b - Master has not generated a STOP or Repeated START condition<br>1b - Master has generated a STOP or Repeated START condition |
| 7-2 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER.<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER.<br>0b - Transmit data is not requested<br>1b - Transmit data is requested |

## 55.3.4.6 Master Interrupt Enable Register (MIER)

### 55.3.4.6.1 Offset

| Register | Offset |
|---|---|
| MIER | 18h |

### 55.3.4.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMIE | PLTIE | FEIE | ALIE | NDIE | SDIE | EPIE | 0 | | | | | | RDIE | TDIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.6.3 Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14 | Data Match Interrupt Enable |

*Table continues on the next page...*

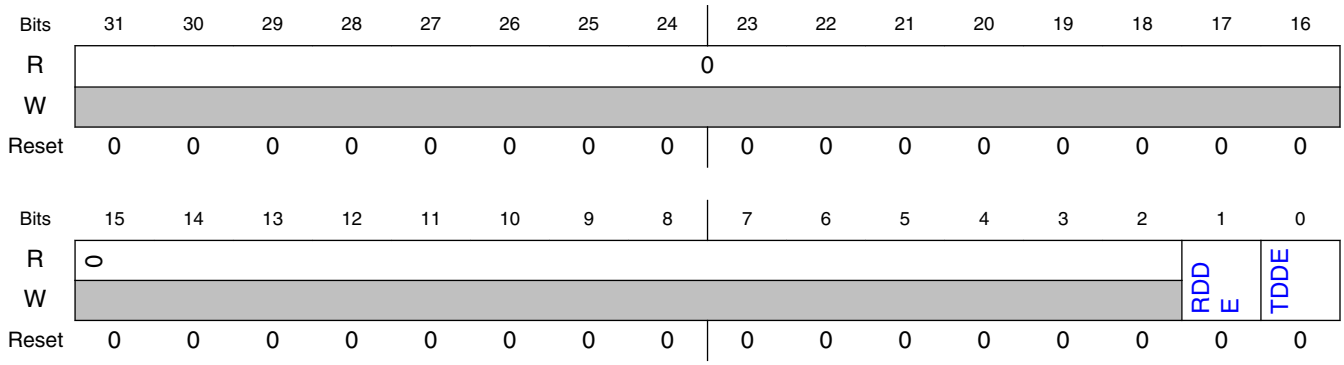
| Field | Function |
|---|---|
| DMIE | 0b - Disabled<br>1b - Enabled |
| 13<br>PLTIE | Pin Low Timeout Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>FEIE | FIFO Error Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>ALIE | Arbitration Lost Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>NDIE | NACK Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>EPIE | End Packet Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.4.7 Master DMA Enable Register (MDER)

### 55.3.4.7.1 Offset

| Register | Offset |
|---|---|
| MDER | 1Ch |

### 55.3.4.7.2 Diagram




### 55.3.4.7.3 Fields

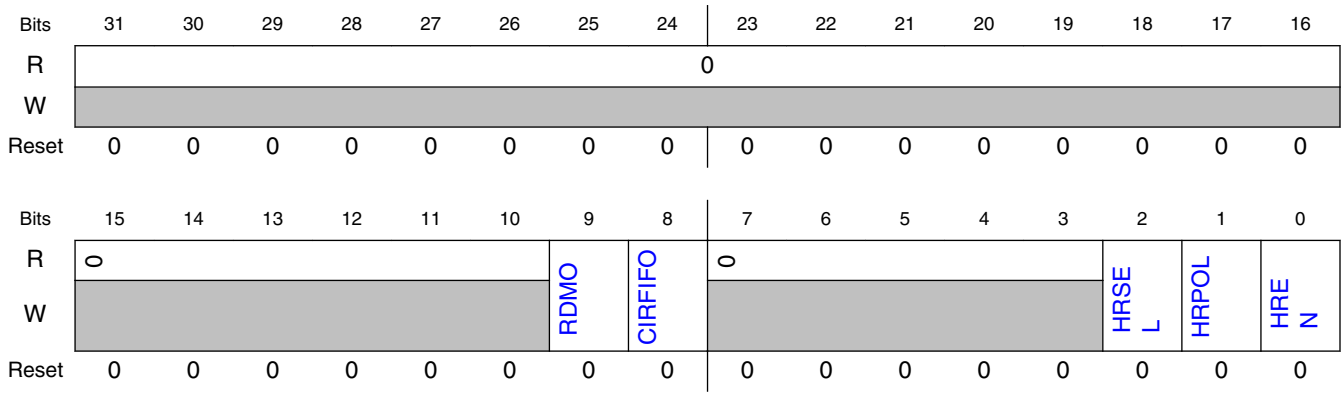
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.4.8 Master Configuration Register 0 (MCFGR0)

### 55.3.4.8.1 Offset

| Register | Offset |
|---|---|
| MCFGR0 | 20h |

### 55.3.4.8.2 Diagram




### 55.3.4.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change).<br>0b - Host request input is pin HREQ<br>1b - Host request input is input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field should remain static (Host Request Polarity should not change).<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable |

| Field | Function |
|---|---|
| | When enabled, the LPI2C master will only initiate a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request.<br>0b - Host request input is disabled<br>1b - Host request input is enabled |

## 55.3.4.9 Master Configuration Register 1 (MCFGR1)

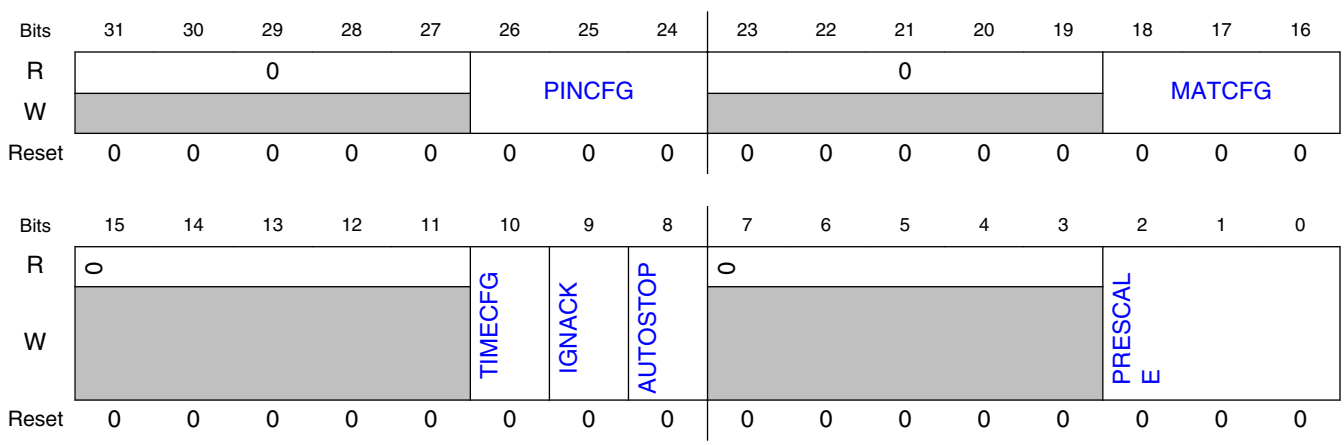
### 55.3.4.9.1 Offset

| Register | Offset |
|---|---|
| MCFGR1 | 24h |

### 55.3.4.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

### 55.3.4.9.3 Diagram




### 55.3.4.9.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>PINCFG | Pin Configuration |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Configures the pin mode for LPI2C.<br>Table 55-6. 2-pin / 4-pin pin configurations for masters and slaves
<table>
<tr><th>PINCFG</th><th>SCL / SDA pins</th><th>SCLS / SDAS pins</th></tr>
<tr><td>000</td><td>Bi-directional open drain for master and slave</td><td>Not used</td></tr>
<tr><td>001</td><td>Output-only (ultra-fast mode) open drain for master and slave</td><td>Not used</td></tr>
<tr><td>010</td><td>Bi-directional push-pull for master and slave</td><td>Not used</td></tr>
<tr><td>011</td><td>Input only for master and slave</td><td>Output-only push-pull for master and slave</td></tr>
<tr><td>100</td><td>Bi-directional open drain for master</td><td>Bi-directional open drain for slave</td></tr>
<tr><td>101</td><td>Output-only (ultra-fast mode) open drain for master</td><td>Output-only open drain for slave</td></tr>
<tr><td>110</td><td>Bi-directional push-pull for master</td><td>Bi-directional push-pull for slave</td></tr>
<tr><td>111</td><td>Input only for master and slave</td><td>Inverted output-only push-pull for master and slave</td></tr>
</table>
000b - 2-pin open drain mode<br>001b - 2-pin output only mode (ultra-fast mode)<br>010b - 2-pin push-pull mode<br>011b - 4-pin push-pull mode<br>100b - 2-pin open drain mode with separate LPI2C slave<br>101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave<br>110b - 2-pin push-pull mode with separate LPI2C slave<br>111b - 4-pin push-pull mode (inverted outputs)</td></tr>
<tr><td>23-19<br>—</td><td>Reserved</td></tr>
<tr><td>18-16<br>MATCFG</td><td>Match Configuration<br>Configures the condition that will cause the DMF to set.<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match is enabled (1st data word equals MATCH0 OR MATCH1)<br>011b - Match is enabled (any data word equals MATCH0 OR MATCH1)<br>100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1)<br>101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1)<br>110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1)<br>111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1)</td></tr>
<tr><td>15-11<br>—</td><td>Reserved</td></tr>
<tr><td>10<br>TIMECFG</td><td>Timeout Configuration<br>0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout<br>1b - Pin Low Timeout Flag will set if either SCL or SDA is low for longer than the configured timeout</td></tr>
<tr><td>9<br>IGNACK</td><td>IGNACK<br>When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode.<br>0b - LPI2C Master will receive ACK and NACK normally<br>1b - LPI2C Master will treat a received NACK as if it (NACK) was an ACK</td></tr>
</table>

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>AUTOSTOP | Automatic STOP Generation<br>When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command.<br>0b - No effect<br>1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy |
| 7-3<br>— | Reserved |
| 2-0<br>PRESCALE | Prescaler<br>Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

## 55.3.4.10 Master Configuration Register 2 (MCFGR2)

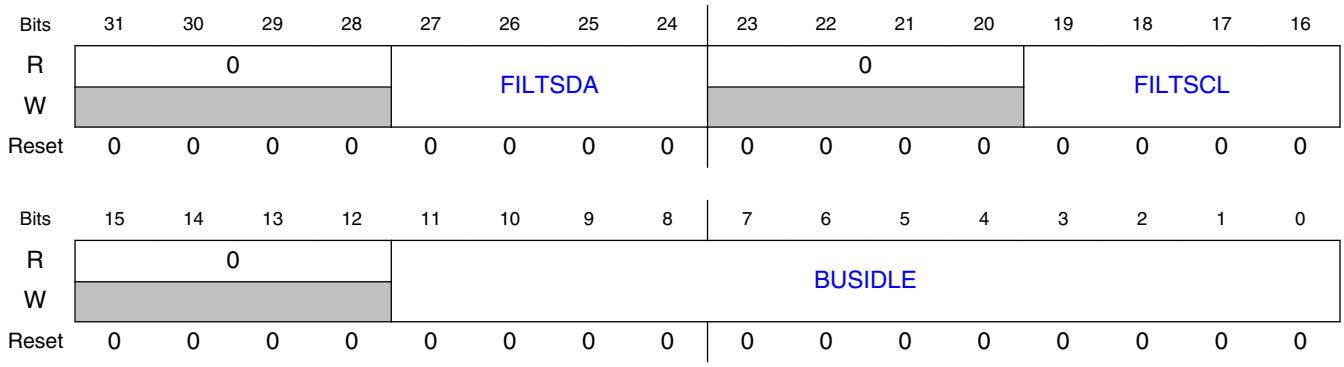
### 55.3.4.10.1 Offset

| Register | Offset |
|---|---|
| MCFGR2 | 28h |

### 55.3.4.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

### 55.3.4.10.3 Diagram




### 55.3.4.10.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C master digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C master digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode |
| 15-12<br>— | Reserved |
| 11-0<br>BUSIDLE | Bus Idle Timeout<br>Configures the bus idle timeout period in clock cycles.<br>• If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition<br>• When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled |

## 55.3.4.11 Master Configuration Register 3 (MCFGR3)

### 55.3.4.11.1 Offset

| Register | Offset |
|---|---|
| MCFGR3 | 2Ch |

### 55.3.4.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

### 55.3.4.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PINLOW | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINLOW | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.11.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-8<br>PINLOW | Pin Low Timeout<br>Configures the pin low timeout flag in clock cycles.<br>• If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then PLTF is set<br>• When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled |
| 7-0<br>— | Reserved |

## 55.3.4.12 Master Data Match Register (MDMR)

### 55.3.4.12.1 Offset

| Register | Offset |
|---|---|
| MDMR | 40h |

### 55.3.4.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

### 55.3.4.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>MATCH1 | Match 1 Value<br>Compared against the received data when receive data match is enabled. |
| 15-8<br>— | Reserved |
| 7-0<br>MATCH0 | Match 0 Value<br>Compared against the received data when receive data match is enabled. |

## 55.3.4.13 Master Clock Configuration Register 0 (MCCR0)

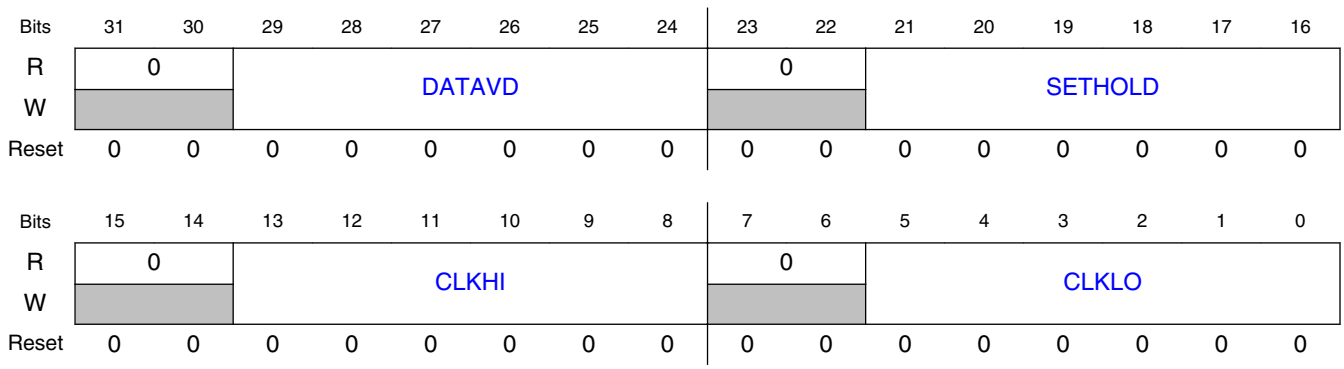
### 55.3.4.13.1 Offset

| Register | Offset |
|---|---|
| MCCR0 | 48h |

### 55.3.4.13.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

### 55.3.4.13.3 Diagram




### 55.3.4.13.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br><br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.4.14 Master Clock Configuration Register 1 (MCCR1)

### 55.3.4.14.1 Offset

| Register | Offset |
|---|---|
| MCCR1 | 50h |

### 55.3.4.14.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by MCCR0), before switching to high speed mode (with timing configured by MCCR1).

### 55.3.4.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DATAVD | | | | | | 0 | | SETHOLD | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | CLKHI | | | | | | 0 | | CLKLO | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.14.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Minimum number of cycles (minus one) that is used as the data hold time for SDA. The Data Valid Delay must be configured to be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Minimum number of cycles (minus one) that is used by the master<br>• as the hold time for a START condition<br>• as the setup and hold time for a repeated START condition<br>• as the setup time for a STOP condition<br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles. |

## 55.3.4.15 Master FIFO Control Register (MFCR)

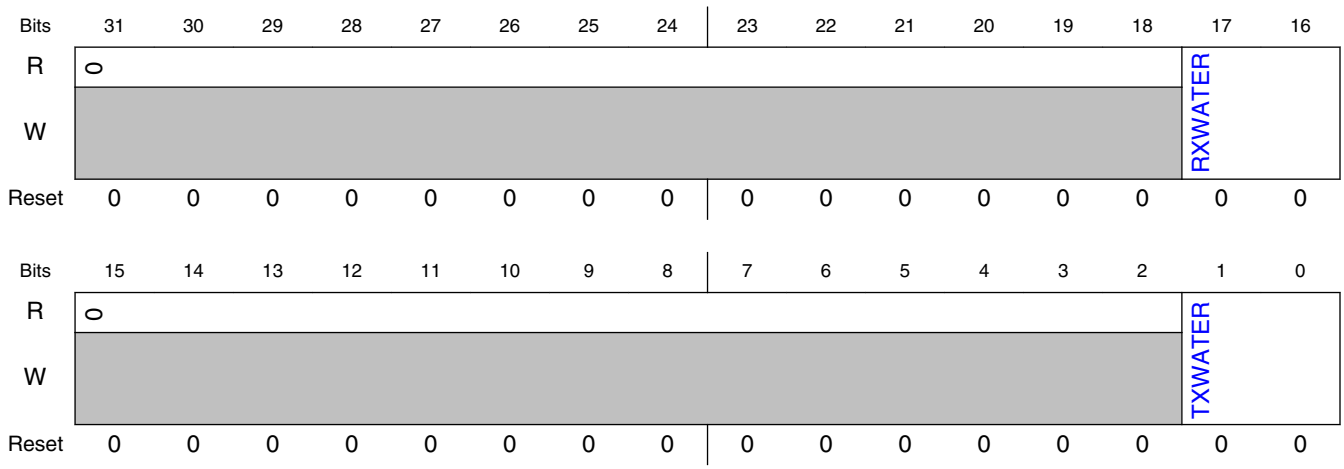
### 55.3.4.15.1 Offset

| Register | Offset |
|---|---|
| MFCR | 58h |

### 55.3.4.15.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

### 55.3.4.15.3 Diagram




### 55.3.4.15.4 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |
| 15-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size will be truncated. |

## 55.3.4.16 Master FIFO Status Register (MFSR)

### 55.3.4.16.1 Offset

| Register | Offset |
|---|---|
| MFSR | 5Ch |

### 55.3.4.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.16.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words in the receive FIFO. |
| 15-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words in the transmit FIFO. |

## 55.3.4.17 Master Transmit Data Register (MTDR)

### 55.3.4.17.1 Offset

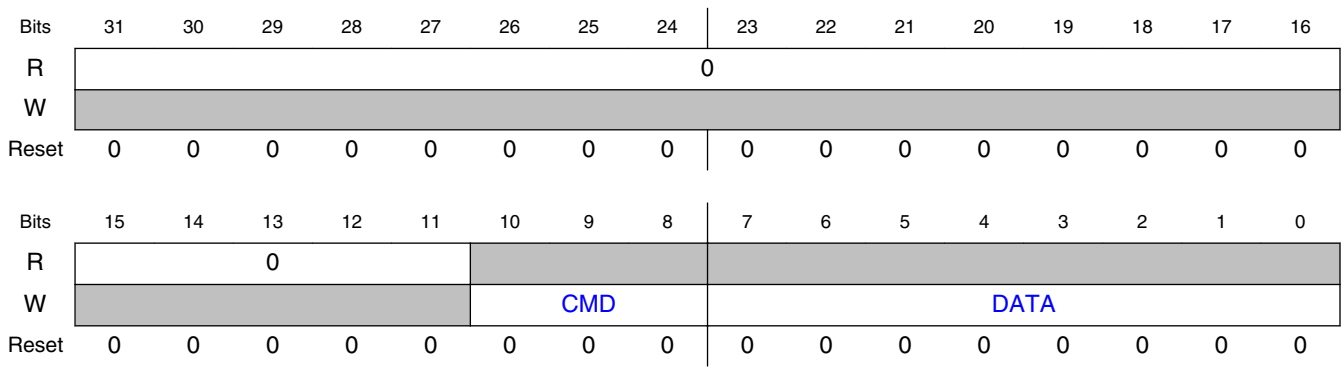
| Register | Offset |
|---|---|
| MTDR | 60h |

### 55.3.4.17.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.

- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit or 32-bit will write both the CMD and DATA fields and increment the FIFO.

### 55.3.4.17.3 Diagram




### 55.3.4.17.4 Fields

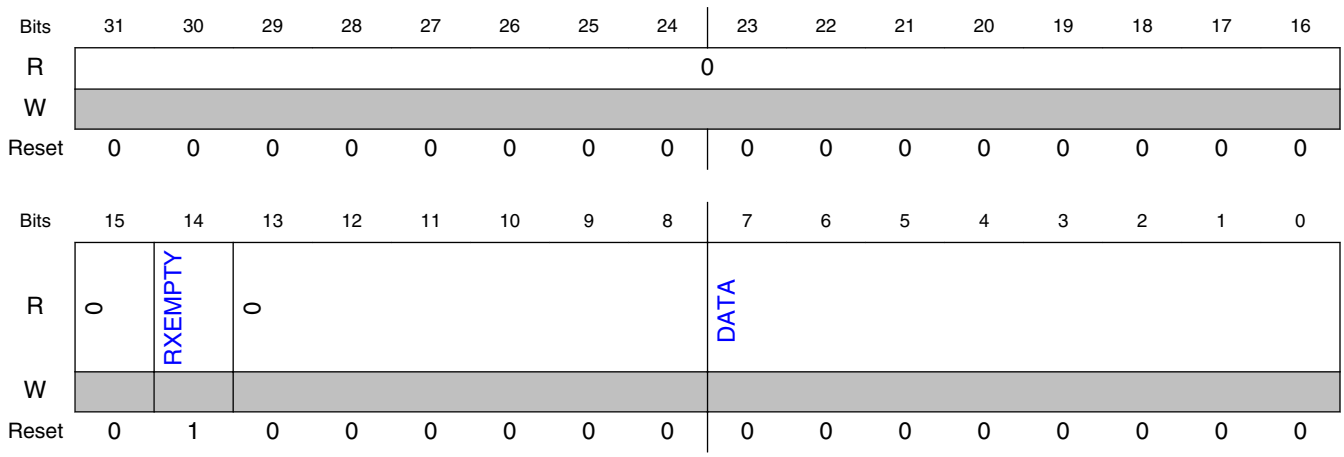
| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>CMD | Command Data<br>000b - Transmit DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes<br>010b - Generate STOP condition<br>011b - Receive and discard (DATA[7:0] + 1) bytes<br>100b - Generate (repeated) START and transmit address in DATA[7:0]<br>101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.<br>110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode<br>111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned. |
| 7-0<br>DATA | Transmit Data<br>Performing an 8-bit write to DATA will zero extend the CMD field. |

## 55.3.4.18 Master Receive Data Register (MRDR)

### 55.3.4.18.1 Offset

| Register | Offset |
|---|---|
| MRDR | 70h |

### 55.3.4.18.2 Diagram




### 55.3.4.18.3 Fields

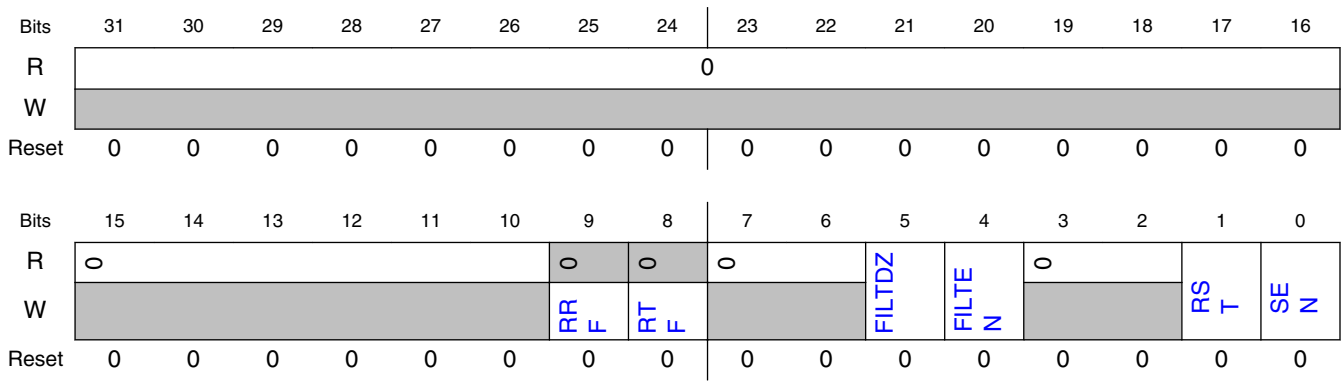
| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14<br>RXEMPTY | RX Empty<br>0b - Receive FIFO is not empty<br>1b - Receive FIFO is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data. |

## 55.3.4.19 Slave Control Register (SCR)

### 55.3.4.19.1 Offset

| Register | Offset |
|---|---|
| SCR | 110h |

### 55.3.4.19.2 Diagram




### 55.3.4.19.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive Data Register is now empty |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit Data Register is now empty |
| 7-6<br>— | Reserved |
| 5<br>FILTDZ | Filter Doze Enable<br>Filter Doze Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Filter remains enabled in Doze mode<br>1b - Filter is disabled in Doze mode |
| 4<br>FILTEN | Filter Enable<br>Filter Enable bit should only be updated when the I2C Slave is disabled.<br>0b - Disable digital filter and output delay counter for slave mode<br>1b - Enable digital filter and output delay counter for slave mode |
| 3-2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RST | 0b - Slave mode logic is not reset<br>1b - Slave mode logic is reset |
| 0<br>SEN | Slave Enable<br>0b - I2C Slave mode is disabled<br>1b - I2C Slave mode is enabled |

## 55.3.4.20 Slave Status Register (SSR)

### 55.3.4.20.1 Offset

| Register | Offset |
|---|---|
| SSR | 114h |

### 55.3.4.20.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | SBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARF | GCF | AM1F | AM0F | FEF | BEF | SDF | RSF | 0 | | | | TAF | AVF | RDF | TDF |
| W | | | | | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.20.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>Indicates if an I2C bus is idle or busy.<br>0b - I2C Bus is idle<br>1b - I2C Bus is busy |
| 24 | Slave Busy Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SBF | Indicates if an I2C slave is idle or busy.<br>0b - I2C Slave is idle<br>1b - I2C Slave is busy |
| 23-16<br>— | Reserved |
| 15<br>SARF | SMBus Alert Response Flag<br>• SMBus Alert Response Flag is cleared by reading the Address Status Register<br>• SMBus Alert Response Flag cannot generate an asynchronous wakeup<br><br>0b - SMBus Alert Response is disabled or not detected<br>1b - SMBus Alert Response is enabled and detected |
| 14<br>GCF | General Call Flag<br>Indicates whether a slave has detected the General Call Address.<br>• General Call Flag is cleared by reading the Address Status Register<br>• General Call Flag cannot generate an asynchronous wakeup<br><br>0b - Slave has not detected the General Call Address or the General Call Address is disabled<br>1b - Slave has detected the General Call Address |
| 13<br>AM1F | Address Match 1 Flag<br>Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by ADDRCFG.<br>• Address Match 1 Flag is cleared by reading the Address Status Register<br>• Address Match 1 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address<br>1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address |
| 12<br>AM0F | Address Match 0 Flag<br>Indicates that the received address has matched the ADDR0 field as configured by ADDRCFG.<br>• Address Match 0 Flag is cleared by reading the Address Status Register<br>• Address Match 0 Flag cannot generate an asynchronous wakeup<br><br>0b - Have not received an ADDR0 matching address<br>1b - Have received an ADDR0 matching address |
| 11<br>FEF | FIFO Error Flag<br>FIFO error flag can only be set when clock stretching is disabled.<br>0b - FIFO underflow or overflow was not detected<br>1b - FIFO underflow or overflow was detected |
| 10<br>BEF | Bit Error Flag<br>Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore the rest of the transfer until the next (repeated) START condition.<br>0b - Slave has not detected a bit error<br>1b - Slave has detected a bit error |
| 9<br>SDF | STOP Detect Flag<br>STOP Detect Flag will set when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte.<br>0b - Slave has not detected a STOP condition<br>1b - Slave has detected a STOP condition |
| 8<br>RSF | Repeated Start Flag |

*Table continues on the next page...*

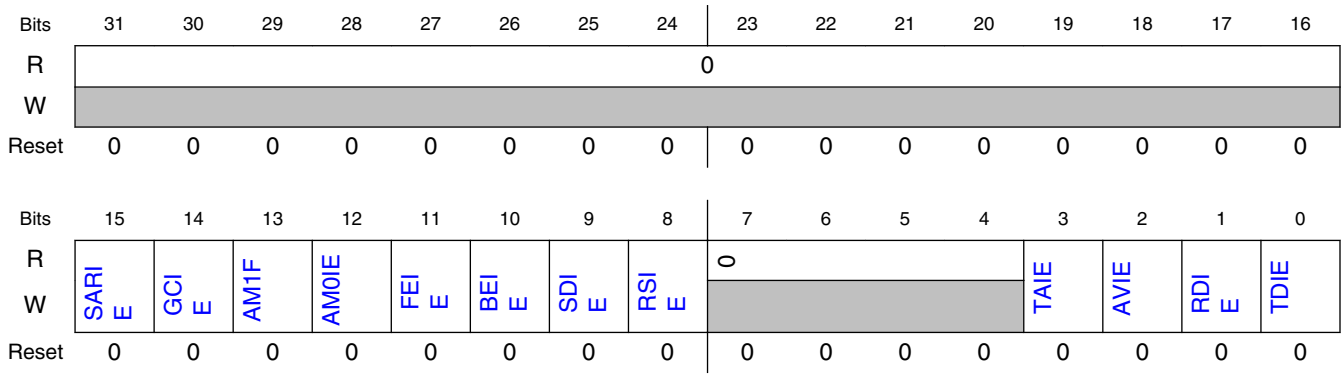
| Field | Function |
|---|---|
| | Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag does not set when the slave first detects a START condition.<br>0b - Slave has not detected a Repeated START condition<br>1b - Slave has detected a Repeated START condition |
| 7-4<br>— | Reserved |
| 3<br>TAF | Transmit ACK Flag<br>Transmit ACK Flag is cleared by writing the Transmit ACK register.<br>0b - Transmit ACK/NACK is not required<br>1b - Transmit ACK/NACK is required |
| 2<br>AVF | Address Valid Flag<br>Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register.<br>0b - Address Status Register is not valid<br>1b - Address Status Register is valid |
| 1<br>RDF | Receive Data Flag<br>Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set.<br>0b - Receive data is not ready<br>1b - Receive data is ready |
| 0<br>TDF | Transmit Data Flag<br>Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared.<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 55.3.4.21 Slave Interrupt Enable Register (SIER)

### 55.3.4.21.1 Offset

| Register | Offset |
|---|---|
| SIER | 118h |

## 55.3.4.21.2 Diagram




## 55.3.4.21.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SARIE | SMBus Alert Response Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 14<br>GCIE | General Call Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 13<br>AM1F | Address Match 1 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>AM0IE | Address Match 0 Interrupt Enable<br>0b - Enabled<br>1b - Disabled |
| 11<br>FEIE | FIFO Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 10<br>BEIE | Bit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>SDIE | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>RSIE | Repeated Start Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-4<br>— | Reserved |
| 3<br>TAIE | Transmit ACK Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

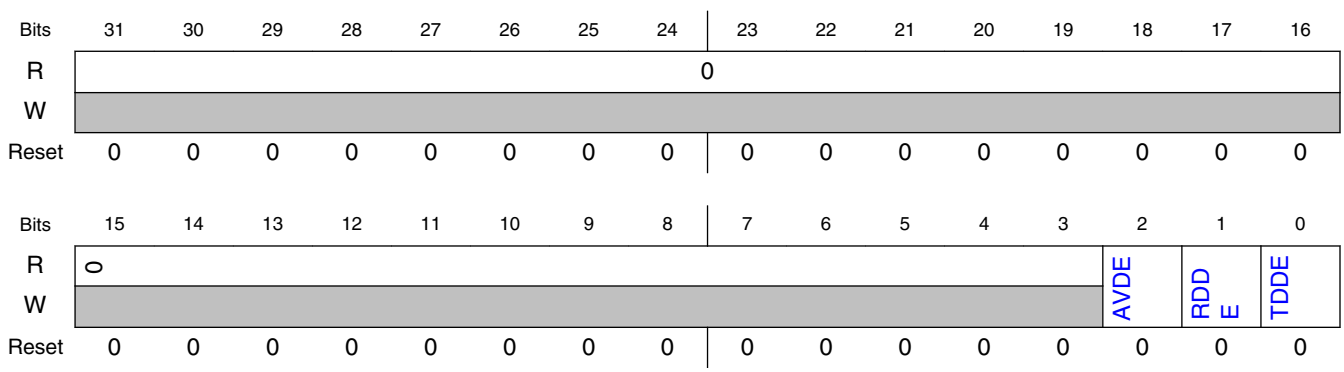
| Field | Function |
|---|---|
| 2<br>AVIE | Address Valid Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 55.3.4.22 Slave DMA Enable Register (SDER)

### 55.3.4.22.1 Offset

| Register | Offset |
|---|---|
| SDER | 11Ch |

### 55.3.4.22.2 Diagram




### 55.3.4.22.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>AVDE | Address Valid DMA Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set Receive Data Configuration (SCFGR1[RXCFG]), to allow the DMA to read the address from the Receive Data Register.<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

## 55.3.4.23 Slave Configuration Register 1 (SCFGR1)

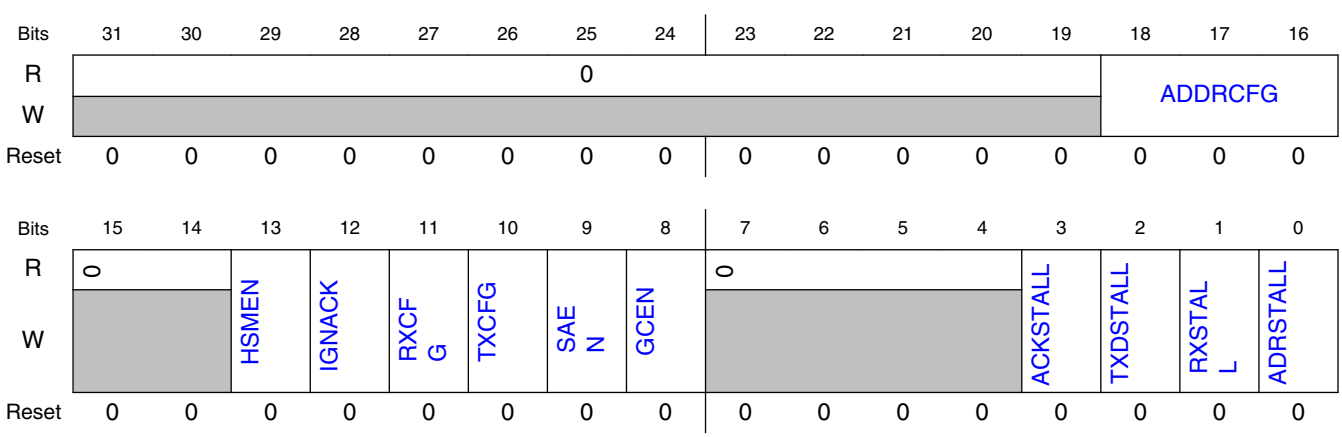
### 55.3.4.23.1 Offset

| Register | Offset |
|---|---|
| SCFGR1 | 124h |

### 55.3.4.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

### 55.3.4.23.3 Diagram



## 55.3.4.23.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>ADDRCFG | Address Configuration<br>Configures the condition that will cause an address to match.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or Address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or Address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or Address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or Address match 1 (7-bit)<br>110b - From Address match 0 (7-bit) to Address match 1 (7-bit)<br>111b - From Address match 0 (10-bit) to Address match 1 (10-bit) |
| 15-14<br>— | Reserved |
| 13<br>HSMEN | High Speed Mode Enable<br>Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, the FILTEN and ACKSTALL bits are ignored until the next STOP condition is detected.<br>0b - Disables detection of HS-mode master code<br>1b - Enables detection of HS-mode master code |
| 12<br>IGNACK | Ignore NACK<br>When Ignore NACK is set, the LPI2C slave will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode.<br>0b - Slave will end transfer when NACK is detected<br>1b - Slave will not end transfer when NACK detected |
| 11<br>RXCFG | Receive Data Configuration<br>0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]).<br>1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]). |
| 10<br>TXCFG | Transmit Flag Configuration<br>The transmit data flag will always assert before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO.<br>• When TXCFG=0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer.<br>• When TXCFG=1, the Transmit Data flag will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.<br>0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty<br>1b - Transmit Data Flag will assert whenever the Transmit Data register is empty |
| 9 | SMBus Alert Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SAEN | 0b - Disables match on SMBus Alert<br>1b - Enables match on SMBus Alert |
| 8<br>GCEN | General Call Enable<br>0b - General Call address is disabled<br>1b - General Call address is enabled |
| 7-4<br>— | Reserved |
| 3<br>ACKSTALL | ACK SCL Stall<br>Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Transmit ACK Register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode.<br>When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]).<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 2<br>TXDSTALL | TX Data SCL Stall<br>Enables SCL clock stretching when the Transmit Data flag (SSR[TDF]) is set during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 1<br>RXSTALL | RX SCL Stall<br>Enables SCL clock stretching when the Receive Data flag (SSR[RDF]) is set during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |
| 0<br>ADRSTALL | Address SCL Stall<br>Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode.<br>0b - Clock stretching is disabled<br>1b - Clock stretching is enabled |

## 55.3.4.24 Slave Configuration Register 2 (SCFGR2)

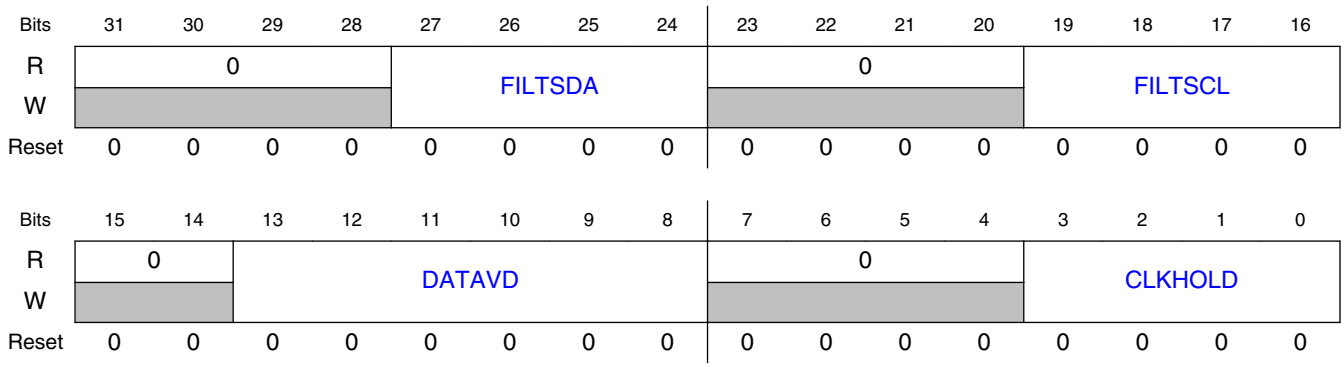
### 55.3.4.24.1 Offset

| Register | Offset |
|---|---|
| SCFGR2 | 128h |

### 55.3.4.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

## 55.3.4.24.3 Diagram




## 55.3.4.24.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C slave digital glitch filters for SDA input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C slave digital glitch filters for SCL input.<br>• A configuration of 0 will disable the glitch filter<br>• Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored<br>• The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period<br>• The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode |
| 15-14<br>— | Reserved |
| 13-8<br>DATAVD | Data Valid Delay<br>Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles.<br>• The data valid delay must be configured to be less than the minimum SCL low period<br>• The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode |
| 7-4<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3-0<br>CLKHOLD | Clock Hold Time<br>Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled.<br>• The minimum hold time is equal to CLKHOLD+3 cycles<br>• The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode |

## 55.3.4.25 Slave Address Match Register (SAMR)

### 55.3.4.25.1 Offset

| Register | Offset |
|---|---|
| SAMR | 140h |

### 55.3.4.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

### 55.3.4.25.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR1 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR0 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.25.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-17<br>ADDR1 | Address 1 Value<br>Compared against the received address to detect the Slave Address. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17]<br>• In 7-bit mode, the address is compared to ADDR1[23:17] |
| 16-11<br>— | Reserved |
| 10-1<br>ADDR0 | Address 0 Value<br>Compared against the received address to detect the Slave Address.<br>• In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1]<br>• In 7-bit mode, the address is compared to ADDR0[7:1]<br>• |
| 0<br>— | Reserved |

## 55.3.4.26 Slave Address Status Register (SASR)

### 55.3.4.26.1 Offset

| Register | Offset |
|---|---|
| SASR | 150h |

### 55.3.4.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ANV | 0 | | | RADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 55.3.4.26.3 Fields

| Field | Function |
|---|---|
| 31-15 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 14<br>ANV | Address Not Valid<br>0b - Received Address (RADDR) is valid<br>1b - Received Address (RADDR) is not valid |
| 13-11<br>— | Reserved |
| 10-0<br>RADDR | Received Address<br>The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register.<br>• In 7-bit mode, the address byte is store in RADDR[7:0]<br>• In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0] |

## 55.3.4.27 Slave Transmit ACK Register (STAR)

### 55.3.4.27.1 Offset

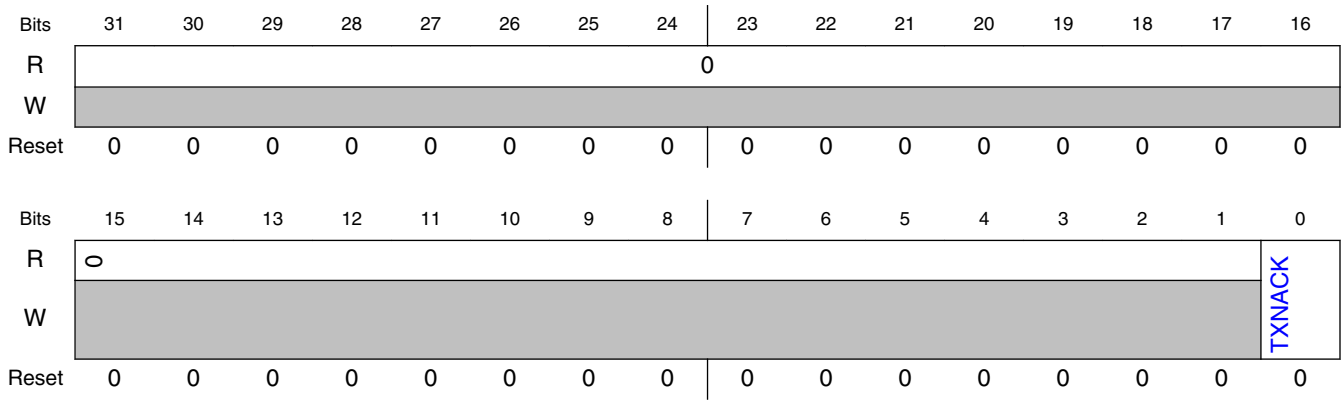
| Register | Offset |
|---|---|
| STAR | 154h |

### 55.3.4.27.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR1[ACKSTALL].

- The ACKSTALL bit will enable clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).

### 55.3.4.27.3 Diagram




### 55.3.4.27.4 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TXNACK | Transmit NACK<br>After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK.<br>• When ACKSTALL is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that will stall the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK).<br>• To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle.<br><br>0b - Write a Transmit ACK for each received word<br>1b - Write a Transmit NACK for each received word |

## 55.3.4.28 Slave Transmit Data Register (STDR)

### 55.3.4.28.1 Offset

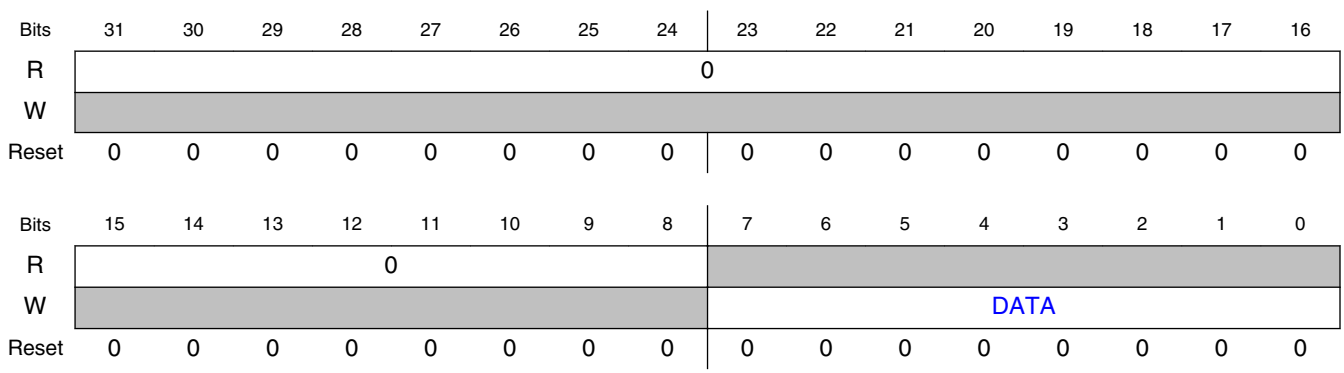
| Register | Offset |
|---|---|
| STDR | 160h |

### 55.3.4.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

### 55.3.4.28.3 Diagram




### 55.3.4.28.4 Fields

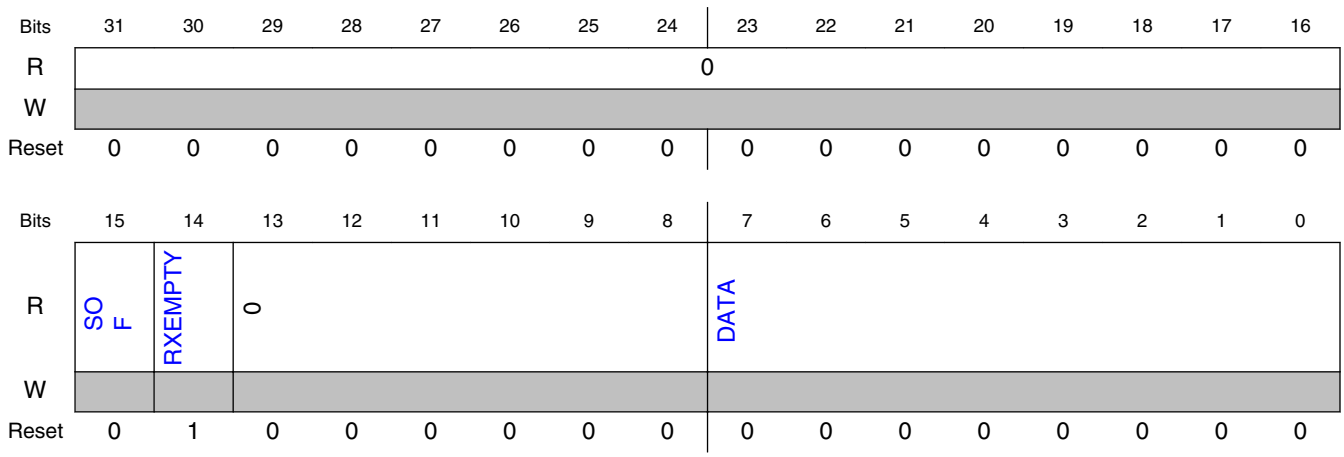
| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>DATA | Transmit Data<br>Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register |

## 55.3.4.29 Slave Receive Data Register (SRDR)

### 55.3.4.29.1 Offset

| Register | Offset |
|---|---|
| SRDR | 170h |

### 55.3.4.29.2 Diagram




### 55.3.4.29.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SOF | Start Of Frame<br>0b - Indicates this is not the first data word since a (repeated) START or STOP condition<br>1b - Indicates this is the first data word since a (repeated) START or STOP condition |
| 14<br>RXEMPTY | RX Empty<br>0b - The Receive Data Register is not empty<br>1b - The Receive Data Register is empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Reading the Slave Receive Data Register returns the data received by the I2C slave |

## 55.4 Functional description

## 55.4.1 Clocking and Resets

**Table 55-7. Clocks**

| | |
|---|---|
| LPI2C Functional clock | The LPI2C functional clock is asynchronous to the bus clock and can remain enabled in low power modes to support I2C bus transfers by the LPI2C master. The functional clock is also used by the LPI2C slave to support digital filter and data hold time configurations. The LPI2C master divides the functional clock by a prescaler and the resulting frequency must be at least 8 times faster than the I2C bus bandwidth. |
| External clock | The LPI2C slave logic is clocked directly from the external pins SCL and SDA (or SCLS and SDAS if master and slave are implemented on separate pins). This allows the LPI2C slave to remain operational, even when the LPI2C functional clock is disabled.<br>**NOTE:** The LPI2C slave digital filter must be disabled if the LPI2C functional clock is disabled, and this can affect compliance with some of the timing parameters of the I2C specification, such as the data hold time. |
| Bus clock | The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPI2C master and slave registers. |

**Table 55-8. Resets**

| | |
|---|---|
| Chip reset | The logic and registers for the LPI2C master and slave are reset to their default state on a chip reset. |
| Software reset | • The LPI2C master implements a software reset bit in its Control Register. The MCR[RST] will reset all master logic and registers to their default state, except for the MCR itself.<br>• The LPI2C slave implements a software reset bit in its Control Register. The SCR[RST] will reset all slave logic and registers to their default state, except for the SCR itself. |
| FIFO reset | • The LPI2C master implements write-only control bits that reset the transmit FIFO (MCR[RTF]) and receive FIFO (MCR[RRF]). After a FIFO is reset, that FIFO is empty.<br>• The LPI2C slave implements write-only control bits that reset the transmit data register (SCR[RTF] and receive data register (SCR[RRF]). After a data register is reset, that data register is empty. |

## 55.4.2 Master Mode

The LPI2C master logic operates independently from the slave logic to perform all master mode transfers on the I2C bus.

### 55.4.2.1 Transmit and Command FIFO commands

The transmit FIFO stores command data to initiate the various I2C operations. The following operations can be initiated through commands in the transmit FIFO:

- START or Repeated START condition with address byte and expecting ACK or NACK.
- Transmit data (this is the default for zero extended byte writes to the transmit FIFO).

- Receive 1-256 bytes of data (can also be configured to discard receive data and not store in receive FIFO).
- STOP condition (can also be configured to send STOP condition when transmit FIFO is empty).

Multiple transmit and receive commands can be inserted between the START condition and STOP conditon; transmit and receive commands must not be interleaved (to comply with the I2C specification). The receive data command and the receive data and discard commands can be interleaved, to ensure that only the desired received data is stored in the receive FIFO (or compared with the data match logic).

The LPI2C master will automatically transmit a NACK on the last byte of a receive data command unless the next command in the FIFO is also a receive data command. A NACK is also automatically transmitted if the transmit FIFO is empty when a receive data command completes.

The LPI2C master supports 10-bit addressing through a (repeated) START condition, followed by a transmit data byte containing the second address byte, followed by any number of data bytes with the master-transmit data.

A START or Repeated START condition that is expecting a NACK (for example,HS-mode master code) must be followed by a STOP or (repeated) START condition.

### 55.4.2.2 Master operations

Whenever the LPI2C is enabled, it monitors the I2C bus to detect when the I2C bus is idle (MSR[BBF]). The I2C bus is no longer considered idle if either SCL or SDA are low, and the I2C bus becomes idle if a STOP condition is detected or if a bus idle timeout is detected (as configured by MCFGR2[BUSIDLE]). After the I2C bus is idle, the transmit FIFO is not empty, and the host request is either asserted or disabled, then the LPI2C master will initiate a transfer on the I2C bus. This involves the following steps:

- Wait the bus idle time equal to (MCCR0[CLKLO] + 1) multiplied by the prescaler (MCFGR1[PRESCALE]).
- Transmit a START condition and address byte using the timing configuration in the Master Clock Configuration Register 0 (MCCR0); if a high speed mode transfer is configured, then the timing configuration from Master Clock Configuration Register 1 (MCCR1) is used instead.
- Perform master-transmit or master-receive transfers, as configured by the transmit FIFO.

- Transmit NACK on the last byte of a master-receive transfer, unless the next command in the transmit FIFO is also a receive data command and the transmit FIFO is not empty.
- Transmit a Repeated START or STOP condition as configured by the transmit FIFO and/or MCFGR1[AUTOSTOP]. A repeated START can change which timing configuration register is used.

When the LPI2C master is disabled (either due to MCR[MEN] being clear or automatically due to mode entry), the LPI2C will continue to empty the transmit FIFO until a STOP condition is transmitted. However, the LPI2C will no longer stall the I2C bus waiting for the transmit or receive FIFO, and after the transmit FIFO is empty, the LPI2C will generate a STOP condition automatically.

The LPI2C master can stall the I2C bus under certain conditions; this will result in SCL pulled low continuously on the first bit of a byte, until the condition is removed:

- LPI2C master is enabled and busy, the transmit FIFO is empty, and MCFGR1[AUTOSTOP] is clear.
- LPI2C master is enabled and receiving data, receive data is not being discarded (due to command or receive data match), and the receive FIFO is full.

#### 55.4.2.3 Receive FIFO and Data Matching

The receive FIFO is used to store receive data during master-receiver transfers. Receive data can also be configured to discard receive data instead of storing in the receive FIFO; this is configured by the command word in the transmit FIFO.

Receive data supports a receive data match function that can match received data against one of two bytes or against a masked data byte. The data match function can also be configured to compare only the first one or two received data words since the last (repeated) START condition. Receive data that is already discarded due to the command word cannot cause the data match to set, and will delay the match on the first received data word until after the discarded data is received. The receiver match function can also be configured to discard all receive data until a data match is detected, using the MCFGR0[RDMO] control bit. When clearing the MCFGR0[RDMO] control bit following a data match, clear MCFGR0[RDMO] before clearing MSR[DMF], to allow all subsequent data to be received.

### 55.4.2.4 Timing Parameters

The following timing parameters can be configured by the LPI2C master. Parameters are configured separately for high speed mode (MCCR1) and other modes (MCCR0). This allows the high speed mode master code to be sent using the regular timing parameters, and then switch to the high speed mode timing (following a repeated START) until the next STOP condition.

The LPI2C master timing parameters in LPI2C functional clock cycles are configured as follows. They must be configured to meet the I2C timing specification for the required mode.

**Table 55-9. Timing Parameters**

| I2C Specification Timing Parameter | I2C Specification Timing Symbol | LPI2C Timing Parameter (LPI2C functional clock cycles) |
|---|---|---|
| hold time (repeated) START condition | tHD:STA | (SETHOLD +1) x (2 ^ PRESCALE) |
| LOW period of the SCL clock | tLOW | (CLKLO + 1) x (2 ^ PRESCALE) |
| HIGH period of the SCL clock | tHIGH | (CLKHI + 1 + SCL_LATENCY) x (2 ^ PRESCALE) |
| setup time for a repeated START condition or STOP condition | tSU:STA, tSU:STO | (SETHOLD + 1 + SCL_LATENCY) x (2 ^ PRESCALE) |
| data hold time | tHD:DAT | (DATAVD + 1) x (2 ^ PRESCALE) |
| data setup time | tSU:DAT | (SDA_LATENCY + 1) x (2 ^ PRESCALE) |
| bus free time between a STOP and START condition | tBUF | (CLKLO + 1 + SDA_LATENCY) x (2 ^ PRESCALE) |
| data valid time, data valid acknowledge time | tVD:DAT, tVD:ACK | (DATAVD + 1) x (2 ^ PRESCALE) |

The latency parameters are defined in the following table, these parameters assume the risetime is less than one LPI2C functional clock cycle. The risetime depends on a number of factors, including the I2C bus loading and external pull-up resistor sizing. A larger risetime will increase the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

**Table 55-10. Synchronization Latency**

| Timing Parameter | Timing Definition |
|---|---|
| SCL_LATENCY | ROUNDDOWN ( (2 + FILTSCL + SCL_RISETIME) / (2 ^ PRESCALE) ) |
| SDA_LATENCY | ROUNDDOWN ( (2 + FILTSDA + SDA_RISETIME) / (2 ^ PRESCALE) ) |

The following timing restrictions must be enforced to avoid unexpected START or STOP conditions on the I2C bus, or to avoid unexpected START or STOP conditions detected by the LPI2C master. The timing restrictions can be summarized as **SDA cannot change when SCL is high outside of a transmitted (repeated) START or STOP condition**.

**Table 55-11. LPI2C Timing Parameter Restrictions**

| Timing Parameter | Minimum | Maximum | Comment |
|---|---|---|---|
| CLKLO | 0x03 | - | Also: CLKLO x (2 ^ PRESCALE) > SCL_LATENCY |
| CLKHI | 0x01 | - | Configure CLKHI to meet the duty cycle requirements in the I2C specification |
| SETHOLD | 0x02 | - | |
| DATAVD | 0x01 | CLKLO - SDA_LATENCY - 1 | Configure DATAVD to meet the data hold requirement in the I2C specification |
| FILTSCL | 0x00 | [CLKLO × (2 ^ PRESCALE)] - 3 | FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE) |
| FILTSDA | FILTSCL | [CLKLO × (2 ^ PRESCALE)] - 3 | Configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board level skew |
| BUSIDLE | (CLKLO+SETHOLD+2) × 2 | - | Must also be greater than (CLKHI+1) |

The timing parameters must be configured to meet the requirements of the I2C specification; this will depend on the mode being supported and the LPI2C functional clock frequency. When switching between two modes using the different clock configuration registers (for example, Fast and HS-mode), the PRESCALE factor must remain constant between the modes. Some example timing configurations are provided below.

**Table 55-12. LPI2C Example Timing Configurations**

| I2C Mode | Clock Frequency | Baud Rate | PRESCALE | FILTSCL / FILTSDA | SETHOLD | CLKLO | CLKHI | DATAVD |
|---|---|---|---|---|---|---|---|---|
| Fast | 8 MHz | 400 kbps | 0x0 | 0x0/0x0 | 0x04 | 0x0B | 0x05 | 0x02 |
| Fast+ | 8 MHz | 1 Mbps | 0x0 | 0x0/0x0 | 0x02 | 0x03 | 0x01 | 0x01 |
| Fast | 48 MHz | 400 kbps | 0x0 | 0x1/0x1 | 0x1D | 0x3E | 0x35 | 0x0F |
| Fast | 48 MHz | 400 kbps | 0x2 | 0x1/0x1 | 0x07 | 0x11 | 0x0B | 0x03 |
| Fast+ | 48 MHz | 1 Mbps | 0x2 | 0x1/0x1 | 0x03 | 0x06 | 0x04 | 0x04 |
| HS-mode | 48 MHz | 3.2 Mbps | 0x0 | 0x0/0x0 | 0x07 | 0x08 | 0x03 | 0x01 |
| Fast | 60 MHz | 400 kbps | 0x1 | 0x2/0x2 | 0x11 | 0x28 | 0x1F | 0x08 |
| Fast+ | 60 MHz | 1 Mbps | 0x1 | 0x2/0x2 | 0x07 | 0x0F | 0x0B | 0x01 |
| HS-mode | 60 MHz | 3.33 Mbps | 0x1 | 0x0/0x0 | 0x04 | 0x04 | 0x02 | 0x01 |

### 55.4.2.5 Error Conditions

The LPI2C master will monitor for errors while it is active, the following conditions will generate an error flag and block a new START condition from being sent, until the flag is cleared by software:

- A START or STOP condition is detected and is not generated by the LPI2C master (sets MSR[ALF]).
- Transmitting data on SDA and different values are being received (sets MSR[ALF]).
- NACK is detected when transmitting data, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- NACK is detected and is expecting ACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- ACK is detected and is expecting NACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- Transmit FIFO is requesting to transmit or receive data without a START condition (sets MSR[FEF]).
- SCL (or SDA if MCFGR1[TIMECFG] is set) is low for (MCFGR2[TIMELOW] * 256) prescaler cycles without a pin transition (sets MSR[PLTF]).

Software must respond to the MSR[PTLF] flag to terminate the existing command either cleanly (by clearing MCR[MEN]), or abruptly (by setting MCR[SWRST]).

The MCFGR2[BUSIDLE] field can be used to force the I2C bus to be considered idle when SCL and SDA remain high for (BUSIDLE+1) prescaler cycles. The I2C bus is normally considered idle when the LPI2C master is first enabled, but when BUSIDLE is configured greater than zero then SCL and/or SDA must be high for (BUSIDLE+1) prescaler cycles before the I2C bus is first considered idle.

### 55.4.2.6 Pin Configuration

- **Open-drain support:** The LPI2C master defaults to open-drain configuration of the SDA and SCL pins. Support for true open drain depends on the specific device, and requires the pins where LPI2C pins are muxed to support true open drain.
- **High Speed mode support:** Support for high speed mode also depends on the specific device, and requires the SCL pin to support the current source pull-up required in the I2C specification.
- **Ultra-Fast mode support:** The LPI2C master also supports the output-only push-pull function required for I2C ultra-fast mode using the SDA and SCL pins. Support for ultra-fast mode also requires the IGNACK bit to be set.
- **Push-pull 2-wire support:** A push-pull 2-wire configuration is also available to the LPI2C master that may support a partial high speed mode, if the LPI2C is the only

master and all I2C pins on the bus are at the same voltage. This will configure the SCL pin as push-pull for every clock except the 9th clock pulse, to allow high speed mode compatible slaves to perform clock stretching. In this mode, the SDA pin is tristated for master-receive data bits and master-transmit ACK/NACK bits, and is configured as push-pull at other times. To avoid the risk of contention when SDA is push-pull, the pin can be configured for open-drain operation, as part of the device-specific configuration.

- **Push-pull 4-wire support:** The push-pull 4-wire configuration separates the SCL input data and output data into separate pins, and separates the SDA input data and output data into separate pins. The SCL/SDA pins are used for input data; the SCLS/SDAS pins are used for output data, with configurable polarity. This simplifies external connections when connecting the LPI2C to the I2C bus through external level shifters or discrete components. When using this 4-wire configuration, the LPI2C master logic and LPI2C slave logic are not able to connect to separate I2C buses.

## 55.4.3 Slave Mode

To perform all slave mode transfers on the I2C bus, the LPI2C slave logic operates independently from the LPI2C master logic.

### 55.4.3.1 Address Matching

The LPI2C slave can be configured:

- to match one of two addresses, using either 7-bit or 10-bit addressing modes for each address
- to match a range of addresses in either 7-bit or 10-bit addressing modes
- to match the General Call Address, and generate appropriate flags
- to match the SMBus Alert Address, and generate appropriate flags
- to detect the high speed mode master code, and to disable the digital filters and output valid delay time until the next STOP condition is detected

After a valid address is matched, the LPI2C slave will automatically perform slave-transmit or slave-receive transfers until:

- a NACK is detected (unless IGNACK is set)
- a bit error is detected (the LPI2C slave is driving SDA, but a different value is sampled)
- a (repeated) START or STOP condition is detected

### 55.4.3.2 Transmit and Receive Data

- The Transmit and Receive Data registers are double-buffered and only update during a slave-transmit and slave-receive transfer, respectively.
- The slave address *that was received* can be configured to be read from either the Receive Data register (for example, when using DMA to transfer data), or from the Address Status register.
- The Transmit Data register can be configured to only request data after a slave-transmit transfer is detected, or to request new data whenever the Transmit Data register is empty.
- The Transmit Data register should only be written when the Transmit Data flag is set.
- The Receive Data register should only be read when the Received Data flag is set (or the Address Valid flag is set and RXCFG=1).
- The Address Status register should only be read when the Address Valid flag is set.

### 55.4.3.3 Clock Stretching

The LPI2C slave supports many configurable options for when clock stretching is performed. The following conditions can be configured to perform clock stretching:

- During the 9th clock pulse of the address byte and the Address Valid flag is set.
- During the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set.
- During the 9th clock pulse of a slave-receive transfer and the Receive Data flag is set.
- During the 8th clock pulse of an address byte or a slave-receive transfer and the Transmit ACK flag is set. In high speed mode, this is disabled.
- Clock stretching can also be extended for CLKHOLD cycles, to allow additional setup time to sample the SDA pin externally. In high speed mode, this is disabled.

Unless extended by the CLKHOLD configuration, clock stretching will extend for one peripheral bus clock cycle after SDA updates when clock stretching is enabled.

### 55.4.3.4 Timing Parameters

The LPI2C slave can configure the following timing parameters. These parameters are disabled when SCR[FILTEN] is clear, when SCR[FILTDZ] is set in Doze mode, and when LPI2C slave detects high speed mode. When disabled, the LPI2C slave is clocked directly from the I2C bus, and may not satisfy all timing requirements of the I2C specification (such as SDA minimum hold time in Standard/Fast mode).

- SDA data valid time from SCL negation to SDA update
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally
- SCL glitch filter time
- SDA glitch filter time

The LPI2C slave imposes the following restrictions on the timing parameters.

- FILTSDA must be configured to greater than or equal to FILTSCL (unless compensating for board level skew between SDA and SCL).
- DATAVD must be configured less than the minimum SCL low period.

#### 55.4.3.5 Error Conditions

The LPI2C slave can detect the following error conditions:

- Bit error flag will set when the LPI2C slave is driving SDA, but samples a different value than what is expected.
- FIFO error flag will set due to a transmit data underrun or a receive data overrun. To eliminate the possibility of underrun and overrun occurring, enable clock stretching.
- FIFO error flag will also set due to an address overrun when RXCFG is set, otherwise an address overrun is not flagged. To eliminate the possibility of overrun occurring, enable clock stretching.

The LPI2C slave does not implement a timeout due to SCL and/or SDA being stuck low. If this detection is required, then the LPI2C master logic should be used and so software can reset the LPI2C slave when this condition is detected.

### 55.4.4 Interrupts and DMA Requests

Depending on the specific device, interrupts and DMA requests can be combined in some ways:

- The LPI2C master and slave interrupts may be combined
- The LPI2C master and slave transmit DMA requests may be combined
- The LPI2C master and slave receive DMA requests may be combined

#### 55.4.4.1 Master mode

The next table lists the master mode sources that can generate LPI2C master interrupts and LPI2C master transmit/receive DMA requests.

**Table 55-13. Master Interrupts and DMA Requests**

| Master Status Register (MSR) | | Description | Can generate | | |
|---|---|---|---|---|---|
| Flag | Name | | Interrupt? | DMA Request? | Low Power Wakeup? |
| TDF | Transmit Data Flag | Data can be written to Transmit FIFO, as configured by the Transmit FIFO Watermark MFCR[TXWATER] | Y | TX | Y |
| RDF | Receive Data Flag | Data can be read from the Receive FIFO, as configured by the Receive FIFO Watermark MFCR[RXWATER] | Y | RX | Y |
| EPF | End Packet Flag | Master has transmitted a Repeated START or STOP condition | Y | N | Y |
| SDF | STOP Detect Flag | Master has transmitted a STOP condition | Y | N | Y |
| NDF | NACK Detect Flag | • During an address byte, the master expected an ACK but detected a NACK<br>• During an address byte, the master expected a NACK but detected an ACK<br>• During a master-transmitter data byte, the master detected a NACK | Y | N | Y |
| ALF | Arbitration Lost Flag | • The master lost arbitration due to a START/STOP condition detected at the wrong time<br>• Or the master was transmitting data but received different data than the data that was transmitted | Y | N | Y |
| FEF | FIFO Error Flag | The master is expecting a START condition in the Command FIFO, but the next entry in the Command FIFO is not a START condition | Y | N | Y |
| PLTF | Pin Low Timeout Flag | Pin low timeout is enabled and SCL (or SDA if configured) is low for longer than the configured timeout | Y | N | Y |
| DMF | Data Match Flag | The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry | Y | N | Y |
| MBF | Master Busy Flag | LPI2C master is busy transmitting/receiving data | N | N | N |
| BBF | Bus Busy Flag | LPI2C master is enabled and activity is detected on I2C bus, but a STOP condition has not been detected and a bus idle timeout (if enabled) has not occurred. | N | N | N |

### 55.4.4.2 Slave mode

The next table lists the slave mode sources that can generate LPI2C slave interrupts and the LPI2C slave transmit/receive DMA requests.

## Table 55-14. Slave Interrupts and DMA Requests

| Slave Status Register (SSR) | | Description | Can generate | | |
|---|---|---|---|---|---|
| Flag | Name | | Interrupt? | DMA Request? | Low Power Wakeup? |
| TDF | Transmit Data Flag | Data can be written to the Slave Transmit Data Register (STDR) | Y | TX | Y |
| RDF | Receive Data Flag | Data can be read from the Slave Receive Data Register (SRDR) | Y | RX | Y |
| AVF | Address Valid Flag | Address can be read from the Slave Address Status Register (SASR) | Y | RX | Y |
| TAF | Transmit ACK Flag | ACK/NACK can be written to the Slave Transmit ACK Register (STAR) | Y | N | Y |
| RSF | Repeated Start Flag | Slave has detected an address match followed by a Repeated START condition | Y | N | Y |
| SDF | STOP Detect Flag | Slave has detected an address match followed by a STOP condition | Y | N | Y |
| BEF | Bit Error Flag | Slave was transmitting data, but received different data than what was transmitted | Y | N | Y |
| FEF | FIFO Error Flag | • Transmit data underrun<br>• Receive data overrun<br>• Address status overrun (when Receive Data Configuration SCFGR1[RXCFG] = 1, )<br><br>FEF flag can only set when clock stretching is disabled. | Y | N | Y |
| AM0F | Address Match 0 Flag | Slave detected an address match with SAMR[ADDR0] field | Y | N | N |
| AM1F | Address Match 1 Flag | Slave detected an address match with SAMR[ADDR1] field or using an address range | Y | N | N |
| GCF | General Call Flag | Slave detected an address match with the General Call address | Y | N | N |
| SARF | SMBus Alert Response Flag | Slave detected an address match with the SMBus Alert address | Y | N | N |
| SBF | Slave Busy Flag | LPI2C slave is busy receiving an address byte or is transmitting/receiving data | N | N | N |
| BBF | Bus Busy Flag | LPI2C slave is enabled and a START condition is detected on I2C bus, but a STOP condition has not been detected | N | N | N |

### 55.4.5 Peripheral Triggers

The connection of the LPI2C peripheral triggers to other peripherals depend upon the specific device being used.

**Table 55-15. LPI2C Triggers**

| Trigger | Description |
|---|---|
| Master Output Trigger | The LPI2C master generates an output trigger that can be connected to other peripherals on the device. The master output trigger asserts on both a Repeated START or STOP condition, and the master output trigger remains asserted for one cycle of the LPI2C functional clock divided by the prescaler. |
| Slave Output Trigger | The LPI2C slave generates an output trigger that can be connected to other peripherals on the device. The slave output trigger asserts on both a Repeated START or STOP condition that occurs after a slave address match, and the slave output trigger remains asserted until the next slave SCL pin negation. |
| Input Trigger | To control the start of a LPI2C bus transfer, the LPI2C input trigger can be selected instead of the HREQ input. The input trigger is synchronized and to be detected, the input trigger must assert for at least 2 cycles of the LPI2C functional clock divided by the PRESCALE configuration. When the LPI2C is busy, the HREQ input (and therefore the input trigger) is ignored. |

# Chapter 56
# Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)

## 56.1 Chip-specific LPUART information

### 56.1.1 LPUART overview

The LPUART module supports basic UART with DMA interface function and x4 to x32 oversampling of baud-rate.

The module can remain functional in Stop and VLPS mode provided the clock it is using remains enabled.

This module supports LIN slave operation.

### 56.1.2 LPUART Instantiation

This device contains 4 LPUART modules. This section describes how each module is configured on this device.

**Table 56-1. LPUART instantiations**

| Feature | Receive Buffer Size |
|---|---|
| LPUARTs instantiated | 4 |
| LPUART0 FIFO (RX & TX) | 8 Byte |
| LPUART1 FIFO (RX & TX) | 8 Byte |
| LPUART2 FIFO (RX & TX) | 8 Byte |
| LPUART3 FIFO (RX & TX) | 8 Byte |

## 56.2 Introduction

### 56.2.1 Features

Features of the LPUART module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Transmit and receive baud rate can operate asynchronous to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency
  - Supports operation in Stop modes
- Interrupt, DMA or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 7-bit, 8-bit, 9-bit or 10-bit character length
- Programmable 1-bit or 2-bit stop bits
- Three receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13-bit break character generation / 11-bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals

- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- Independent FIFO structure for transmit and receive
  - Separate configurable watermark for receive and transmit requests
  - Option for receiver to assert request after a configurable number of idle characters if receive FIFO is not empty

## 56.2.2 Modes of operation

### 56.2.2.1 Stop mode

The LPUART will remain functional during Stop mode, provided the CTRL[DOZEEN] bit is clear and the asynchronous transmit and receive clock remain enabled. The LPUART can generate an interrupt or DMA request to cause a wakeup from Stop mode.

If the LPUART is disabled in Stop mode, then it can generate a wakeup via the STAT[RXEDGIF] flag if the receiver detects an active edge.

### 56.2.2.2 Wait mode

The LPUART can be configured to Stop in Wait modes, when the CTRL[DOZEEN] bit is set. The transmitter and receiver will finish transmitting/receiving the current word.

### 56.2.2.3 Debug mode

The LPUART remains functional in debug mode.

## 56.2.3 Signal Descriptions

| Signal | Description | I/O |
|---|---|---|
| TXD | Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data. | I/O |
| RXD | Receive data. | I |
| CTS_B | Clear to send. | I |
| RTS_B | Request to send. | O |

## 56.2.4 Block diagram

The following figure shows the transmitter portion of the LPUART.

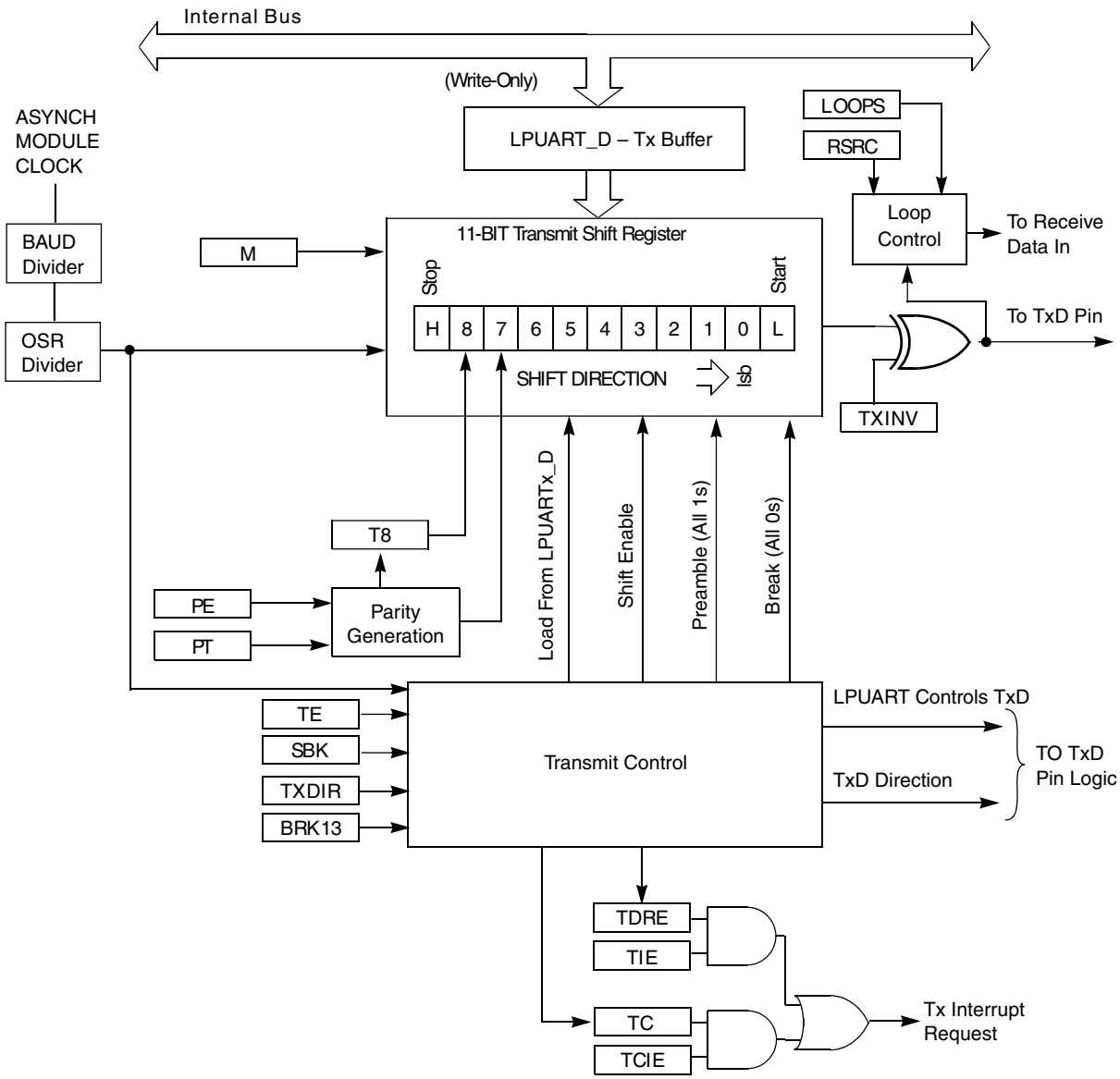



**Figure 56-1. LPUART transmitter block diagram**

The following figure shows the receiver portion of the LPUART.

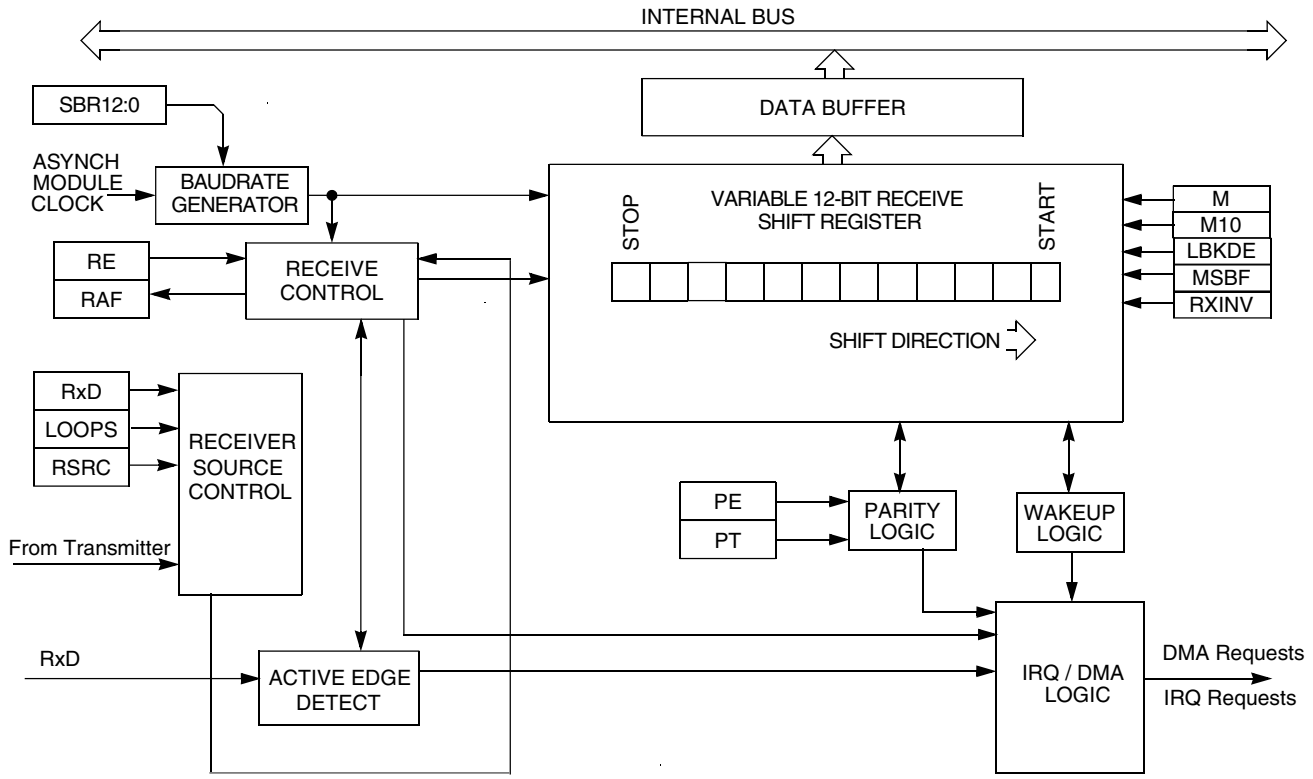



**Figure 56-2. LPUART receiver block diagram**

## 56.3 Register definition

The LPUART includes registers to control baud rate, select options, report status, and store transmit/receive data. Access to an address outside the valid memory map will generate a bus error.

### 56.3.1 LPUART0 register descriptions

#### 56.3.1.1 LPUART0 Memory map

LPUART0 base address: 4004_2000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0401_0003h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0303h |
| 8h | LPUART Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| Ch | LPUART Pin Configuration Register (PINCFG) | 32 | RW | 0000_0000h |
| 10h | LPUART Baud Rate Register (BAUD) | 32 | RW | 0F00_0004h |
| 14h | LPUART Status Register (STAT) | 32 | RW | 00C0_0000h |
| 18h | LPUART Control Register (CTRL) | 32 | RW | 0000_0000h |
| 1Ch | LPUART Data Register (DATA) | 32 | RW | 0000_1000h |
| 20h | LPUART Match Address Register (MATCH) | 32 | RW | 0000_0000h |
| 24h | LPUART Modem IrDA Register (MODIR) | 32 | RW | 0000_0000h |
| 28h | LPUART FIFO Register (FIFO) | 32 | RW | 00C0_0022h |
| 2Ch | LPUART Watermark Register (WATER) | 32 | RW | 0000_0000h |

## 56.3.1.2 Version ID Register (VERID)

### 56.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 56.3.1.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

### 56.3.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with MODEM/IrDA support. |

### 56.3.1.3 Parameter Register (PARAM)

#### 56.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 56.3.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.1.3.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |

*Table continues on the next page...*

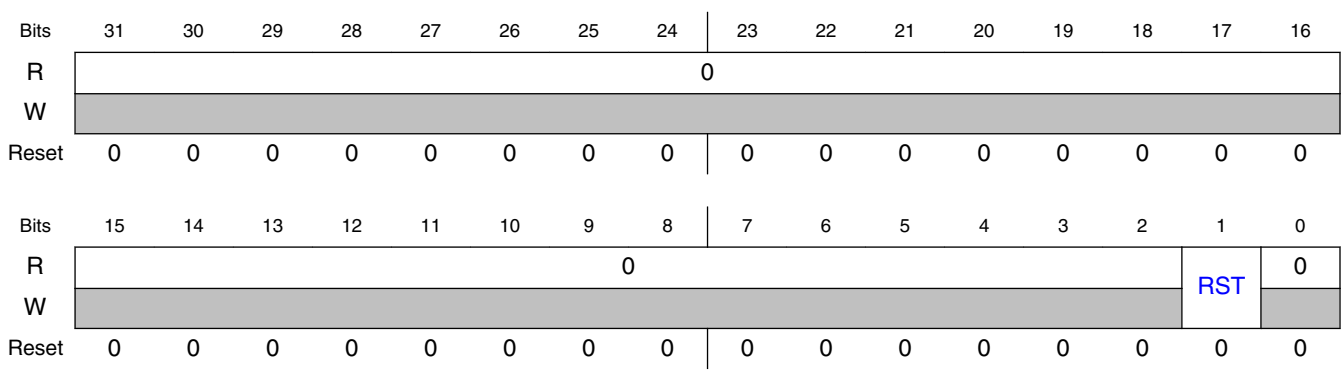
| Field | Function |
|---|---|
| 15-8<br>RXFIFO | Receive FIFO Size<br>The number of words in the receive FIFO is 2^RXFIFO. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>The number of words in the transmit FIFO is 2^TXFIFO. |

## 56.3.1.4 LPUART Global Register (GLOBAL)

### 56.3.1.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 56.3.1.4.2 Diagram




### 56.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>— | Reserved |

## 56.3.1.5 LPUART Pin Configuration Register (PINCFG)

### 56.3.1.5.1 Offset

| Register | Offset |
|---|---|
| PINCFG | Ch |

### 56.3.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TRGSEL | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.1.5.3 Fields

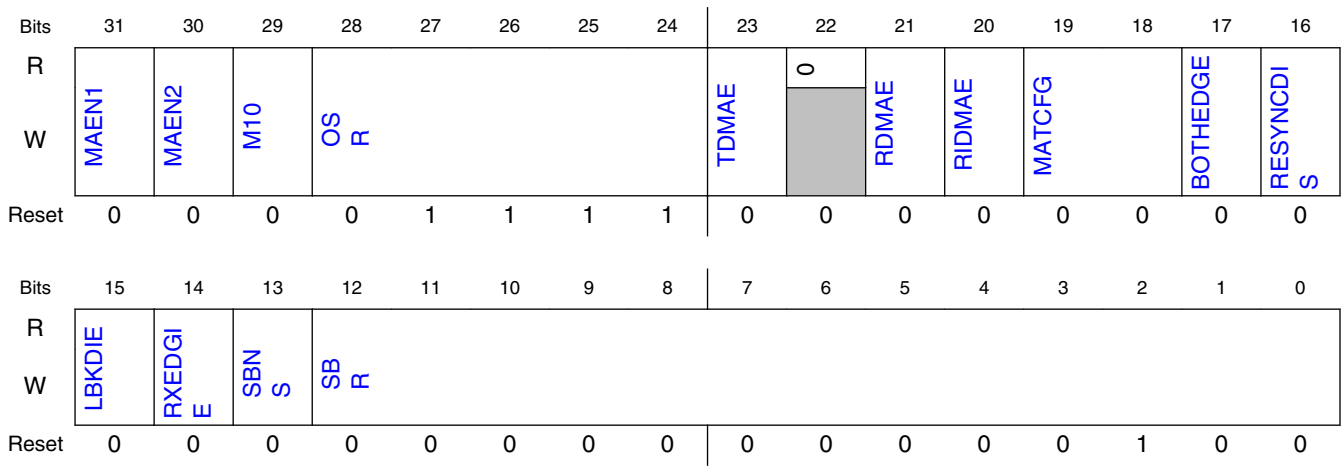
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>TRGSEL | Trigger Select<br>Configures the input trigger usage. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Input trigger is disabled.<br>01b - Input trigger is used instead of RXD pin input.<br>10b - Input trigger is used instead of CTS_B pin input.<br>11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger. |

## 56.3.1.6 LPUART Baud Rate Register (BAUD)

## 56.3.1.6.1 Offset

| Register | Offset |
|---|---|
| BAUD | 10h |

## 56.3.1.6.2 Diagram




## 56.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31<br>MAEN1 | Match Address Mode Enable 1<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA1]. |
| 30<br>MAEN2 | Match Address Mode Enable 2<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA2]. |
| 29<br>M10 | 10-bit Mode select<br>The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 7-bit to 9-bit data characters.<br>1b - Receiver and transmitter use 10-bit data characters. |
| 28-24<br>OSR | Oversampling Ratio<br>This field configures the oversampling ratio for the receiver. This field should only be changed when the transmitter and receiver are both disabled.<br>00000b - Writing 0 to this field will result in an oversampling ratio of 16<br>00001b - Reserved<br>00010b - Reserved<br>00011b - Oversampling ratio of 4, requires BOTHEDGE to be set.<br>00100b - Oversampling ratio of 5, requires BOTHEDGE to be set.<br>00101b - Oversampling ratio of 6, requires BOTHEDGE to be set.<br>00110b - Oversampling ratio of 7, requires BOTHEDGE to be set.<br>00111b - Oversampling ratio of 8. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 01000b - Oversampling ratio of 9.<br>01001b - Oversampling ratio of 10.<br>01010b - Oversampling ratio of 11.<br>01011b - Oversampling ratio of 12.<br>01100b - Oversampling ratio of 13.<br>01101b - Oversampling ratio of 14.<br>01110b - Oversampling ratio of 15.<br>01111b - Oversampling ratio of 16.<br>10000b - Oversampling ratio of 17.<br>10001b - Oversampling ratio of 18.<br>10010b - Oversampling ratio of 19.<br>10011b - Oversampling ratio of 20.<br>10100b - Oversampling ratio of 21.<br>10101b - Oversampling ratio of 22.<br>10110b - Oversampling ratio of 23.<br>10111b - Oversampling ratio of 24.<br>11000b - Oversampling ratio of 25.<br>11001b - Oversampling ratio of 26.<br>11010b - Oversampling ratio of 27.<br>11011b - Oversampling ratio of 28.<br>11100b - Oversampling ratio of 29.<br>11101b - Oversampling ratio of 30.<br>11110b - Oversampling ratio of 31.<br>11111b - Oversampling ratio of 32. |
| 23<br>TDMAE | Transmitter DMA Enable<br>TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 22<br>— | Reserved |
| 21<br>RDMAE | Receiver Full DMA Enable<br>RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 20<br>RIDMAE | Receiver Idle DMA Enable<br>RIDMAE configures the receiver idle flag, LPUART_STAT[IDLE], to generate a DMA request. When this bit is set, reading LPUART_DATA when either DATA[RXEMPT] or DATA[IDLINE] bit is set, will generate an End Of Packet response until the completion of the existing DMA transfer. During an End of Packet response, reading the LPUART_DATA register will return 0x0000_33FF and does not pull data from the FIFO.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 19-18<br>MATCFG | Match Configuration<br>Configures the match addressing mode used. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Address Match Wakeup<br>01b - Idle Match Wakeup<br>10b - Match On and Match Off<br>11b - Enables RWU on Data Match and Match On/Off for transmitter CTS input |
| 17 | Both Edge Sampling |

*Table continues on the next page...*

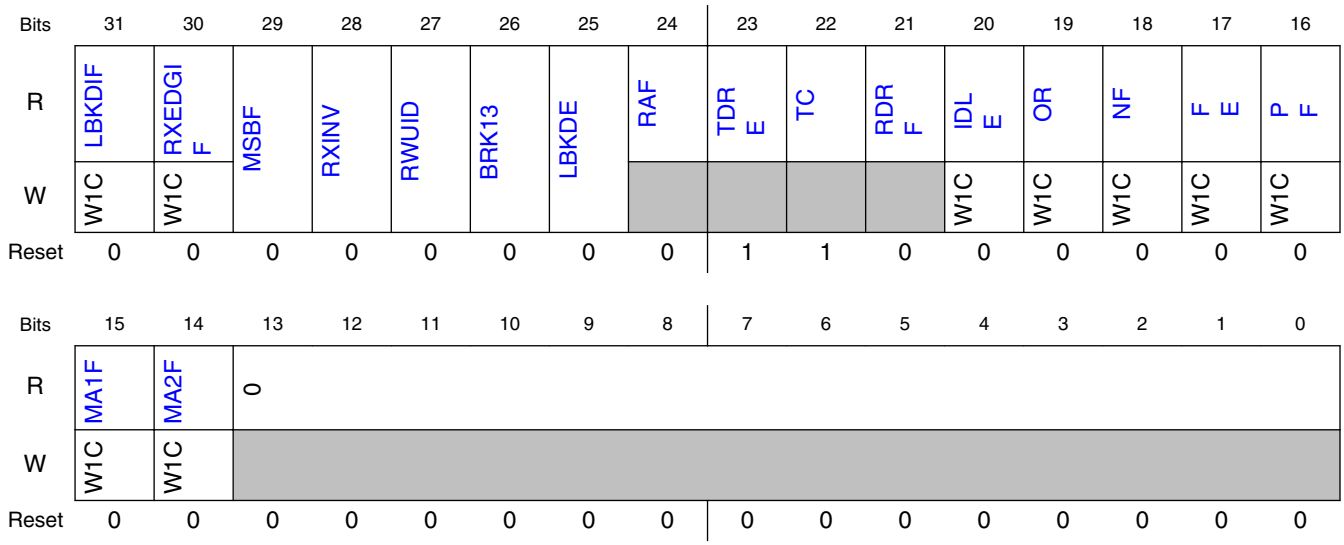
| Field | Function |
|---|---|
| BOTHEDGE | Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.<br>0b - Receiver samples input data using the rising edge of the baud rate clock.<br>1b - Receiver samples input data using the rising and falling edge of the baud rate clock. |
| 16<br>RESYNCDIS | Resynchronization Disable<br>When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.<br>0b - Resynchronization during received data word is supported<br>1b - Resynchronization during received data word is disabled |
| 15<br>LBKDIE | LIN Break Detect Interrupt Enable<br>LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.<br>0b - Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling).<br>1b - Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1. |
| 14<br>RXEDGIE | RX Input Active Edge Interrupt Enable<br>Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.<br>0b - Hardware interrupts from LPUART_STAT[RXEDGIF] disabled.<br>1b - Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1. |
| 13<br>SBNS | Stop Bit Number Select<br>SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - One stop bit.<br>1b - Two stop bits. |
| 12-0<br>SBR | Baud Rate Modulo Divisor.<br>The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0). |

## 56.3.1.7 LPUART Status Register (STAT)

### 56.3.1.7.1 Offset

| Register | Offset |
|---|---|
| STAT | 14h |

### 56.3.1.7.2 Diagram




### 56.3.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>LBKDIF | LIN Break Detect Interrupt Flag<br>LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.<br>0b - No LIN break character has been detected.<br>1b - LIN break character has been detected. |
| 30<br>RXEDGIF | RXD Pin Active Edge Interrupt Flag<br>RXEDGIF is set whenever the receiver is enabled and an active edge, falling if RXINV = 0, rising if RXINV=1, on the RXD pin occurs. RXEDGIF is cleared by writing a 1 to it.<br>0b - No active edge on the receive pin has occurred.<br>1b - An active edge on the receive pin has occurred. |
| 29<br>MSBF | MSB First<br>Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0.<br>1b - MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE]. |
| 28<br>RXINV | Receive Data Inversion<br>Setting this bit reverses the polarity of the received data input. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.<br>0b - Receive data not inverted.<br>1b - Receive data inverted. |
| 27 | Receive Wake Up Idle Detect |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RWUID | For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.<br>0b - During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not set when an address does not match.<br>1b - During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does set when an address does not match. |
| 26<br>BRK13 | Break Character Generation Length<br>BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled. A break character can be sent by setting CTRL[SBK] or by writing the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Break character is transmitted with length of 9 to 13 bit times.<br>1b - Break character is transmitted with length of 12 to 15 bit times. |
| 25<br>LBKDE | LIN Break Detection Enable<br>LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.<br>0b - LIN break detect is disabled, normal break character can be detected.<br>1b - LIN break detect is enabled. LIN break character is detected at length of 11 bit times (if M = 0) or 12 (if M = 1) or 13 (M10 = 1). |
| 24<br>RAF | Receiver Active Flag<br>RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.<br>0b - LPUART receiver idle waiting for a start bit.<br>1b - LPUART receiver active (RXD input not idle). |
| 23<br>TDRE | Transmit Data Register Empty Flag<br>When the transmit FIFO is enabled, TDRE will set when the number of datawords in the transmit FIFO (LPUART_DATA) is equal to or less than the number indicated by LPUART_WATER[TXWATER]). To clear TDRE, write to the LPUART data register (LPUART_DATA) until the number of words in the transmit FIFO is greater than the number indicated by LPUART_WATER[TXWATER]. When the transmit FIFO is disabled,TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA).<br>TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.<br>0b - Transmit data buffer full.<br>1b - Transmit data buffer empty. |
| 22<br>TC | Transmission Complete Flag<br>TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK].<br>0b - Transmitter active (sending data, a preamble, or a break).<br>1b - Transmitter idle (transmission activity complete). |
| 21<br>RDRF | Receive Data Register Full Flag<br>When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by LPUART_WATER[RXWATER]. To clear RDRF, read LPUART_DATA until the number of datawords in the receive data buffer is equal to or less than the number indicated by LPUART_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.<br>0b - Receive data buffer empty.<br>1b - Receive data buffer full. |
| 20<br>IDLE | Idle Line Flag<br>IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period.<br>0b - No idle line detected.<br>1b - Idle line was detected. |
| 19<br>OR | Receiver Overrun Flag<br>OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.<br>While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.<br>0b - No overrun.<br>1b - Receive overrun (new LPUART data lost). |
| 18<br>NF | Noise Flag<br>The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.<br>0b - No noise detected.<br>1b - Noise detected in the received character in LPUART_DATA. |
| 17<br>FE | Framing Error Flag<br>FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE.<br>0b - No framing error detected. This does not guarantee the framing is correct.<br>1b - Framing error. |
| 16<br>PF | Parity Error Flag<br>PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.<br>0b - No parity error.<br>1b - Parity error. |
| 15<br>MA1F | Match 1 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.<br>0b - Received data is not equal to MA1<br>1b - Received data is equal to MA1 |
| 14<br>MA2F | Match 2 Flag<br>MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.<br>0b - Received data is not equal to MA2<br>1b - Received data is equal to MA2 |
| 13-0<br>— | Reserved |

## 56.3.1.8 LPUART Control Register (CTRL)

### 56.3.1.8.1 Offset

| Register | Offset |
|---|---|
| CTRL | 18h |

### 56.3.1.8.2 Function

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

### 56.3.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MA1IE | MA2IE | 0 | | M7 | IDLECFG | | | LOOPS | DOZEEN | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.1.8.4 Fields

| Field | Function |
|---|---|
| 31<br>R8T9 | Receive Bit 8 / Transmit Bit 9<br>R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA.<br>T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R8 is a read-only bit and T9 is a write-only bit, the value read is different from the value written. |
| 30<br>R9T8 | Receive Bit 9 / Transmit Bit 8<br>R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA<br>T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R9 is a read-only bit and T8 is a write-only bit, the value read is different from the value written. |
| 29<br>TXDIR | TXD Pin Direction in Single-Wire Mode<br>When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the TXD pin.<br>0b - TXD pin is an input in single-wire mode.<br>1b - TXD pin is an output in single-wire mode. |
| 28<br>TXINV | Transmit Data Inversion<br>Setting this bit reverses the polarity of the transmitted data output.<br>**NOTE:** Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.<br>0b - Transmit data not inverted.<br>1b - Transmit data inverted. |
| 27<br>ORIE | Overrun Interrupt Enable<br>This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0b - OR interrupts disabled; use polling.<br>1b - Hardware interrupt requested when OR is set. |
| 26<br>NEIE | Noise Error Interrupt Enable<br>This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0b - NF interrupts disabled; use polling.<br>1b - Hardware interrupt requested when NF is set. |
| 25<br>FEIE | Framing Error Interrupt Enable<br>This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0b - FE interrupts disabled; use polling.<br>1b - Hardware interrupt requested when FE is set. |
| 24<br>PEIE | Parity Error Interrupt Enable<br>This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0b - PF interrupts disabled; use polling).<br>1b - Hardware interrupt requested when PF is set. |
| 23 | Transmit Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIE | Enables STAT[TDRE] to generate interrupt requests.<br>0b - Hardware interrupts from TDRE disabled; use polling.<br>1b - Hardware interrupt requested when TDRE flag is 1. |
| 22<br>TCIE | Transmission Complete Interrupt Enable for<br>TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0b - Hardware interrupts from TC disabled; use polling.<br>1b - Hardware interrupt requested when TC flag is 1. |
| 21<br>RIE | Receiver Interrupt Enable<br>Enables STAT[RDRF] to generate interrupt requests.<br>0b - Hardware interrupts from RDRF disabled; use polling.<br>1b - Hardware interrupt requested when RDRF flag is 1. |
| 20<br>ILIE | Idle Line Interrupt Enable<br>ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.<br>0b - Hardware interrupts from IDLE disabled; use polling.<br>1b - Hardware interrupt requested when IDLE flag is 1. |
| 19<br>TE | Transmitter Enable<br>Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the TXD pin is tristated.<br>A single idle character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] set.<br>0b - Transmitter disabled.<br>1b - Transmitter enabled. |
| 18<br>RE | Receiver Enable<br>Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).<br>0b - Receiver disabled.<br>1b - Receiver enabled. |
| 17<br>RWU | Receiver Wakeup Control<br>This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.<br>**NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.<br>0b - Normal receiver operation.<br>1b - LPUART receiver in standby waiting for wakeup condition. |
| 16<br>SBK | Send Break<br>Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if LPUART_STAT[BRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK.<br>A single break character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Normal transmitter operation.<br>1b - Queue break character(s) to be sent. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>MA1IE | Match 1 Interrupt Enable<br>0b - MA1F interrupt disabled<br>1b - MA1F interrupt enabled |
| 14<br>MA2IE | Match 2 Interrupt Enable<br>0b - MA2F interrupt disabled<br>1b - MA2F interrupt enabled |
| 13-12<br>— | Reserved |
| 11<br>M7 | 7-Bit Mode Select<br>This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 8-bit to 10-bit data characters.<br>1b - Receiver and transmitter use 7-bit data characters. |
| 10-8<br>IDLECFG | Idle Configuration<br>Configures the number of idle characters that must be received before the IDLE flag is set.<br>000b - 1 idle character<br>001b - 2 idle characters<br>010b - 4 idle characters<br>011b - 8 idle characters<br>100b - 16 idle characters<br>101b - 32 idle characters<br>110b - 64 idle characters<br>111b - 128 idle characters |
| 7<br>LOOPS | Loop Mode Select<br>When LOOPS is set, the RXD pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.<br>0b - Normal operation - RXD and TXD use separate pins.<br>1b - Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit). |
| 6<br>DOZEEN | Doze Enable<br>0b - LPUART is enabled in Doze mode.<br>1b - LPUART is disabled in Doze mode. |
| 5<br>RSRC | Receiver Source Select<br>This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.<br>0b - Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the RXD pin.<br>1b - Single-wire LPUART mode where the TXD pin is connected to the transmitter output and receiver input. |
| 4<br>M | 9-Bit or 8-Bit Mode Select<br>0b - Receiver and transmitter use 8-bit data characters.<br>1b - Receiver and transmitter use 9-bit data characters. |
| 3<br>WAKE | Receiver Wakeup Method Select<br>Determines which condition wakes the LPUART when RWU=1:<br>• Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character, or<br>• An idle condition on the receive pin input signal.<br><br>0b - Configures RWU for idle-line wakeup.<br>1b - Configures RWU with address-mark wakeup. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>ILT | Idle Line Type Select<br>Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>**NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>0b - Idle character bit count starts after start bit.<br>1b - Idle character bit count starts after stop bit. |
| 1<br>PE | Parity Enable<br>Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.<br>0b - No hardware parity generation or checking.<br>1b - Parity enabled. |
| 0<br>PT | Parity Type<br>Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0b - Even parity.<br>1b - Odd parity. |

## 56.3.1.9 LPUART Data Register (DATA)

### 56.3.1.9.1 Offset

| Register | Offset |
|---|---|
| DATA | 1Ch |

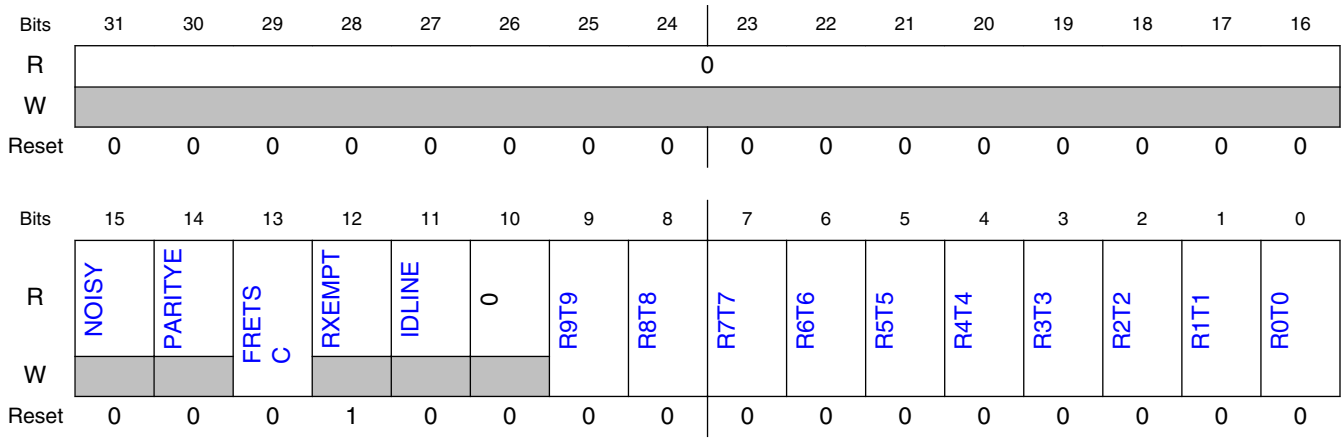
### 56.3.1.9.2 Function

**NOTE**

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

### 56.3.1.9.3 Diagram




### 56.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>NOISY | NOISY<br>The current received dataword contained in DATA[R9:R0] was received with noise.<br>0b - The dataword was received without noise.<br>1b - The data was received with noise. |
| 14<br>PARITYE | PARITYE<br>The current received dataword contained in DATA[R9:R0] was received with a parity error.<br>0b - The dataword was received without a parity error.<br>1b - The dataword was received with a parity error. |
| 13<br>FRETSC | Frame Error / Transmit Special Character<br>For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero.<br>0b - The dataword was received without a frame error on read, or transmit a normal character on write.<br>1b - The dataword was received with a frame error, or transmit an idle or break character on transmit. |
| 12<br>RXEMPT | Receive Buffer Empty<br>Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer contains valid data.<br>1b - Receive buffer is empty, data returned on read is not valid. |
| 11<br>IDLINE | Idle Line<br>Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.<br>0b - Receiver was not idle before receiving this character.<br>1b - Receiver was idle before receiving this character. |

*Table continues on the next page...*

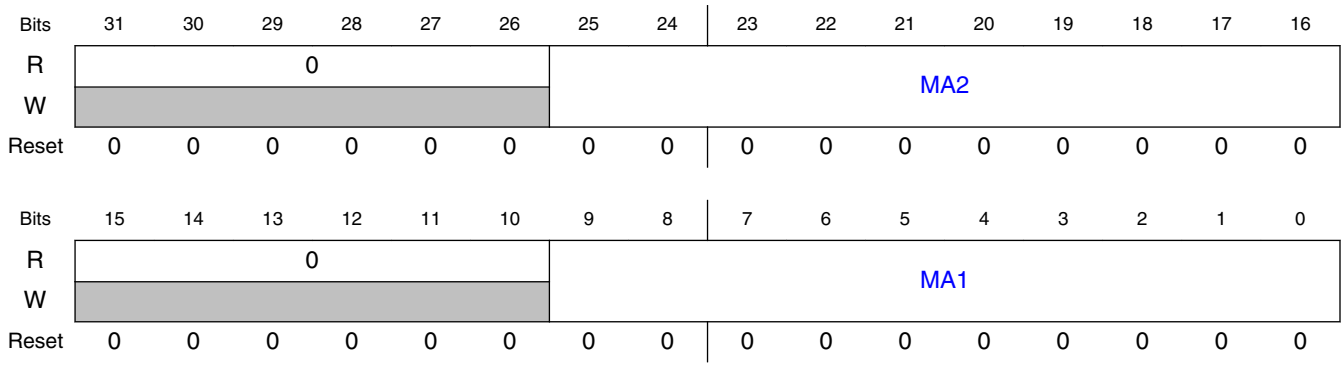
| Field | Function |
|---|---|
| 10<br>— | Reserved |
| 9<br>R9T9 | R9T9<br>Read receive data buffer 9 or write transmit data buffer 9. |
| 8<br>R8T8 | R8T8<br>Read receive data buffer 8 or write transmit data buffer 8. |
| 7<br>R7T7 | R7T7<br>Read receive data buffer 7 or write transmit data buffer 7. |
| 6<br>R6T6 | R6T6<br>Read receive data buffer 6 or write transmit data buffer 6. |
| 5<br>R5T5 | R5T5<br>Read receive data buffer 5 or write transmit data buffer 5. |
| 4<br>R4T4 | R4T4<br>Read receive data buffer 4 or write transmit data buffer 4. |
| 3<br>R3T3 | R3T3<br>Read receive data buffer 3 or write transmit data buffer 3. |
| 2<br>R2T2 | R2T2<br>Read receive data buffer 2 or write transmit data buffer 2. |
| 1<br>R1T1 | R1T1<br>Read receive data buffer 1 or write transmit data buffer 1. |
| 0<br>R0T0 | R0T0<br>Read receive data buffer 0 or write transmit data buffer 0. |

## 56.3.1.10 LPUART Match Address Register (MATCH)

### 56.3.1.10.1 Offset

| Register | Offset |
|---|---|
| MATCH | 20h |

### 56.3.1.10.2 Diagram




### 56.3.1.10.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>MA2 | Match Address 2<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |
| 15-10<br>— | Reserved |
| 9-0<br>MA1 | Match Address 1<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |

## 56.3.1.11 LPUART Modem IrDA Register (MODIR)

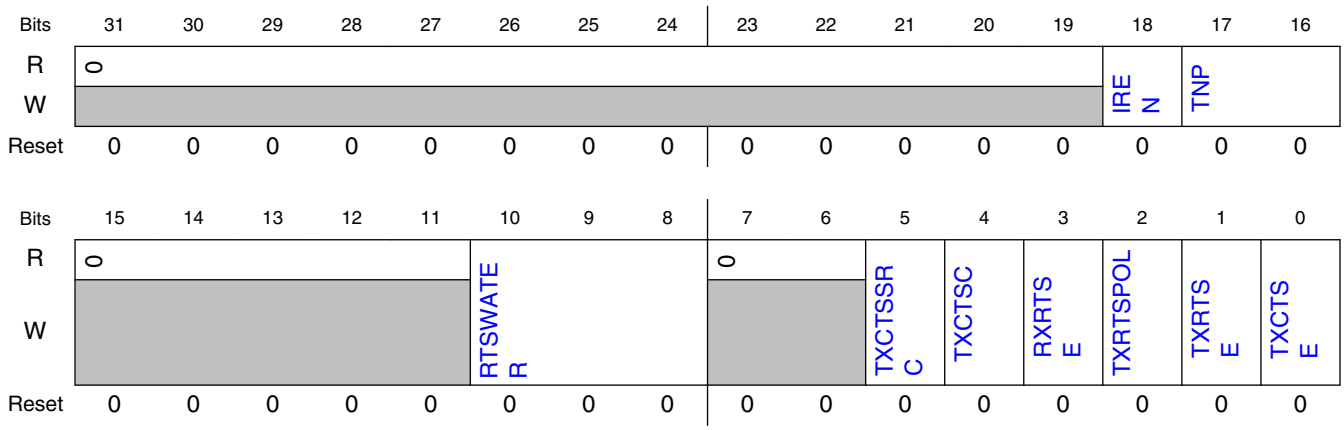
### 56.3.1.11.1 Offset

| Register | Offset |
|---|---|
| MODIR | 24h |

### 56.3.1.11.2 Function

The MODEM register controls options for setting the modem configuration.

### 56.3.1.11.3 Diagram




### 56.3.1.11.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>IREN | Infrared enable<br>Enables/disables the infrared modulation/demodulation. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - IR disabled.<br>1b - IR enabled. |
| 17-16<br>TNP | Transmitter narrow pulse<br>Configures whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse when IR is enabled. This bit should only be changed when the transmitter and receiver are both disabled.<br>The IR pulse width should be configured to less than half of the oversampling ratio. Common pulse widths are 3/16, 1/16, 1/32 or 1/4 of the bit length. These can be configured by selecting the appropriate oversample ratio and pulse width.<br>00b - 1/OSR.<br>01b - 2/OSR.<br>10b - 3/OSR.<br>11b - 4/OSR. |
| 15-11<br>— | Reserved |
| 10-8<br>RTSWATER | Receive RTS Configuration<br>Configures the assertion and negation of the RX RTS_B output. The RX RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to RTSWATER. If RTSWATER is configured to zero, RTS_B negates when the receive FIFO is full. For the purpose of RX RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both RX RTS_B and address/data matching is enabled, then RTS_B could assert at the end of a character if there is not a match. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This field should only be changed when the receiver is disabled. |
| 7-6<br>— | Reserved |
| 5<br>TXCTSSRC | Transmit CTS Source<br>Configures the source of the CTS input.<br>0b - CTS input is the CTS_B pin.<br>1b - CTS input is the inverted Receiver Match result. |
| 4<br>TXCTSC | Transmit CTS Configuration<br>Configures if the CTS state is checked at the start of each character or only when the transmitter is idle.<br>0b - CTS input is sampled at the start of each character.<br>1b - CTS input is sampled when the transmitter is idle. |
| 3<br>RXRTSE | Receiver request-to-send enable<br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Do not set both RXRTSE and TXRTSE.<br>0b - The receiver has no effect on RTS.<br>1b - RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full. |
| 2<br>TXRTSPOL | Transmitter request-to-send polarity<br>Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set. This bit should only be changed when the transmitter is disabled.<br>0b - Transmitter RTS is active low.<br>1b - Transmitter RTS is active high. |
| 1<br>TXRTSE | Transmitter request-to-send enable<br>Controls RTS before and after a transmission. This bit should only be changed when the transmitter is disabled.<br>0b - The transmitter has no effect on RTS.<br>1b - When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. |
| 0<br>TXCTSE | Transmitter clear-to-send enable<br>TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.<br>0b - CTS has no effect on the transmitter.<br>1b - Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission. |

## 56.3.1.12 LPUART FIFO Register (FIFO)

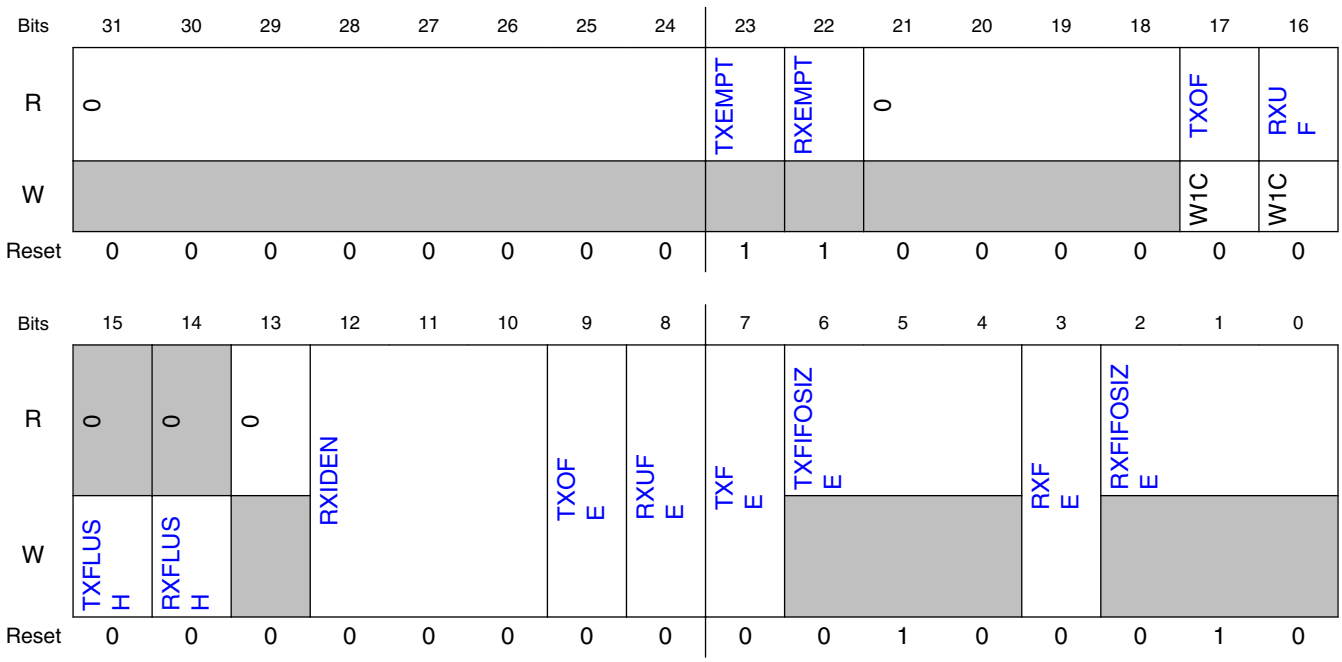
### 56.3.1.12.1 Offset

| Register | Offset |
|---|---|
| FIFO | 28h |

### 56.3.1.12.2 Function

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when CTRL[RE] and CTRL[TE] are cleared/not set and when the data buffer/FIFO is empty.

### 56.3.1.12.3 Diagram




### 56.3.1.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>TXEMPT | Transmit Buffer/FIFO Empty<br>Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register.<br>0b - Transmit buffer is not empty. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Transmit buffer is empty. |
| 22<br>RXEMPT | Receive Buffer/FIFO Empty<br>Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer is not empty.<br>1b - Receive buffer is empty. |
| 21-18<br>— | Reserved |
| 17<br>TXOF | Transmitter Buffer Overflow Flag<br>Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1.<br>0b - No transmit buffer overflow has occurred since the last time the flag was cleared.<br>1b - At least one transmit buffer overflow has occurred since the last time the flag was cleared. |
| 16<br>RXUF | Receiver Buffer Underflow Flag<br>Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1.<br>0b - No receive buffer underflow has occurred since the last time the flag was cleared.<br>1b - At least one receive buffer underflow has occurred since the last time the flag was cleared. |
| 15<br>TXFLUSH | Transmit FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.<br>0b - No flush operation occurs.<br>1b - All data in the transmit FIFO/Buffer is cleared out. |
| 14<br>RXFLUSH | Receive FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.<br>0b - No flush operation occurs.<br>1b - All data in the receive FIFO/buffer is cleared out. |
| 13<br>— | Reserved |
| 12-10<br>RXIDEN | Receiver Idle Empty Enable<br>When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty.<br>000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle.<br>001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character.<br>010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters.<br>011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters.<br>100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters.<br>101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters.<br>110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters.<br>111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters. |
| 9<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br>When this field is set, the TXOF flag generates an interrupt to the host.<br>0b - TXOF flag does not generate an interrupt to the host.<br>1b - TXOF flag generates an interrupt to the host. |
| 8 | Receive FIFO Underflow Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXUFE | When this field is set, the RXUF flag generates an interrupt to the host.<br>0b - RXUF flag does not generate an interrupt to the host.<br>1b - RXUF flag generates an interrupt to the host. |
| 7<br>TXFE | Transmit FIFO Enable<br>When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).<br>1b - Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE. |
| 6-4<br>TXFIFOSIZE | Transmit FIFO. Buffer Depth<br>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.<br>000b - Transmit FIFO/Buffer depth = 1 dataword.<br>001b - Transmit FIFO/Buffer depth = 4 datawords.<br>010b - Transmit FIFO/Buffer depth = 8 datawords.<br>011b - Transmit FIFO/Buffer depth = 16 datawords.<br>100b - Transmit FIFO/Buffer depth = 32 datawords.<br>101b - Transmit FIFO/Buffer depth = 64 datawords.<br>110b - Transmit FIFO/Buffer depth = 128 datawords.<br>111b - Transmit FIFO/Buffer depth = 256 datawords |
| 3<br>RXFE | Receive FIFO Enable<br>When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)<br>1b - Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE. |
| 2-0<br>RXFIFOSIZE | Receive FIFO. Buffer Depth<br>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.<br>000b - Receive FIFO/Buffer depth = 1 dataword.<br>001b - Receive FIFO/Buffer depth = 4 datawords.<br>010b - Receive FIFO/Buffer depth = 8 datawords.<br>011b - Receive FIFO/Buffer depth = 16 datawords.<br>100b - Receive FIFO/Buffer depth = 32 datawords.<br>101b - Receive FIFO/Buffer depth = 64 datawords.<br>110b - Receive FIFO/Buffer depth = 128 datawords.<br>111b - Receive FIFO/Buffer depth = 256 datawords. |

## 56.3.1.13 LPUART Watermark Register (WATER)

### 56.3.1.13.1 Offset

| Register | Offset |
|---|---|
| WATER | 2Ch |

### 56.3.1.13.2 Function

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when CTRL[TE] is not set.

### 56.3.1.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RXCOUNT | | | | 0 | | | | | RXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TXCOUNT | | | | 0 | | | | | TXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.1.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>RXCOUNT | Receive Counter<br>The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer. |
| 23-19<br>— | Reserved |
| 18-16<br>RXWATER | Receive Watermark<br>When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0. |
| 15-12<br>— | Reserved |
| 11-8<br>TXCOUNT | Transmit Counter<br>The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer. |
| 7-3 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 2-0<br>TXWATER | Transmit Watermark<br>When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE]. |

## 56.3.2 LPUART1 register descriptions

### 56.3.2.1 LPUART1 Memory map

LPUART1 base address: 4004_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0401_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0303h |
| 8h | LPUART Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| Ch | LPUART Pin Configuration Register (PINCFG) | 32 | RW | 0000_0000h |
| 10h | LPUART Baud Rate Register (BAUD) | 32 | RW | 0F00_0004h |
| 14h | LPUART Status Register (STAT) | 32 | RW | 00C0_0000h |
| 18h | LPUART Control Register (CTRL) | 32 | RW | 0000_0000h |
| 1Ch | LPUART Data Register (DATA) | 32 | RW | 0000_1000h |
| 20h | LPUART Match Address Register (MATCH) | 32 | RW | 0000_0000h |
| 24h | LPUART Modem IrDA Register (MODIR) | 32 | RW | 0000_0000h |
| 28h | LPUART FIFO Register (FIFO) | 32 | RW | 00C0_0022h |
| 2Ch | LPUART Watermark Register (WATER) | 32 | RW | 0000_0000h |

### 56.3.2.2 Version ID Register (VERID)

#### 56.3.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 56.3.2.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

#### 56.3.2.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with MODEM/IrDA support. |

### 56.3.2.3 Parameter Register (PARAM)

#### 56.3.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 56.3.2.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 56.3.2.3.3 Fields

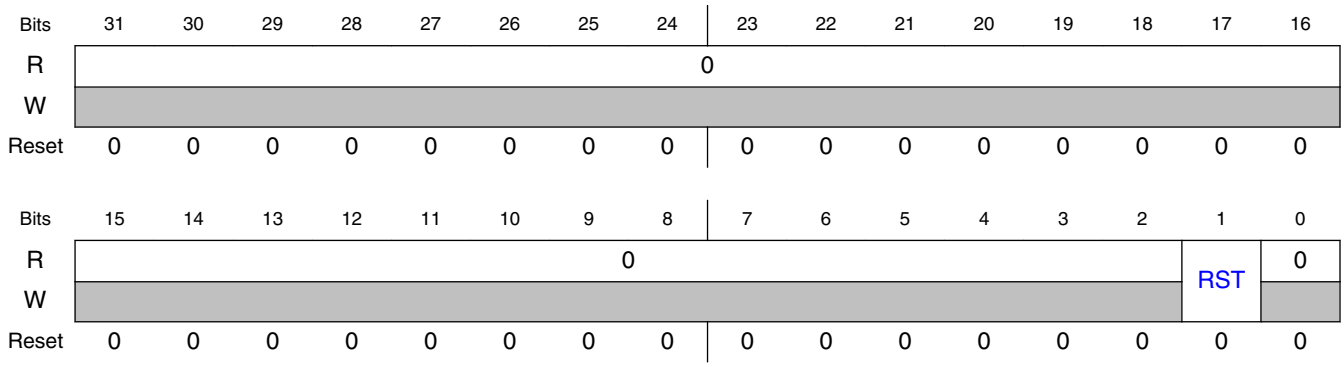
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>RXFIFO | Receive FIFO Size<br>The number of words in the receive FIFO is 2^RXFIFO. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>The number of words in the transmit FIFO is 2^TXFIFO. |

## 56.3.2.4 LPUART Global Register (GLOBAL)

### 56.3.2.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 56.3.2.4.2 Diagram




### 56.3.2.4.3 Fields

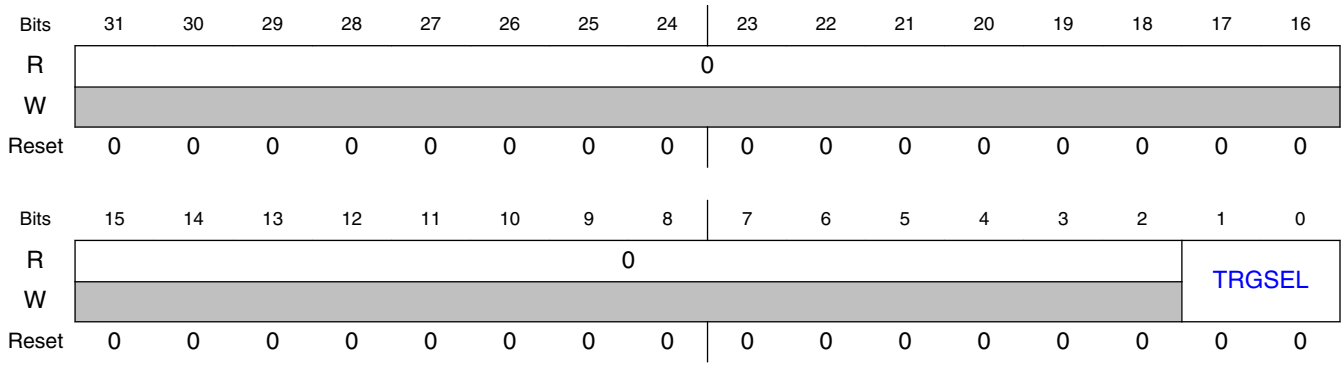
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>— | Reserved |

## 56.3.2.5 LPUART Pin Configuration Register (PINCFG)

### 56.3.2.5.1 Offset

| Register | Offset |
|---|---|
| PINCFG | Ch |

### 56.3.2.5.2 Diagram




### 56.3.2.5.3 Fields

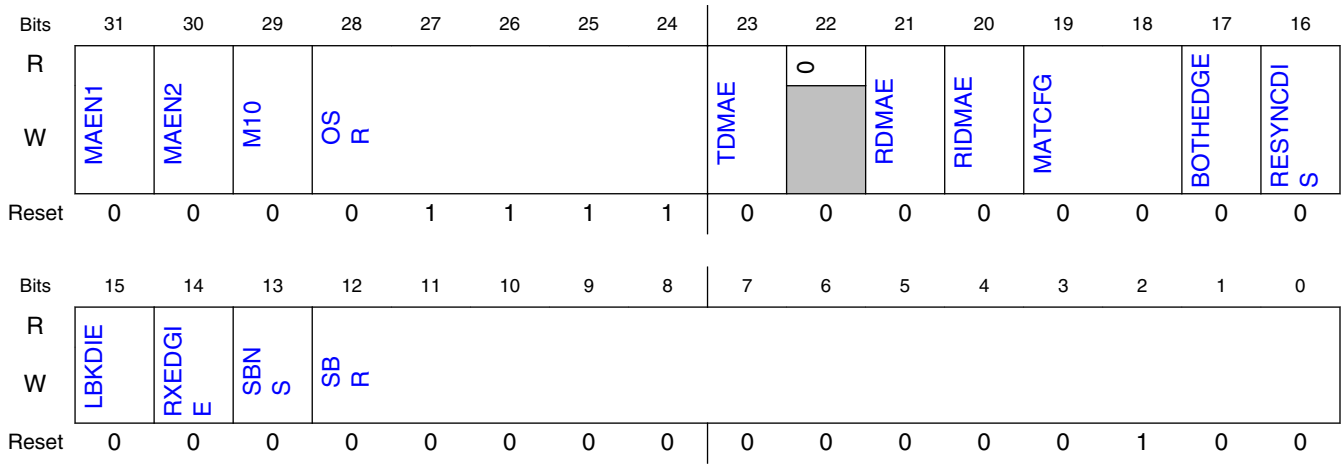
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>TRGSEL | Trigger Select<br>Configures the input trigger usage. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Input trigger is disabled.<br>01b - Input trigger is used instead of RXD pin input.<br>10b - Input trigger is used instead of CTS_B pin input.<br>11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger. |

## 56.3.2.6 LPUART Baud Rate Register (BAUD)

### 56.3.2.6.1 Offset

| Register | Offset |
|---|---|
| BAUD | 10h |

### 56.3.2.6.2 Diagram




### 56.3.2.6.3 Fields

| Field | Function |
|---|---|
| 31<br>MAEN1 | Match Address Mode Enable 1<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA1]. |
| 30<br>MAEN2 | Match Address Mode Enable 2<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA2]. |
| 29<br>M10 | 10-bit Mode select<br>The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 7-bit to 9-bit data characters.<br>1b - Receiver and transmitter use 10-bit data characters. |
| 28-24<br>OSR | Oversampling Ratio<br>This field configures the oversampling ratio for the receiver. This field should only be changed when the transmitter and receiver are both disabled.<br>00000b - Writing 0 to this field will result in an oversampling ratio of 16<br>00001b - Reserved<br>00010b - Reserved<br>00011b - Oversampling ratio of 4, requires BOTHEDGE to be set.<br>00100b - Oversampling ratio of 5, requires BOTHEDGE to be set.<br>00101b - Oversampling ratio of 6, requires BOTHEDGE to be set.<br>00110b - Oversampling ratio of 7, requires BOTHEDGE to be set.<br>00111b - Oversampling ratio of 8.<br>01000b - Oversampling ratio of 9.<br>01001b - Oversampling ratio of 10.<br>01010b - Oversampling ratio of 11.<br>01011b - Oversampling ratio of 12.<br>01100b - Oversampling ratio of 13.<br>01101b - Oversampling ratio of 14.<br>01110b - Oversampling ratio of 15.<br>01111b - Oversampling ratio of 16.<br>10000b - Oversampling ratio of 17. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 10001b - Oversampling ratio of 18.<br>10010b - Oversampling ratio of 19.<br>10011b - Oversampling ratio of 20.<br>10100b - Oversampling ratio of 21.<br>10101b - Oversampling ratio of 22.<br>10110b - Oversampling ratio of 23.<br>10111b - Oversampling ratio of 24.<br>11000b - Oversampling ratio of 25.<br>11001b - Oversampling ratio of 26.<br>11010b - Oversampling ratio of 27.<br>11011b - Oversampling ratio of 28.<br>11100b - Oversampling ratio of 29.<br>11101b - Oversampling ratio of 30.<br>11110b - Oversampling ratio of 31.<br>11111b - Oversampling ratio of 32. |
| 23<br>TDMAE | Transmitter DMA Enable<br>TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 22<br>— | Reserved |
| 21<br>RDMAE | Receiver Full DMA Enable<br>RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 20<br>RIDMAE | Receiver Idle DMA Enable<br>RIDMAE configures the receiver idle flag, LPUART_STAT[IDLE], to generate a DMA request. When this bit is set, reading LPUART_DATA when either DATA[RXEMPT] or DATA[IDLINE] bit is set, will generate an End Of Packet response until the completion of the existing DMA transfer. During an End of Packet response, reading the LPUART_DATA register will return 0x0000_33FF and does not pull data from the FIFO.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 19-18<br>MATCFG | Match Configuration<br>Configures the match addressing mode used. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Address Match Wakeup<br>01b - Idle Match Wakeup<br>10b - Match On and Match Off<br>11b - Enables RWU on Data Match and Match On/Off for transmitter CTS input |
| 17<br>BOTHEDGE | Both Edge Sampling<br>Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.<br>0b - Receiver samples input data using the rising edge of the baud rate clock.<br>1b - Receiver samples input data using the rising and falling edge of the baud rate clock. |
| 16<br>RESYNCDIS | Resynchronization Disable |

*Table continues on the next page...*

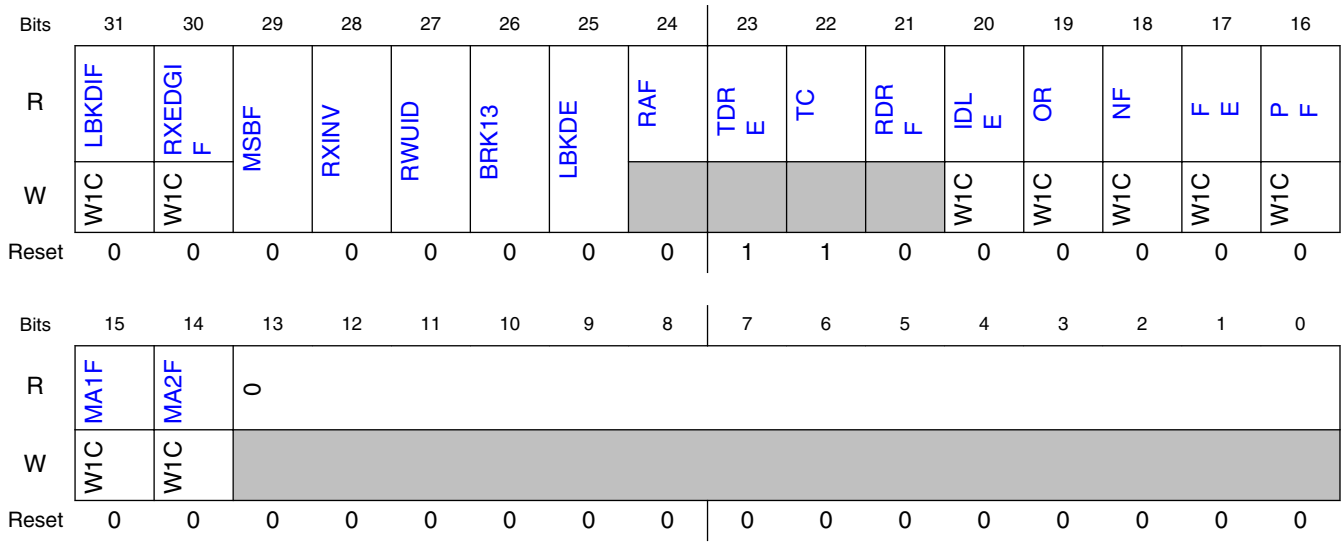
| Field | Function |
|---|---|
| | When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.<br>0b - Resynchronization during received data word is supported<br>1b - Resynchronization during received data word is disabled |
| 15<br>LBKDIE | LIN Break Detect Interrupt Enable<br>LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.<br>0b - Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling).<br>1b - Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1. |
| 14<br>RXEDGIE | RX Input Active Edge Interrupt Enable<br>Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.<br>0b - Hardware interrupts from LPUART_STAT[RXEDGIF] disabled.<br>1b - Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1. |
| 13<br>SBNS | Stop Bit Number Select<br>SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - One stop bit.<br>1b - Two stop bits. |
| 12-0<br>SBR | Baud Rate Modulo Divisor.<br>The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0). |

## 56.3.2.7 LPUART Status Register (STAT)

### 56.3.2.7.1 Offset

| Register | Offset |
|---|---|
| STAT | 14h |

## 56.3.2.7.2 Diagram




## 56.3.2.7.3 Fields

| Field | Function |
|---|---|
| 31<br>LBKDIF | LIN Break Detect Interrupt Flag<br>LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.<br>0b - No LIN break character has been detected.<br>1b - LIN break character has been detected. |
| 30<br>RXEDGIF | RXD Pin Active Edge Interrupt Flag<br>RXEDGIF is set whenever the receiver is enabled and an active edge, falling if RXINV = 0, rising if RXINV=1, on the RXD pin occurs. RXEDGIF is cleared by writing a 1 to it.<br>0b - No active edge on the receive pin has occurred.<br>1b - An active edge on the receive pin has occurred. |
| 29<br>MSBF | MSB First<br>Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0.<br>1b - MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE]. |
| 28<br>RXINV | Receive Data Inversion<br>Setting this bit reverses the polarity of the received data input. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.<br>0b - Receive data not inverted.<br>1b - Receive data inverted. |
| 27 | Receive Wake Up Idle Detect |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RWUID | For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.<br>0b - During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not set when an address does not match.<br>1b - During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does set when an address does not match. |
| 26<br>BRK13 | Break Character Generation Length<br>BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled. A break character can be sent by setting CTRL[SBK] or by writing the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Break character is transmitted with length of 9 to 13 bit times.<br>1b - Break character is transmitted with length of 12 to 15 bit times. |
| 25<br>LBKDE | LIN Break Detection Enable<br>LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.<br>0b - LIN break detect is disabled, normal break character can be detected.<br>1b - LIN break detect is enabled. LIN break character is detected at length of 11 bit times (if M = 0) or 12 (if M = 1) or 13 (M10 = 1). |
| 24<br>RAF | Receiver Active Flag<br>RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.<br>0b - LPUART receiver idle waiting for a start bit.<br>1b - LPUART receiver active (RXD input not idle). |
| 23<br>TDRE | Transmit Data Register Empty Flag<br>When the transmit FIFO is enabled, TDRE will set when the number of datawords in the transmit FIFO (LPUART_DATA) is equal to or less than the number indicated by LPUART_WATER[TXWATER]). To clear TDRE, write to the LPUART data register (LPUART_DATA) until the number of words in the transmit FIFO is greater than the number indicated by LPUART_WATER[TXWATER]. When the transmit FIFO is disabled,TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA).<br>TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.<br>0b - Transmit data buffer full.<br>1b - Transmit data buffer empty. |
| 22<br>TC | Transmission Complete Flag<br>TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK].<br>0b - Transmitter active (sending data, a preamble, or a break).<br>1b - Transmitter idle (transmission activity complete). |
| 21<br>RDRF | Receive Data Register Full Flag<br>When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by LPUART_WATER[RXWATER]. To clear RDRF, read LPUART_DATA until the number of datawords in the receive data buffer is equal to or less than the number indicated by LPUART_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.<br>0b - Receive data buffer empty.<br>1b - Receive data buffer full. |
| 20<br>IDLE | Idle Line Flag<br>IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period.<br>0b - No idle line detected.<br>1b - Idle line was detected. |
| 19<br>OR | Receiver Overrun Flag<br>OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.<br>While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.<br>0b - No overrun.<br>1b - Receive overrun (new LPUART data lost). |
| 18<br>NF | Noise Flag<br>The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.<br>0b - No noise detected.<br>1b - Noise detected in the received character in LPUART_DATA. |
| 17<br>FE | Framing Error Flag<br>FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE.<br>0b - No framing error detected. This does not guarantee the framing is correct.<br>1b - Framing error. |
| 16<br>PF | Parity Error Flag<br>PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.<br>0b - No parity error.<br>1b - Parity error. |
| 15<br>MA1F | Match 1 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.<br>0b - Received data is not equal to MA1<br>1b - Received data is equal to MA1 |
| 14<br>MA2F | Match 2 Flag<br>MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.<br>0b - Received data is not equal to MA2<br>1b - Received data is equal to MA2 |
| 13-0<br>— | Reserved |

## 56.3.2.8 LPUART Control Register (CTRL)

### 56.3.2.8.1 Offset

| Register | Offset |
|---|---|
| CTRL | 18h |

### 56.3.2.8.2 Function

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

### 56.3.2.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MA1IE | MA2IE | 0 | | M7 | IDLECFG | | | LOOPS | DOZEEN | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 56.3.2.8.4 Fields

| Field | Function |
|---|---|
| 31<br>R8T9 | Receive Bit 8 / Transmit Bit 9<br>R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA.<br>T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R8 is a read-only bit and T9 is a write-only bit, the value read is different from the value written. |
| 30<br>R9T8 | Receive Bit 9 / Transmit Bit 8<br>R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA<br>T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R9 is a read-only bit and T8 is a write-only bit, the value read is different from the value written. |
| 29<br>TXDIR | TXD Pin Direction in Single-Wire Mode<br>When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the TXD pin.<br>0b - TXD pin is an input in single-wire mode.<br>1b - TXD pin is an output in single-wire mode. |
| 28<br>TXINV | Transmit Data Inversion<br>Setting this bit reverses the polarity of the transmitted data output.<br>**NOTE:** Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.<br>0b - Transmit data not inverted.<br>1b - Transmit data inverted. |
| 27<br>ORIE | Overrun Interrupt Enable<br>This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0b - OR interrupts disabled; use polling.<br>1b - Hardware interrupt requested when OR is set. |
| 26<br>NEIE | Noise Error Interrupt Enable<br>This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0b - NF interrupts disabled; use polling.<br>1b - Hardware interrupt requested when NF is set. |
| 25<br>FEIE | Framing Error Interrupt Enable<br>This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0b - FE interrupts disabled; use polling.<br>1b - Hardware interrupt requested when FE is set. |
| 24<br>PEIE | Parity Error Interrupt Enable<br>This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0b - PF interrupts disabled; use polling).<br>1b - Hardware interrupt requested when PF is set. |
| 23 | Transmit Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIE | Enables STAT[TDRE] to generate interrupt requests.<br>0b - Hardware interrupts from TDRE disabled; use polling.<br>1b - Hardware interrupt requested when TDRE flag is 1. |
| 22<br>TCIE | Transmission Complete Interrupt Enable for<br>TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0b - Hardware interrupts from TC disabled; use polling.<br>1b - Hardware interrupt requested when TC flag is 1. |
| 21<br>RIE | Receiver Interrupt Enable<br>Enables STAT[RDRF] to generate interrupt requests.<br>0b - Hardware interrupts from RDRF disabled; use polling.<br>1b - Hardware interrupt requested when RDRF flag is 1. |
| 20<br>ILIE | Idle Line Interrupt Enable<br>ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.<br>0b - Hardware interrupts from IDLE disabled; use polling.<br>1b - Hardware interrupt requested when IDLE flag is 1. |
| 19<br>TE | Transmitter Enable<br>Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the TXD pin is tristated.<br>A single idle character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] set.<br>0b - Transmitter disabled.<br>1b - Transmitter enabled. |
| 18<br>RE | Receiver Enable<br>Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).<br>0b - Receiver disabled.<br>1b - Receiver enabled. |
| 17<br>RWU | Receiver Wakeup Control<br>This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.<br>**NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.<br>0b - Normal receiver operation.<br>1b - LPUART receiver in standby waiting for wakeup condition. |
| 16<br>SBK | Send Break<br>Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if LPUART_STAT[BRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK.<br>A single break character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Normal transmitter operation.<br>1b - Queue break character(s) to be sent. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>MA1IE | Match 1 Interrupt Enable<br>0b - MA1F interrupt disabled<br>1b - MA1F interrupt enabled |
| 14<br>MA2IE | Match 2 Interrupt Enable<br>0b - MA2F interrupt disabled<br>1b - MA2F interrupt enabled |
| 13-12<br>— | Reserved |
| 11<br>M7 | 7-Bit Mode Select<br>This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 8-bit to 10-bit data characters.<br>1b - Receiver and transmitter use 7-bit data characters. |
| 10-8<br>IDLECFG | Idle Configuration<br>Configures the number of idle characters that must be received before the IDLE flag is set.<br>000b - 1 idle character<br>001b - 2 idle characters<br>010b - 4 idle characters<br>011b - 8 idle characters<br>100b - 16 idle characters<br>101b - 32 idle characters<br>110b - 64 idle characters<br>111b - 128 idle characters |
| 7<br>LOOPS | Loop Mode Select<br>When LOOPS is set, the RXD pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.<br>0b - Normal operation - RXD and TXD use separate pins.<br>1b - Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit). |
| 6<br>DOZEEN | Doze Enable<br>0b - LPUART is enabled in Doze mode.<br>1b - LPUART is disabled in Doze mode. |
| 5<br>RSRC | Receiver Source Select<br>This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.<br>0b - Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the RXD pin.<br>1b - Single-wire LPUART mode where the TXD pin is connected to the transmitter output and receiver input. |
| 4<br>M | 9-Bit or 8-Bit Mode Select<br>0b - Receiver and transmitter use 8-bit data characters.<br>1b - Receiver and transmitter use 9-bit data characters. |
| 3<br>WAKE | Receiver Wakeup Method Select<br>Determines which condition wakes the LPUART when RWU=1:<br>• Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character, or<br>• An idle condition on the receive pin input signal.<br><br>0b - Configures RWU for idle-line wakeup.<br>1b - Configures RWU with address-mark wakeup. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>ILT | Idle Line Type Select<br>Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>**NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>0b - Idle character bit count starts after start bit.<br>1b - Idle character bit count starts after stop bit. |
| 1<br>PE | Parity Enable<br>Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.<br>0b - No hardware parity generation or checking.<br>1b - Parity enabled. |
| 0<br>PT | Parity Type<br>Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0b - Even parity.<br>1b - Odd parity. |

## 56.3.2.9 LPUART Data Register (DATA)

### 56.3.2.9.1 Offset

| Register | Offset |
|---|---|
| DATA | 1Ch |

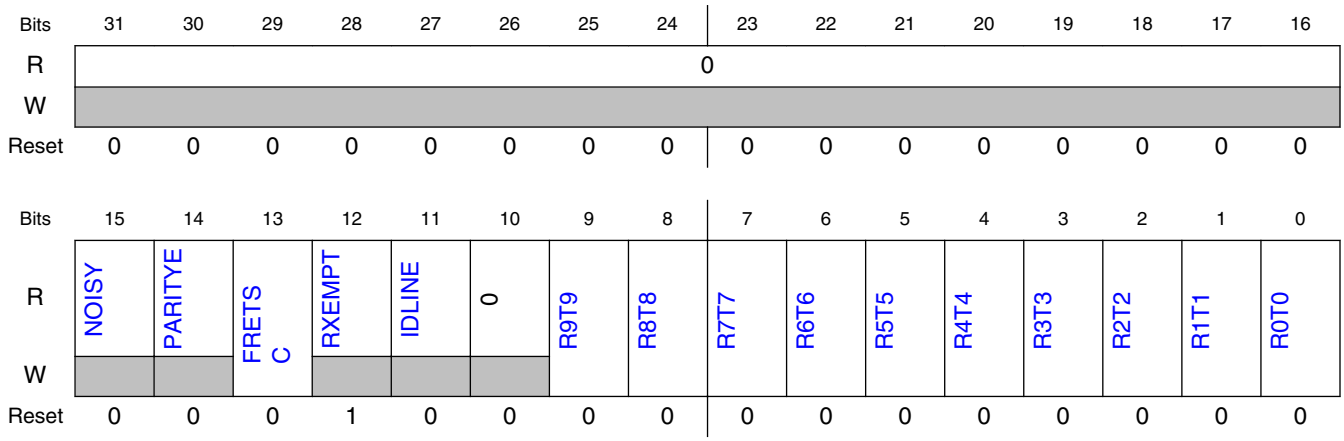
### 56.3.2.9.2 Function

**NOTE**

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

### 56.3.2.9.3 Diagram




### 56.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>NOISY | NOISY<br>The current received dataword contained in DATA[R9:R0] was received with noise.<br>0b - The dataword was received without noise.<br>1b - The data was received with noise. |
| 14<br>PARITYE | PARITYE<br>The current received dataword contained in DATA[R9:R0] was received with a parity error.<br>0b - The dataword was received without a parity error.<br>1b - The dataword was received with a parity error. |
| 13<br>FRETSC | Frame Error / Transmit Special Character<br>For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero.<br>0b - The dataword was received without a frame error on read, or transmit a normal character on write.<br>1b - The dataword was received with a frame error, or transmit an idle or break character on transmit. |
| 12<br>RXEMPT | Receive Buffer Empty<br>Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer contains valid data.<br>1b - Receive buffer is empty, data returned on read is not valid. |
| 11<br>IDLINE | Idle Line<br>Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.<br>0b - Receiver was not idle before receiving this character.<br>1b - Receiver was idle before receiving this character. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>— | Reserved |
| 9<br>R9T9 | R9T9<br>Read receive data buffer 9 or write transmit data buffer 9. |
| 8<br>R8T8 | R8T8<br>Read receive data buffer 8 or write transmit data buffer 8. |
| 7<br>R7T7 | R7T7<br>Read receive data buffer 7 or write transmit data buffer 7. |
| 6<br>R6T6 | R6T6<br>Read receive data buffer 6 or write transmit data buffer 6. |
| 5<br>R5T5 | R5T5<br>Read receive data buffer 5 or write transmit data buffer 5. |
| 4<br>R4T4 | R4T4<br>Read receive data buffer 4 or write transmit data buffer 4. |
| 3<br>R3T3 | R3T3<br>Read receive data buffer 3 or write transmit data buffer 3. |
| 2<br>R2T2 | R2T2<br>Read receive data buffer 2 or write transmit data buffer 2. |
| 1<br>R1T1 | R1T1<br>Read receive data buffer 1 or write transmit data buffer 1. |
| 0<br>R0T0 | R0T0<br>Read receive data buffer 0 or write transmit data buffer 0. |

## 56.3.2.10 LPUART Match Address Register (MATCH)

### 56.3.2.10.1 Offset

| Register | Offset |
|---|---|
| MATCH | 20h |

### 56.3.2.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MA2 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MA1 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.2.10.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>MA2 | Match Address 2<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |
| 15-10<br>— | Reserved |
| 9-0<br>MA1 | Match Address 1<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |

## 56.3.2.11 LPUART Modem IrDA Register (MODIR)

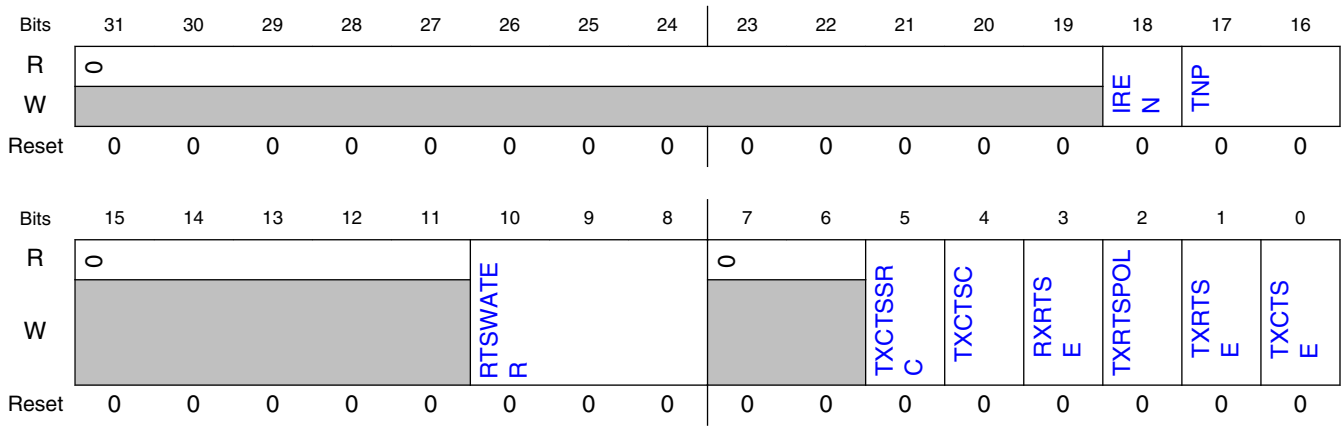
### 56.3.2.11.1 Offset

| Register | Offset |
|---|---|
| MODIR | 24h |

### 56.3.2.11.2 Function

The MODEM register controls options for setting the modem configuration.

### 56.3.2.11.3 Diagram




### 56.3.2.11.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>IREN | Infrared enable<br>Enables/disables the infrared modulation/demodulation. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - IR disabled.<br>1b - IR enabled. |
| 17-16<br>TNP | Transmitter narrow pulse<br>Configures whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse when IR is enabled. This bit should only be changed when the transmitter and receiver are both disabled.<br>The IR pulse width should be configured to less than half of the oversampling ratio. Common pulse widths are 3/16, 1/16, 1/32 or 1/4 of the bit length. These can be configured by selecting the appropriate oversample ratio and pulse width.<br>00b - 1/OSR.<br>01b - 2/OSR.<br>10b - 3/OSR.<br>11b - 4/OSR. |
| 15-11<br>— | Reserved |
| 10-8<br>RTSWATER | Receive RTS Configuration<br>Configures the assertion and negation of the RX RTS_B output. The RX RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to RTSWATER. If RTSWATER is configured to zero, RTS_B negates when the receive FIFO is full. For the purpose of RX RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both RX RTS_B and address/data matching is enabled, then RTS_B could assert at the end of a character if there is not a match. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This field should only be changed when the receiver is disabled. |
| 7-6<br>— | Reserved |
| 5<br>TXCTSSRC | Transmit CTS Source<br>Configures the source of the CTS input.<br>0b - CTS input is the CTS_B pin.<br>1b - CTS input is the inverted Receiver Match result. |
| 4<br>TXCTSC | Transmit CTS Configuration<br>Configures if the CTS state is checked at the start of each character or only when the transmitter is idle.<br>0b - CTS input is sampled at the start of each character.<br>1b - CTS input is sampled when the transmitter is idle. |
| 3<br>RXRTSE | Receiver request-to-send enable<br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Do not set both RXRTSE and TXRTSE.<br>0b - The receiver has no effect on RTS.<br>1b - RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full. |
| 2<br>TXRTSPOL | Transmitter request-to-send polarity<br>Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set. This bit should only be changed when the transmitter is disabled.<br>0b - Transmitter RTS is active low.<br>1b - Transmitter RTS is active high. |
| 1<br>TXRTSE | Transmitter request-to-send enable<br>Controls RTS before and after a transmission. This bit should only be changed when the transmitter is disabled.<br>0b - The transmitter has no effect on RTS.<br>1b - When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. |
| 0<br>TXCTSE | Transmitter clear-to-send enable<br>TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.<br>0b - CTS has no effect on the transmitter.<br>1b - Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission. |

## 56.3.2.12 LPUART FIFO Register (FIFO)

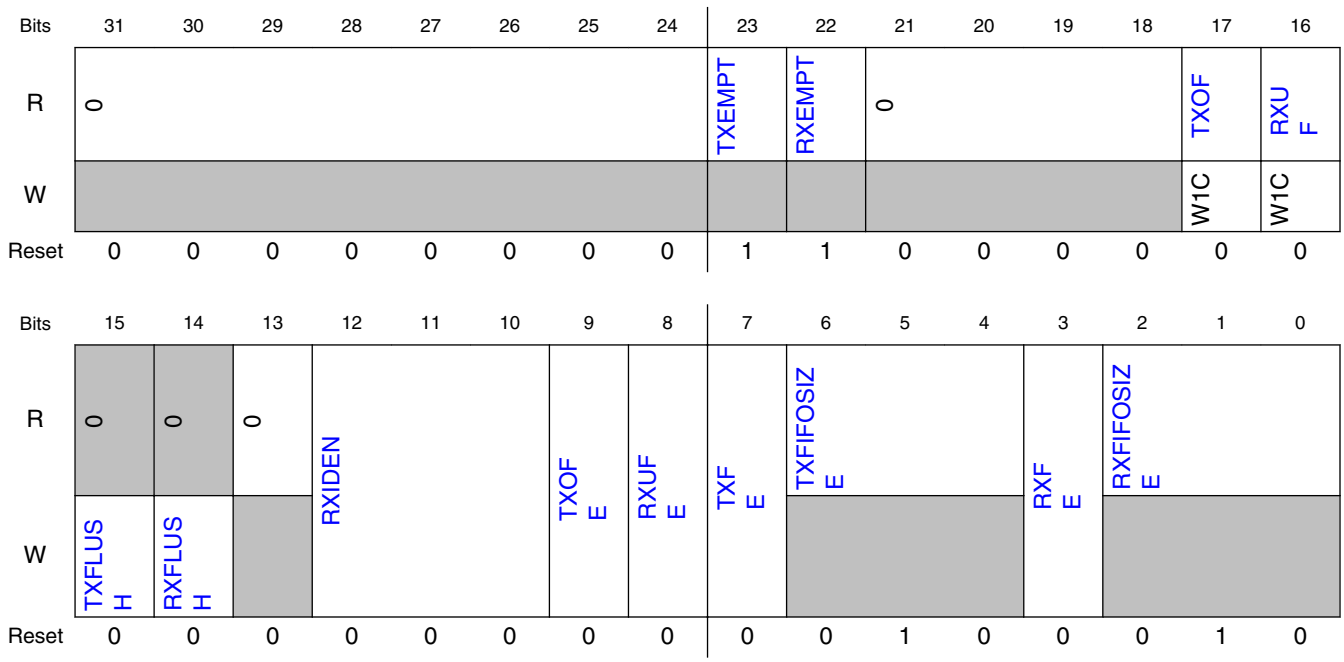
### 56.3.2.12.1 Offset

| Register | Offset |
|---|---|
| FIFO | 28h |

### 56.3.2.12.2 Function

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when CTRL[RE] and CTRL[TE] are cleared/not set and when the data buffer/FIFO is empty.

### 56.3.2.12.3 Diagram




### 56.3.2.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>TXEMPT | Transmit Buffer/FIFO Empty<br>Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register.<br>0b - Transmit buffer is not empty. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Transmit buffer is empty. |
| 22<br>RXEMPT | Receive Buffer/FIFO Empty<br>Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer is not empty.<br>1b - Receive buffer is empty. |
| 21-18<br>— | Reserved |
| 17<br>TXOF | Transmitter Buffer Overflow Flag<br>Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1.<br>0b - No transmit buffer overflow has occurred since the last time the flag was cleared.<br>1b - At least one transmit buffer overflow has occurred since the last time the flag was cleared. |
| 16<br>RXUF | Receiver Buffer Underflow Flag<br>Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1.<br>0b - No receive buffer underflow has occurred since the last time the flag was cleared.<br>1b - At least one receive buffer underflow has occurred since the last time the flag was cleared. |
| 15<br>TXFLUSH | Transmit FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.<br>0b - No flush operation occurs.<br>1b - All data in the transmit FIFO/Buffer is cleared out. |
| 14<br>RXFLUSH | Receive FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.<br>0b - No flush operation occurs.<br>1b - All data in the receive FIFO/buffer is cleared out. |
| 13<br>— | Reserved |
| 12-10<br>RXIDEN | Receiver Idle Empty Enable<br>When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty.<br>000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle.<br>001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character.<br>010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters.<br>011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters.<br>100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters.<br>101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters.<br>110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters.<br>111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters. |
| 9<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br>When this field is set, the TXOF flag generates an interrupt to the host.<br>0b - TXOF flag does not generate an interrupt to the host.<br>1b - TXOF flag generates an interrupt to the host. |
| 8 | Receive FIFO Underflow Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXUFE | When this field is set, the RXUF flag generates an interrupt to the host.<br>0b - RXUF flag does not generate an interrupt to the host.<br>1b - RXUF flag generates an interrupt to the host. |
| 7<br>TXFE | Transmit FIFO Enable<br>When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).<br>1b - Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE. |
| 6-4<br>TXFIFOSIZE | Transmit FIFO. Buffer Depth<br>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.<br>000b - Transmit FIFO/Buffer depth = 1 dataword.<br>001b - Transmit FIFO/Buffer depth = 4 datawords.<br>010b - Transmit FIFO/Buffer depth = 8 datawords.<br>011b - Transmit FIFO/Buffer depth = 16 datawords.<br>100b - Transmit FIFO/Buffer depth = 32 datawords.<br>101b - Transmit FIFO/Buffer depth = 64 datawords.<br>110b - Transmit FIFO/Buffer depth = 128 datawords.<br>111b - Transmit FIFO/Buffer depth = 256 datawords |
| 3<br>RXFE | Receive FIFO Enable<br>When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)<br>1b - Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE. |
| 2-0<br>RXFIFOSIZE | Receive FIFO. Buffer Depth<br>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.<br>000b - Receive FIFO/Buffer depth = 1 dataword.<br>001b - Receive FIFO/Buffer depth = 4 datawords.<br>010b - Receive FIFO/Buffer depth = 8 datawords.<br>011b - Receive FIFO/Buffer depth = 16 datawords.<br>100b - Receive FIFO/Buffer depth = 32 datawords.<br>101b - Receive FIFO/Buffer depth = 64 datawords.<br>110b - Receive FIFO/Buffer depth = 128 datawords.<br>111b - Receive FIFO/Buffer depth = 256 datawords. |

## 56.3.2.13 LPUART Watermark Register (WATER)

### 56.3.2.13.1 Offset

| Register | Offset |
|---|---|
| WATER | 2Ch |

### 56.3.2.13.2 Function

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when CTRL[TE] is not set.

### 56.3.2.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RXCOUNT | | | | 0 | | | | | RXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TXCOUNT | | | | 0 | | | | | TXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.2.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>RXCOUNT | Receive Counter<br>The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer. |
| 23-19<br>— | Reserved |
| 18-16<br>RXWATER | Receive Watermark<br>When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0. |
| 15-12<br>— | Reserved |
| 11-8<br>TXCOUNT | Transmit Counter<br>The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer. |
| 7-3 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 2-0<br>TXWATER | Transmit Watermark<br>When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE]. |

## 56.3.3 LPUART2 register descriptions

### 56.3.3.1 LPUART2 Memory map

LPUART2 base address: 4004_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0401_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0303h |
| 8h | LPUART Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| Ch | LPUART Pin Configuration Register (PINCFG) | 32 | RW | 0000_0000h |
| 10h | LPUART Baud Rate Register (BAUD) | 32 | RW | 0F00_0004h |
| 14h | LPUART Status Register (STAT) | 32 | RW | 00C0_0000h |
| 18h | LPUART Control Register (CTRL) | 32 | RW | 0000_0000h |
| 1Ch | LPUART Data Register (DATA) | 32 | RW | 0000_1000h |
| 20h | LPUART Match Address Register (MATCH) | 32 | RW | 0000_0000h |
| 24h | LPUART Modem IrDA Register (MODIR) | 32 | RW | 0000_0000h |
| 28h | LPUART FIFO Register (FIFO) | 32 | RW | 00C0_0022h |
| 2Ch | LPUART Watermark Register (WATER) | 32 | RW | 0000_0000h |

### 56.3.3.2 Version ID Register (VERID)

#### 56.3.3.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 56.3.3.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

#### 56.3.3.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.3.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with MODEM/IrDA support. |

### 56.3.3.3 Parameter Register (PARAM)

#### 56.3.3.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 56.3.3.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.3.3.3 Fields

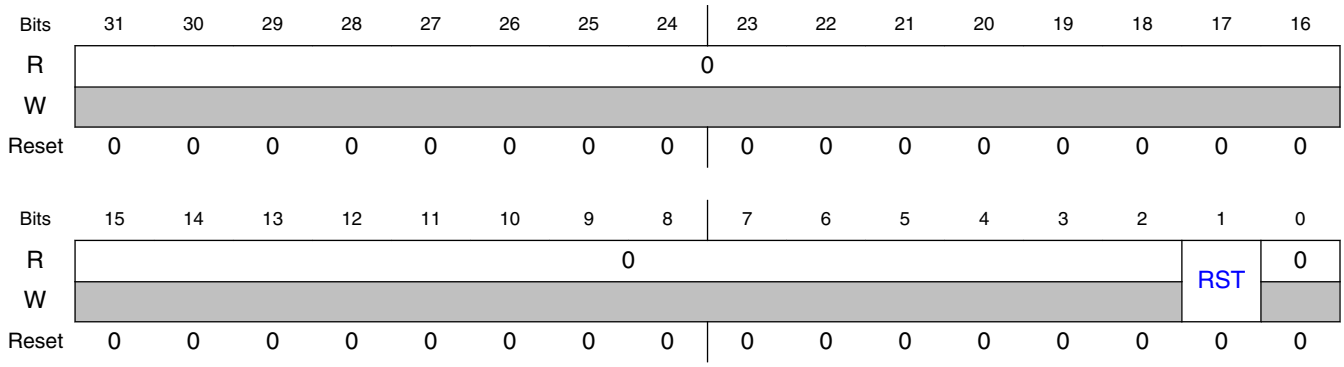
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>RXFIFO | Receive FIFO Size<br>The number of words in the receive FIFO is 2^RXFIFO. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>The number of words in the transmit FIFO is 2^TXFIFO. |

### 56.3.3.4 LPUART Global Register (GLOBAL)

#### 56.3.3.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 56.3.3.4.2 Diagram




### 56.3.3.4.3 Fields

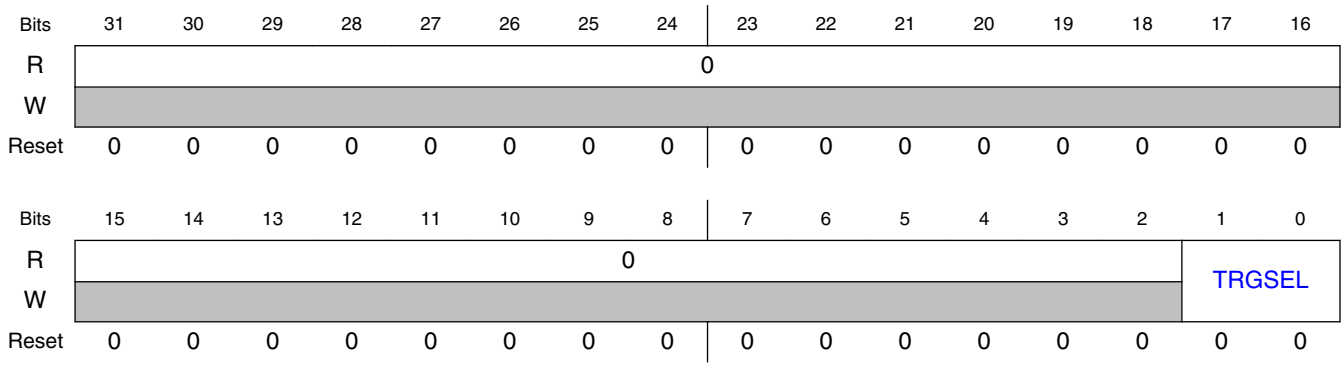
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>— | Reserved |

## 56.3.3.5 LPUART Pin Configuration Register (PINCFG)

### 56.3.3.5.1 Offset

| Register | Offset |
|---|---|
| PINCFG | Ch |

#### 56.3.3.5.2 Diagram




#### 56.3.3.5.3 Fields

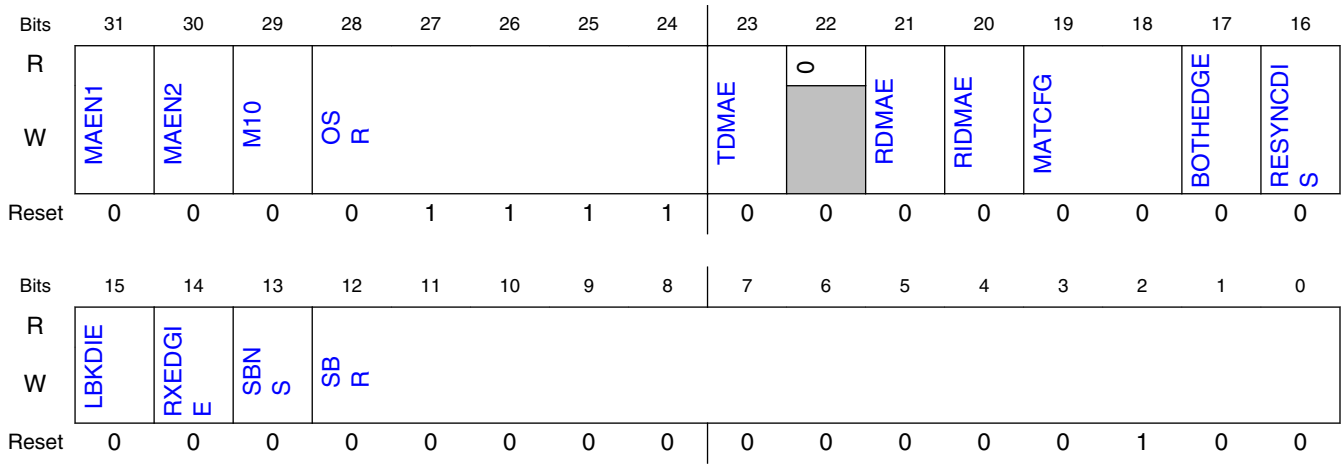
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>TRGSEL | Trigger Select<br>Configures the input trigger usage. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Input trigger is disabled.<br>01b - Input trigger is used instead of RXD pin input.<br>10b - Input trigger is used instead of CTS_B pin input.<br>11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger. |

### 56.3.3.6 LPUART Baud Rate Register (BAUD)

#### 56.3.3.6.1 Offset

| Register | Offset |
|---|---|
| BAUD | 10h |

## 56.3.3.6.2 Diagram




## 56.3.3.6.3 Fields

| Field | Function |
|---|---|
| 31<br>MAEN1 | Match Address Mode Enable 1<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA1]. |
| 30<br>MAEN2 | Match Address Mode Enable 2<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA2]. |
| 29<br>M10 | 10-bit Mode select<br>The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 7-bit to 9-bit data characters.<br>1b - Receiver and transmitter use 10-bit data characters. |
| 28-24<br>OSR | Oversampling Ratio<br>This field configures the oversampling ratio for the receiver. This field should only be changed when the transmitter and receiver are both disabled.<br>00000b - Writing 0 to this field will result in an oversampling ratio of 16<br>00001b - Reserved<br>00010b - Reserved<br>00011b - Oversampling ratio of 4, requires BOTHEDGE to be set.<br>00100b - Oversampling ratio of 5, requires BOTHEDGE to be set.<br>00101b - Oversampling ratio of 6, requires BOTHEDGE to be set.<br>00110b - Oversampling ratio of 7, requires BOTHEDGE to be set.<br>00111b - Oversampling ratio of 8.<br>01000b - Oversampling ratio of 9.<br>01001b - Oversampling ratio of 10.<br>01010b - Oversampling ratio of 11.<br>01011b - Oversampling ratio of 12.<br>01100b - Oversampling ratio of 13.<br>01101b - Oversampling ratio of 14.<br>01110b - Oversampling ratio of 15.<br>01111b - Oversampling ratio of 16.<br>10000b - Oversampling ratio of 17. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 10001b - Oversampling ratio of 18.<br>10010b - Oversampling ratio of 19.<br>10011b - Oversampling ratio of 20.<br>10100b - Oversampling ratio of 21.<br>10101b - Oversampling ratio of 22.<br>10110b - Oversampling ratio of 23.<br>10111b - Oversampling ratio of 24.<br>11000b - Oversampling ratio of 25.<br>11001b - Oversampling ratio of 26.<br>11010b - Oversampling ratio of 27.<br>11011b - Oversampling ratio of 28.<br>11100b - Oversampling ratio of 29.<br>11101b - Oversampling ratio of 30.<br>11110b - Oversampling ratio of 31.<br>11111b - Oversampling ratio of 32. |
| 23<br>TDMAE | Transmitter DMA Enable<br>TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 22<br>— | Reserved |
| 21<br>RDMAE | Receiver Full DMA Enable<br>RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 20<br>RIDMAE | Receiver Idle DMA Enable<br>RIDMAE configures the receiver idle flag, LPUART_STAT[IDLE], to generate a DMA request. When this bit is set, reading LPUART_DATA when either DATA[RXEMPT] or DATA[IDLINE] bit is set, will generate an End Of Packet response until the completion of the existing DMA transfer. During an End of Packet response, reading the LPUART_DATA register will return 0x0000_33FF and does not pull data from the FIFO.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 19-18<br>MATCFG | Match Configuration<br>Configures the match addressing mode used. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Address Match Wakeup<br>01b - Idle Match Wakeup<br>10b - Match On and Match Off<br>11b - Enables RWU on Data Match and Match On/Off for transmitter CTS input |
| 17<br>BOTHEDGE | Both Edge Sampling<br>Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.<br>0b - Receiver samples input data using the rising edge of the baud rate clock.<br>1b - Receiver samples input data using the rising and falling edge of the baud rate clock. |
| 16<br>RESYNCDIS | Resynchronization Disable |

*Table continues on the next page...*

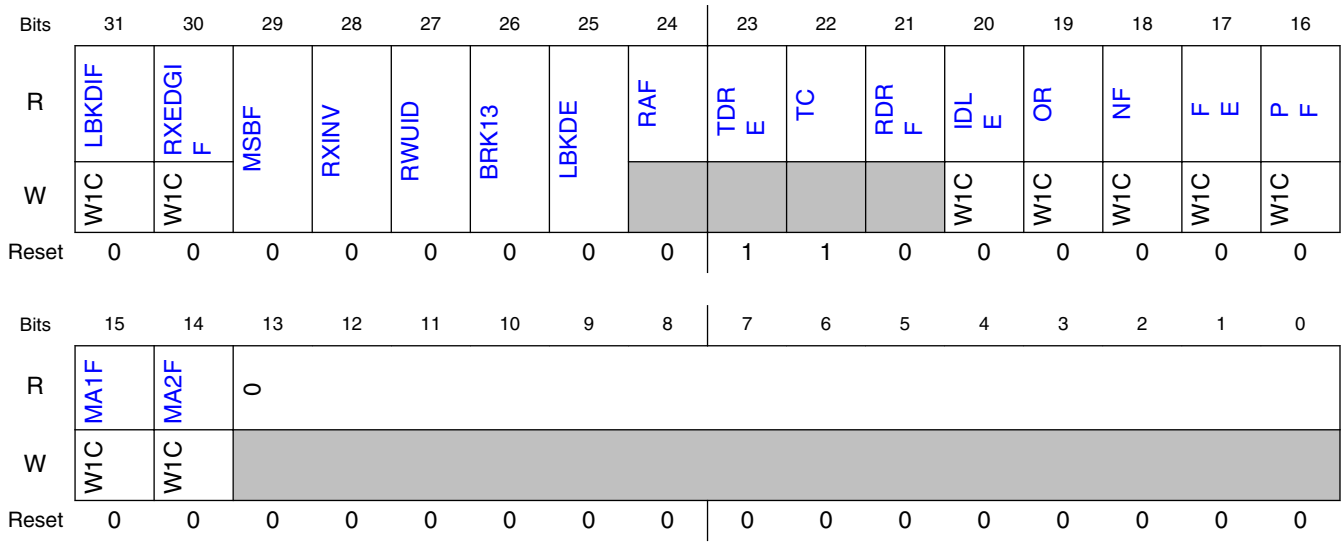
| Field | Function |
|---|---|
| | When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.<br>0b - Resynchronization during received data word is supported<br>1b - Resynchronization during received data word is disabled |
| 15<br>LBKDIE | LIN Break Detect Interrupt Enable<br>LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.<br>0b - Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling).<br>1b - Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1. |
| 14<br>RXEDGIE | RX Input Active Edge Interrupt Enable<br>Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.<br>0b - Hardware interrupts from LPUART_STAT[RXEDGIF] disabled.<br>1b - Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1. |
| 13<br>SBNS | Stop Bit Number Select<br>SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - One stop bit.<br>1b - Two stop bits. |
| 12-0<br>SBR | Baud Rate Modulo Divisor.<br>The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0). |

## 56.3.3.7 LPUART Status Register (STAT)

### 56.3.3.7.1 Offset

| Register | Offset |
|---|---|
| STAT | 14h |

### 56.3.3.7.2 Diagram




### 56.3.3.7.3 Fields

| Field | Function |
|---|---|
| 31<br>LBKDIF | LIN Break Detect Interrupt Flag<br>LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.<br>0b - No LIN break character has been detected.<br>1b - LIN break character has been detected. |
| 30<br>RXEDGIF | RXD Pin Active Edge Interrupt Flag<br>RXEDGIF is set whenever the receiver is enabled and an active edge, falling if RXINV = 0, rising if RXINV=1, on the RXD pin occurs. RXEDGIF is cleared by writing a 1 to it.<br>0b - No active edge on the receive pin has occurred.<br>1b - An active edge on the receive pin has occurred. |
| 29<br>MSBF | MSB First<br>Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0.<br>1b - MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE]. |
| 28<br>RXINV | Receive Data Inversion<br>Setting this bit reverses the polarity of the received data input. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.<br>0b - Receive data not inverted.<br>1b - Receive data inverted. |
| 27 | Receive Wake Up Idle Detect |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RWUID | For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.<br>0b - During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not set when an address does not match.<br>1b - During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does set when an address does not match. |
| 26<br>BRK13 | Break Character Generation Length<br>BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled. A break character can be sent by setting CTRL[SBK] or by writing the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Break character is transmitted with length of 9 to 13 bit times.<br>1b - Break character is transmitted with length of 12 to 15 bit times. |
| 25<br>LBKDE | LIN Break Detection Enable<br>LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.<br>0b - LIN break detect is disabled, normal break character can be detected.<br>1b - LIN break detect is enabled. LIN break character is detected at length of 11 bit times (if M = 0) or 12 (if M = 1) or 13 (M10 = 1). |
| 24<br>RAF | Receiver Active Flag<br>RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.<br>0b - LPUART receiver idle waiting for a start bit.<br>1b - LPUART receiver active (RXD input not idle). |
| 23<br>TDRE | Transmit Data Register Empty Flag<br>When the transmit FIFO is enabled, TDRE will set when the number of datawords in the transmit FIFO (LPUART_DATA) is equal to or less than the number indicated by LPUART_WATER[TXWATER]). To clear TDRE, write to the LPUART data register (LPUART_DATA) until the number of words in the transmit FIFO is greater than the number indicated by LPUART_WATER[TXWATER]. When the transmit FIFO is disabled,TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA).<br>TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.<br>0b - Transmit data buffer full.<br>1b - Transmit data buffer empty. |
| 22<br>TC | Transmission Complete Flag<br>TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK].<br>0b - Transmitter active (sending data, a preamble, or a break).<br>1b - Transmitter idle (transmission activity complete). |
| 21<br>RDRF | Receive Data Register Full Flag<br>When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by LPUART_WATER[RXWATER]. To clear RDRF, read LPUART_DATA until the number of datawords in the receive data buffer is equal to or less than the number indicated by LPUART_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.<br>0b - Receive data buffer empty.<br>1b - Receive data buffer full. |
| 20<br>IDLE | Idle Line Flag<br>IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period.<br>0b - No idle line detected.<br>1b - Idle line was detected. |
| 19<br>OR | Receiver Overrun Flag<br>OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.<br>While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.<br>0b - No overrun.<br>1b - Receive overrun (new LPUART data lost). |
| 18<br>NF | Noise Flag<br>The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.<br>0b - No noise detected.<br>1b - Noise detected in the received character in LPUART_DATA. |
| 17<br>FE | Framing Error Flag<br>FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE.<br>0b - No framing error detected. This does not guarantee the framing is correct.<br>1b - Framing error. |
| 16<br>PF | Parity Error Flag<br>PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.<br>0b - No parity error.<br>1b - Parity error. |
| 15<br>MA1F | Match 1 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.<br>0b - Received data is not equal to MA1<br>1b - Received data is equal to MA1 |
| 14<br>MA2F | Match 2 Flag<br>MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.<br>0b - Received data is not equal to MA2<br>1b - Received data is equal to MA2 |
| 13-0<br>— | Reserved |

## 56.3.3.8 LPUART Control Register (CTRL)

### 56.3.3.8.1 Offset

| Register | Offset |
|---|---|
| CTRL | 18h |

### 56.3.3.8.2 Function

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

### 56.3.3.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MA1IE | MA2IE | 0 | | M7 | IDLECFG | | | LOOPS | DOZEEN | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.3.8.4 Fields

| Field | Function |
|---|---|
| 31<br>R8T9 | Receive Bit 8 / Transmit Bit 9<br>R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA.<br>T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R8 is a read-only bit and T9 is a write-only bit, the value read is different from the value written. |
| 30<br>R9T8 | Receive Bit 9 / Transmit Bit 8<br>R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA<br>T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R9 is a read-only bit and T8 is a write-only bit, the value read is different from the value written. |
| 29<br>TXDIR | TXD Pin Direction in Single-Wire Mode<br>When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the TXD pin.<br>0b - TXD pin is an input in single-wire mode.<br>1b - TXD pin is an output in single-wire mode. |
| 28<br>TXINV | Transmit Data Inversion<br>Setting this bit reverses the polarity of the transmitted data output.<br>**NOTE:** Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.<br>0b - Transmit data not inverted.<br>1b - Transmit data inverted. |
| 27<br>ORIE | Overrun Interrupt Enable<br>This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0b - OR interrupts disabled; use polling.<br>1b - Hardware interrupt requested when OR is set. |
| 26<br>NEIE | Noise Error Interrupt Enable<br>This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0b - NF interrupts disabled; use polling.<br>1b - Hardware interrupt requested when NF is set. |
| 25<br>FEIE | Framing Error Interrupt Enable<br>This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0b - FE interrupts disabled; use polling.<br>1b - Hardware interrupt requested when FE is set. |
| 24<br>PEIE | Parity Error Interrupt Enable<br>This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0b - PF interrupts disabled; use polling).<br>1b - Hardware interrupt requested when PF is set. |
| 23 | Transmit Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIE | Enables STAT[TDRE] to generate interrupt requests.<br>0b - Hardware interrupts from TDRE disabled; use polling.<br>1b - Hardware interrupt requested when TDRE flag is 1. |
| 22<br>TCIE | Transmission Complete Interrupt Enable for<br>TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0b - Hardware interrupts from TC disabled; use polling.<br>1b - Hardware interrupt requested when TC flag is 1. |
| 21<br>RIE | Receiver Interrupt Enable<br>Enables STAT[RDRF] to generate interrupt requests.<br>0b - Hardware interrupts from RDRF disabled; use polling.<br>1b - Hardware interrupt requested when RDRF flag is 1. |
| 20<br>ILIE | Idle Line Interrupt Enable<br>ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.<br>0b - Hardware interrupts from IDLE disabled; use polling.<br>1b - Hardware interrupt requested when IDLE flag is 1. |
| 19<br>TE | Transmitter Enable<br>Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the TXD pin is tristated.<br>A single idle character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] set.<br>0b - Transmitter disabled.<br>1b - Transmitter enabled. |
| 18<br>RE | Receiver Enable<br>Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).<br>0b - Receiver disabled.<br>1b - Receiver enabled. |
| 17<br>RWU | Receiver Wakeup Control<br>This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.<br>**NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.<br>0b - Normal receiver operation.<br>1b - LPUART receiver in standby waiting for wakeup condition. |
| 16<br>SBK | Send Break<br>Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if LPUART_STAT[BRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK.<br>A single break character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Normal transmitter operation.<br>1b - Queue break character(s) to be sent. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>MA1IE | Match 1 Interrupt Enable<br>0b - MA1F interrupt disabled<br>1b - MA1F interrupt enabled |
| 14<br>MA2IE | Match 2 Interrupt Enable<br>0b - MA2F interrupt disabled<br>1b - MA2F interrupt enabled |
| 13-12<br>— | Reserved |
| 11<br>M7 | 7-Bit Mode Select<br>This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 8-bit to 10-bit data characters.<br>1b - Receiver and transmitter use 7-bit data characters. |
| 10-8<br>IDLECFG | Idle Configuration<br>Configures the number of idle characters that must be received before the IDLE flag is set.<br>000b - 1 idle character<br>001b - 2 idle characters<br>010b - 4 idle characters<br>011b - 8 idle characters<br>100b - 16 idle characters<br>101b - 32 idle characters<br>110b - 64 idle characters<br>111b - 128 idle characters |
| 7<br>LOOPS | Loop Mode Select<br>When LOOPS is set, the RXD pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.<br>0b - Normal operation - RXD and TXD use separate pins.<br>1b - Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit). |
| 6<br>DOZEEN | Doze Enable<br>0b - LPUART is enabled in Doze mode.<br>1b - LPUART is disabled in Doze mode. |
| 5<br>RSRC | Receiver Source Select<br>This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.<br>0b - Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the RXD pin.<br>1b - Single-wire LPUART mode where the TXD pin is connected to the transmitter output and receiver input. |
| 4<br>M | 9-Bit or 8-Bit Mode Select<br>0b - Receiver and transmitter use 8-bit data characters.<br>1b - Receiver and transmitter use 9-bit data characters. |
| 3<br>WAKE | Receiver Wakeup Method Select<br>Determines which condition wakes the LPUART when RWU=1:<br>• Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character, or<br>• An idle condition on the receive pin input signal.<br><br>0b - Configures RWU for idle-line wakeup.<br>1b - Configures RWU with address-mark wakeup. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>ILT | Idle Line Type Select<br>Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>**NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>0b - Idle character bit count starts after start bit.<br>1b - Idle character bit count starts after stop bit. |
| 1<br>PE | Parity Enable<br>Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.<br>0b - No hardware parity generation or checking.<br>1b - Parity enabled. |
| 0<br>PT | Parity Type<br>Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0b - Even parity.<br>1b - Odd parity. |

## 56.3.3.9 LPUART Data Register (DATA)

### 56.3.3.9.1 Offset

| Register | Offset |
|---|---|
| DATA | 1Ch |

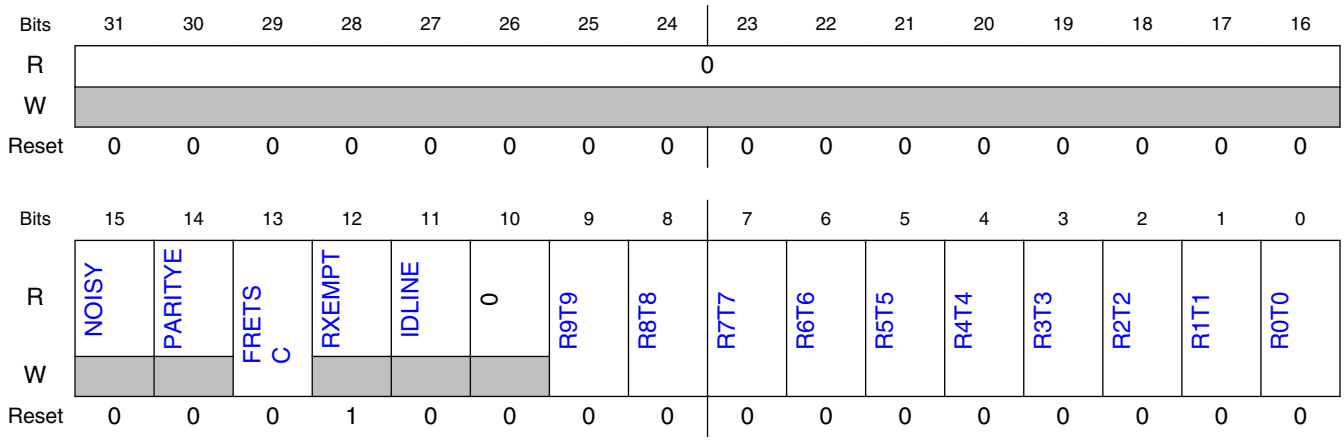
### 56.3.3.9.2 Function

**NOTE**

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

### 56.3.3.9.3 Diagram




### 56.3.3.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>NOISY | NOISY<br>The current received dataword contained in DATA[R9:R0] was received with noise.<br>0b - The dataword was received without noise.<br>1b - The data was received with noise. |
| 14<br>PARITYE | PARITYE<br>The current received dataword contained in DATA[R9:R0] was received with a parity error.<br>0b - The dataword was received without a parity error.<br>1b - The dataword was received with a parity error. |
| 13<br>FRETSC | Frame Error / Transmit Special Character<br>For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero.<br>0b - The dataword was received without a frame error on read, or transmit a normal character on write.<br>1b - The dataword was received with a frame error, or transmit an idle or break character on transmit. |
| 12<br>RXEMPT | Receive Buffer Empty<br>Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer contains valid data.<br>1b - Receive buffer is empty, data returned on read is not valid. |
| 11<br>IDLINE | Idle Line<br>Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.<br>0b - Receiver was not idle before receiving this character.<br>1b - Receiver was idle before receiving this character. |

*Table continues on the next page...*

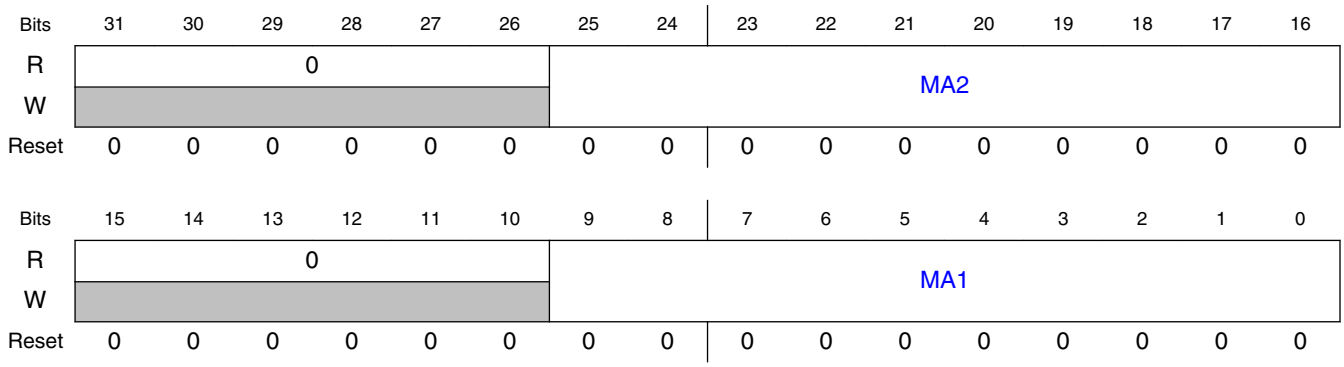
| Field | Function |
|---|---|
| 10<br>— | Reserved |
| 9<br>R9T9 | R9T9<br>Read receive data buffer 9 or write transmit data buffer 9. |
| 8<br>R8T8 | R8T8<br>Read receive data buffer 8 or write transmit data buffer 8. |
| 7<br>R7T7 | R7T7<br>Read receive data buffer 7 or write transmit data buffer 7. |
| 6<br>R6T6 | R6T6<br>Read receive data buffer 6 or write transmit data buffer 6. |
| 5<br>R5T5 | R5T5<br>Read receive data buffer 5 or write transmit data buffer 5. |
| 4<br>R4T4 | R4T4<br>Read receive data buffer 4 or write transmit data buffer 4. |
| 3<br>R3T3 | R3T3<br>Read receive data buffer 3 or write transmit data buffer 3. |
| 2<br>R2T2 | R2T2<br>Read receive data buffer 2 or write transmit data buffer 2. |
| 1<br>R1T1 | R1T1<br>Read receive data buffer 1 or write transmit data buffer 1. |
| 0<br>R0T0 | R0T0<br>Read receive data buffer 0 or write transmit data buffer 0. |

## 56.3.3.10 LPUART Match Address Register (MATCH)

### 56.3.3.10.1 Offset

| Register | Offset |
|---|---|
| MATCH | 20h |

### 56.3.3.10.2 Diagram




### 56.3.3.10.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>MA2 | Match Address 2<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |
| 15-10<br>— | Reserved |
| 9-0<br>MA1 | Match Address 1<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |

## 56.3.3.11 LPUART Modem IrDA Register (MODIR)

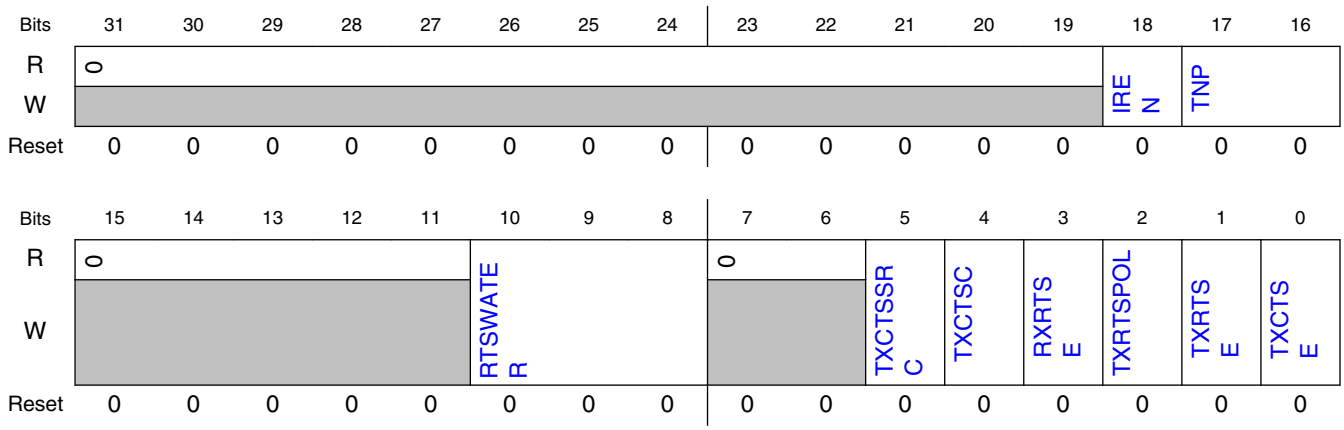
### 56.3.3.11.1 Offset

| Register | Offset |
|---|---|
| MODIR | 24h |

### 56.3.3.11.2 Function

The MODEM register controls options for setting the modem configuration.

## 56.3.3.11.3 Diagram




## 56.3.3.11.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>IREN | Infrared enable<br>Enables/disables the infrared modulation/demodulation. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - IR disabled.<br>1b - IR enabled. |
| 17-16<br>TNP | Transmitter narrow pulse<br>Configures whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse when IR is enabled. This bit should only be changed when the transmitter and receiver are both disabled.<br>The IR pulse width should be configured to less than half of the oversampling ratio. Common pulse widths are 3/16, 1/16, 1/32 or 1/4 of the bit length. These can be configured by selecting the appropriate oversample ratio and pulse width.<br>00b - 1/OSR.<br>01b - 2/OSR.<br>10b - 3/OSR.<br>11b - 4/OSR. |
| 15-11<br>— | Reserved |
| 10-8<br>RTSWATER | Receive RTS Configuration<br>Configures the assertion and negation of the RX RTS_B output. The RX RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to RTSWATER. If RTSWATER is configured to zero, RTS_B negates when the receive FIFO is full. For the purpose of RX RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both RX RTS_B and address/data matching is enabled, then RTS_B could assert at the end of a character if there is not a match. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This field should only be changed when the receiver is disabled. |
| 7-6<br>— | Reserved |
| 5<br>TXCTSSRC | Transmit CTS Source<br>Configures the source of the CTS input.<br>0b - CTS input is the CTS_B pin.<br>1b - CTS input is the inverted Receiver Match result. |
| 4<br>TXCTSC | Transmit CTS Configuration<br>Configures if the CTS state is checked at the start of each character or only when the transmitter is idle.<br>0b - CTS input is sampled at the start of each character.<br>1b - CTS input is sampled when the transmitter is idle. |
| 3<br>RXRTSE | Receiver request-to-send enable<br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Do not set both RXRTSE and TXRTSE.<br>0b - The receiver has no effect on RTS.<br>1b - RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full. |
| 2<br>TXRTSPOL | Transmitter request-to-send polarity<br>Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set. This bit should only be changed when the transmitter is disabled.<br>0b - Transmitter RTS is active low.<br>1b - Transmitter RTS is active high. |
| 1<br>TXRTSE | Transmitter request-to-send enable<br>Controls RTS before and after a transmission. This bit should only be changed when the transmitter is disabled.<br>0b - The transmitter has no effect on RTS.<br>1b - When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. |
| 0<br>TXCTSE | Transmitter clear-to-send enable<br>TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.<br>0b - CTS has no effect on the transmitter.<br>1b - Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission. |

## 56.3.3.12 LPUART FIFO Register (FIFO)

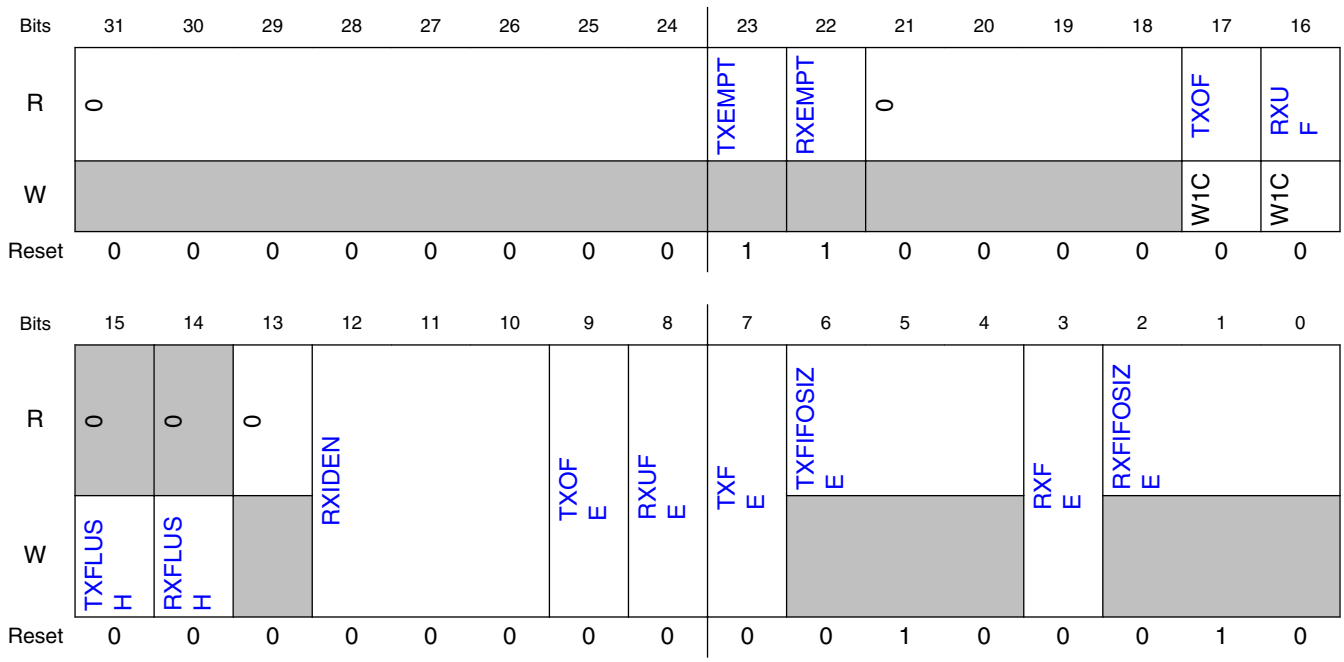
### 56.3.3.12.1 Offset

| Register | Offset |
|---|---|
| FIFO | 28h |

### 56.3.3.12.2 Function

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when CTRL[RE] and CTRL[TE] are cleared/not set and when the data buffer/FIFO is empty.

### 56.3.3.12.3 Diagram




### 56.3.3.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>TXEMPT | Transmit Buffer/FIFO Empty<br>Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register.<br>0b - Transmit buffer is not empty. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Transmit buffer is empty. |
| 22<br>RXEMPT | Receive Buffer/FIFO Empty<br>Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer is not empty.<br>1b - Receive buffer is empty. |
| 21-18<br>— | Reserved |
| 17<br>TXOF | Transmitter Buffer Overflow Flag<br>Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1.<br>0b - No transmit buffer overflow has occurred since the last time the flag was cleared.<br>1b - At least one transmit buffer overflow has occurred since the last time the flag was cleared. |
| 16<br>RXUF | Receiver Buffer Underflow Flag<br>Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1.<br>0b - No receive buffer underflow has occurred since the last time the flag was cleared.<br>1b - At least one receive buffer underflow has occurred since the last time the flag was cleared. |
| 15<br>TXFLUSH | Transmit FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.<br>0b - No flush operation occurs.<br>1b - All data in the transmit FIFO/Buffer is cleared out. |
| 14<br>RXFLUSH | Receive FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.<br>0b - No flush operation occurs.<br>1b - All data in the receive FIFO/buffer is cleared out. |
| 13<br>— | Reserved |
| 12-10<br>RXIDEN | Receiver Idle Empty Enable<br>When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty.<br>000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle.<br>001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character.<br>010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters.<br>011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters.<br>100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters.<br>101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters.<br>110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters.<br>111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters. |
| 9<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br>When this field is set, the TXOF flag generates an interrupt to the host.<br>0b - TXOF flag does not generate an interrupt to the host.<br>1b - TXOF flag generates an interrupt to the host. |
| 8 | Receive FIFO Underflow Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXUFE | When this field is set, the RXUF flag generates an interrupt to the host.<br>0b - RXUF flag does not generate an interrupt to the host.<br>1b - RXUF flag generates an interrupt to the host. |
| 7<br>TXFE | Transmit FIFO Enable<br>When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).<br>1b - Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE. |
| 6-4<br>TXFIFOSIZE | Transmit FIFO. Buffer Depth<br>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.<br>000b - Transmit FIFO/Buffer depth = 1 dataword.<br>001b - Transmit FIFO/Buffer depth = 4 datawords.<br>010b - Transmit FIFO/Buffer depth = 8 datawords.<br>011b - Transmit FIFO/Buffer depth = 16 datawords.<br>100b - Transmit FIFO/Buffer depth = 32 datawords.<br>101b - Transmit FIFO/Buffer depth = 64 datawords.<br>110b - Transmit FIFO/Buffer depth = 128 datawords.<br>111b - Transmit FIFO/Buffer depth = 256 datawords |
| 3<br>RXFE | Receive FIFO Enable<br>When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)<br>1b - Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE. |
| 2-0<br>RXFIFOSIZE | Receive FIFO. Buffer Depth<br>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.<br>000b - Receive FIFO/Buffer depth = 1 dataword.<br>001b - Receive FIFO/Buffer depth = 4 datawords.<br>010b - Receive FIFO/Buffer depth = 8 datawords.<br>011b - Receive FIFO/Buffer depth = 16 datawords.<br>100b - Receive FIFO/Buffer depth = 32 datawords.<br>101b - Receive FIFO/Buffer depth = 64 datawords.<br>110b - Receive FIFO/Buffer depth = 128 datawords.<br>111b - Receive FIFO/Buffer depth = 256 datawords. |

## 56.3.3.13 LPUART Watermark Register (WATER)

### 56.3.3.13.1 Offset

| Register | Offset |
|---|---|
| WATER | 2Ch |

### 56.3.3.13.2 Function

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when CTRL[TE] is not set.

### 56.3.3.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RXCOUNT | | | | 0 | | | | | RXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TXCOUNT | | | | 0 | | | | | TXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.3.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>RXCOUNT | Receive Counter<br>The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer. |
| 23-19<br>— | Reserved |
| 18-16<br>RXWATER | Receive Watermark<br>When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0. |
| 15-12<br>— | Reserved |
| 11-8<br>TXCOUNT | Transmit Counter<br>The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer. |
| 7-3 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 2-0<br>TXWATER | Transmit Watermark<br>When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE]. |

## 56.3.4 LPUART3 register descriptions

### 56.3.4.1 LPUART3 Memory map

LPUART3 base address: 4103_6000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0401_0003h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0303h |
| 8h | LPUART Global Register (GLOBAL) | 32 | RW | 0000_0000h |
| Ch | LPUART Pin Configuration Register (PINCFG) | 32 | RW | 0000_0000h |
| 10h | LPUART Baud Rate Register (BAUD) | 32 | RW | 0F00_0004h |
| 14h | LPUART Status Register (STAT) | 32 | RW | 00C0_0000h |
| 18h | LPUART Control Register (CTRL) | 32 | RW | 0000_0000h |
| 1Ch | LPUART Data Register (DATA) | 32 | RW | 0000_1000h |
| 20h | LPUART Match Address Register (MATCH) | 32 | RW | 0000_0000h |
| 24h | LPUART Modem IrDA Register (MODIR) | 32 | RW | 0000_0000h |
| 28h | LPUART FIFO Register (FIFO) | 32 | RW | 00C0_0022h |
| 2Ch | LPUART Watermark Register (WATER) | 32 | RW | 0000_0000h |

### 56.3.4.2 Version ID Register (VERID)

#### 56.3.4.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 56.3.4.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

#### 56.3.4.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

#### 56.3.4.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>This read only field returns the feature set number.<br>0000000000000001b - Standard feature set.<br>0000000000000011b - Standard feature set with MODEM/IrDA support. |

### 56.3.4.3 Parameter Register (PARAM)

#### 56.3.4.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 56.3.4.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### 56.3.4.3.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>RXFIFO | Receive FIFO Size<br>The number of words in the receive FIFO is 2^RXFIFO. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>The number of words in the transmit FIFO is 2^TXFIFO. |

## 56.3.4.4 LPUART Global Register (GLOBAL)

### 56.3.4.4.1 Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### 56.3.4.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RST | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.4.4.3 Fields

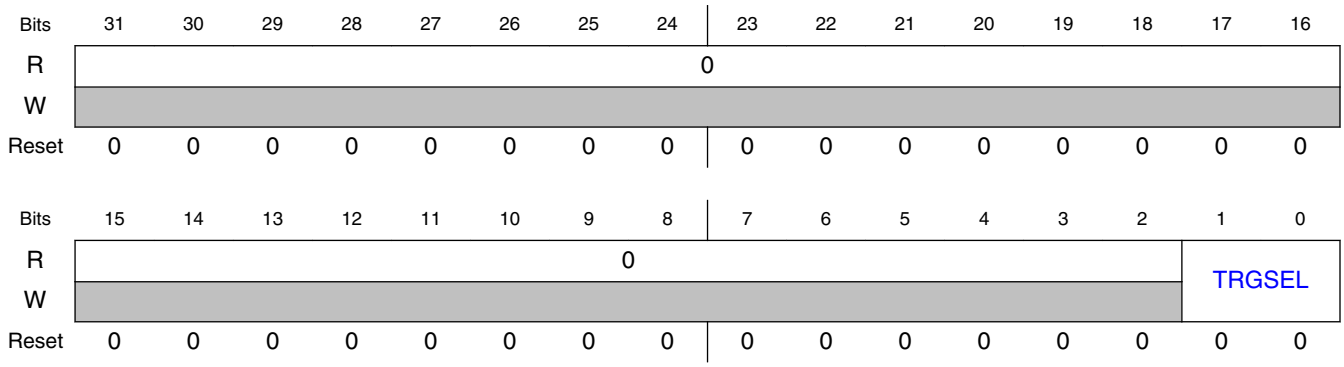
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Global Register. Remains set until cleared by software.<br>0b - Module is not reset.<br>1b - Module is reset. |
| 0<br>— | Reserved |

## 56.3.4.5 LPUART Pin Configuration Register (PINCFG)

### 56.3.4.5.1 Offset

| Register | Offset |
|---|---|
| PINCFG | Ch |

### 56.3.4.5.2 Diagram




### 56.3.4.5.3 Fields

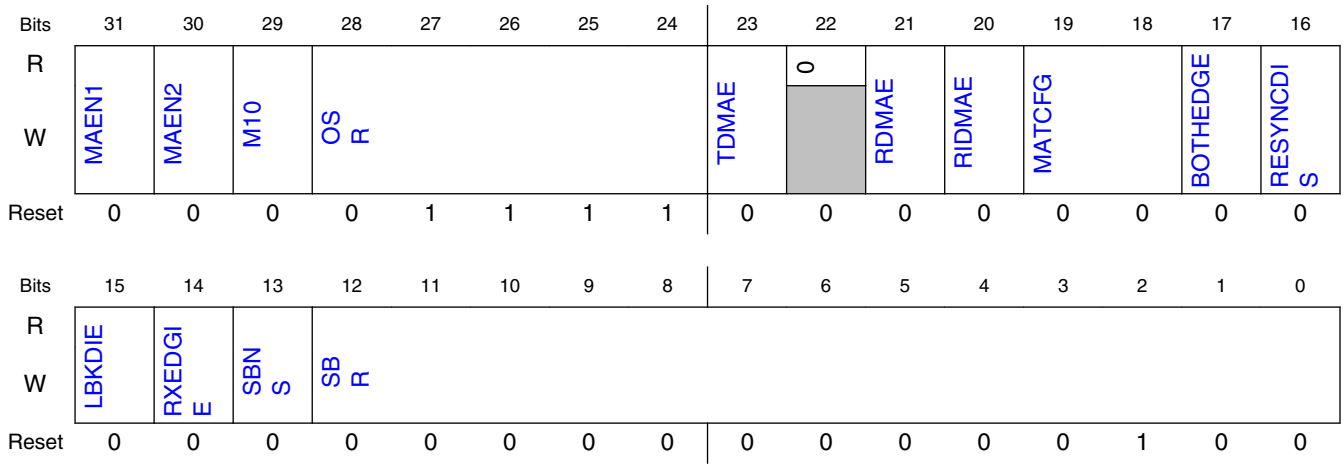
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>TRGSEL | Trigger Select<br>Configures the input trigger usage. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Input trigger is disabled.<br>01b - Input trigger is used instead of RXD pin input.<br>10b - Input trigger is used instead of CTS_B pin input.<br>11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger. |

## 56.3.4.6 LPUART Baud Rate Register (BAUD)

### 56.3.4.6.1 Offset

| Register | Offset |
|---|---|
| BAUD | 10h |

### 56.3.4.6.2 Diagram




### 56.3.4.6.3 Fields

| Field | Function |
|---|---|
| 31<br>MAEN1 | Match Address Mode Enable 1<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA1]. |
| 30<br>MAEN2 | Match Address Mode Enable 2<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MA2]. |
| 29<br>M10 | 10-bit Mode select<br>The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 7-bit to 9-bit data characters.<br>1b - Receiver and transmitter use 10-bit data characters. |
| 28-24<br>OSR | Oversampling Ratio<br>This field configures the oversampling ratio for the receiver. This field should only be changed when the transmitter and receiver are both disabled.<br>00000b - Writing 0 to this field will result in an oversampling ratio of 16<br>00001b - Reserved<br>00010b - Reserved<br>00011b - Oversampling ratio of 4, requires BOTHEDGE to be set.<br>00100b - Oversampling ratio of 5, requires BOTHEDGE to be set.<br>00101b - Oversampling ratio of 6, requires BOTHEDGE to be set.<br>00110b - Oversampling ratio of 7, requires BOTHEDGE to be set.<br>00111b - Oversampling ratio of 8.<br>01000b - Oversampling ratio of 9.<br>01001b - Oversampling ratio of 10.<br>01010b - Oversampling ratio of 11.<br>01011b - Oversampling ratio of 12.<br>01100b - Oversampling ratio of 13.<br>01101b - Oversampling ratio of 14.<br>01110b - Oversampling ratio of 15.<br>01111b - Oversampling ratio of 16.<br>10000b - Oversampling ratio of 17. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 10001b - Oversampling ratio of 18.<br>10010b - Oversampling ratio of 19.<br>10011b - Oversampling ratio of 20.<br>10100b - Oversampling ratio of 21.<br>10101b - Oversampling ratio of 22.<br>10110b - Oversampling ratio of 23.<br>10111b - Oversampling ratio of 24.<br>11000b - Oversampling ratio of 25.<br>11001b - Oversampling ratio of 26.<br>11010b - Oversampling ratio of 27.<br>11011b - Oversampling ratio of 28.<br>11100b - Oversampling ratio of 29.<br>11101b - Oversampling ratio of 30.<br>11110b - Oversampling ratio of 31.<br>11111b - Oversampling ratio of 32. |
| 23<br>TDMAE | Transmitter DMA Enable<br>TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 22<br>— | Reserved |
| 21<br>RDMAE | Receiver Full DMA Enable<br>RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 20<br>RIDMAE | Receiver Idle DMA Enable<br>RIDMAE configures the receiver idle flag, LPUART_STAT[IDLE], to generate a DMA request. When this bit is set, reading LPUART_DATA when either DATA[RXEMPT] or DATA[IDLINE] bit is set, will generate an End Of Packet response until the completion of the existing DMA transfer. During an End of Packet response, reading the LPUART_DATA register will return 0x0000_33FF and does not pull data from the FIFO.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| 19-18<br>MATCFG | Match Configuration<br>Configures the match addressing mode used. This field should only be changed when the transmitter and receiver are both disabled.<br>00b - Address Match Wakeup<br>01b - Idle Match Wakeup<br>10b - Match On and Match Off<br>11b - Enables RWU on Data Match and Match On/Off for transmitter CTS input |
| 17<br>BOTHEDGE | Both Edge Sampling<br>Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.<br>0b - Receiver samples input data using the rising edge of the baud rate clock.<br>1b - Receiver samples input data using the rising and falling edge of the baud rate clock. |
| 16<br>RESYNCDIS | Resynchronization Disable |

*Table continues on the next page...*

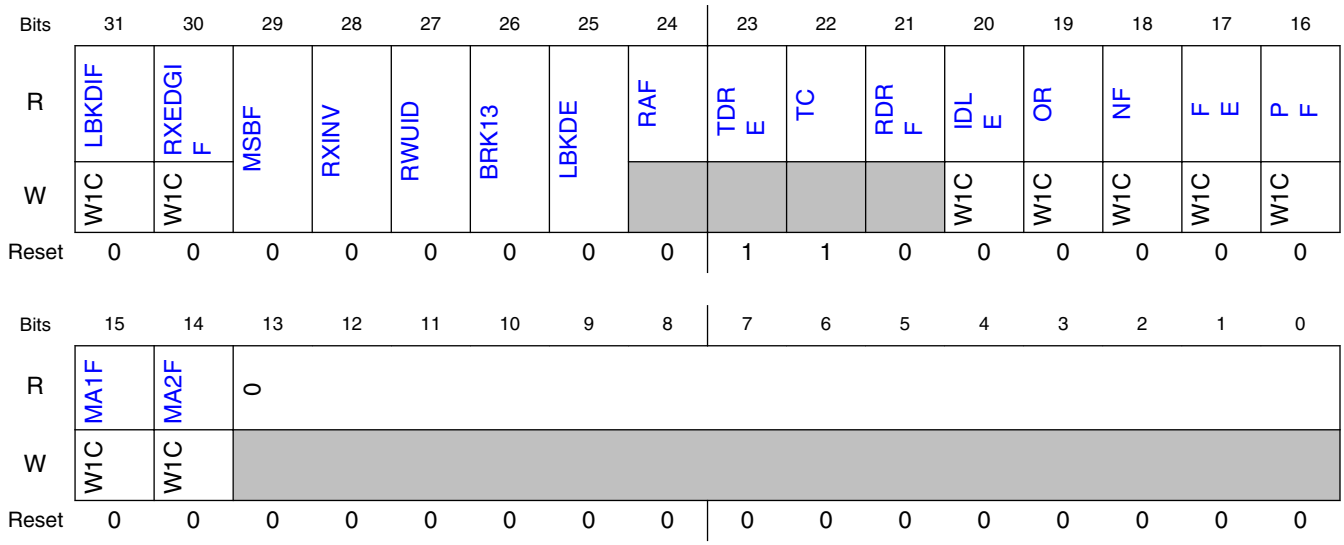
| Field | Function |
|---|---|
| | When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.<br>0b - Resynchronization during received data word is supported<br>1b - Resynchronization during received data word is disabled |
| 15<br>LBKDIE | LIN Break Detect Interrupt Enable<br>LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.<br>0b - Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling).<br>1b - Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1. |
| 14<br>RXEDGIE | RX Input Active Edge Interrupt Enable<br>Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.<br>0b - Hardware interrupts from LPUART_STAT[RXEDGIF] disabled.<br>1b - Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1. |
| 13<br>SBNS | Stop Bit Number Select<br>SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - One stop bit.<br>1b - Two stop bits. |
| 12-0<br>SBR | Baud Rate Modulo Divisor.<br>The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0). |

## 56.3.4.7 LPUART Status Register (STAT)

### 56.3.4.7.1 Offset

| Register | Offset |
|---|---|
| STAT | 14h |

## 56.3.4.7.2 Diagram




## 56.3.4.7.3 Fields

| Field | Function |
|---|---|
| 31<br>LBKDIF | LIN Break Detect Interrupt Flag<br>LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.<br>0b - No LIN break character has been detected.<br>1b - LIN break character has been detected. |
| 30<br>RXEDGIF | RXD Pin Active Edge Interrupt Flag<br>RXEDGIF is set whenever the receiver is enabled and an active edge, falling if RXINV = 0, rising if RXINV=1, on the RXD pin occurs. RXEDGIF is cleared by writing a 1 to it.<br>0b - No active edge on the receive pin has occurred.<br>1b - An active edge on the receive pin has occurred. |
| 29<br>MSBF | MSB First<br>Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0.<br>1b - MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE]. |
| 28<br>RXINV | Receive Data Inversion<br>Setting this bit reverses the polarity of the received data input. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.<br>0b - Receive data not inverted.<br>1b - Receive data inverted. |
| 27 | Receive Wake Up Idle Detect |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RWUID | For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.<br>0b - During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not set when an address does not match.<br>1b - During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does set when an address does not match. |
| 26<br>BRK13 | Break Character Generation Length<br>BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled. A break character can be sent by setting CTRL[SBK] or by writing the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Break character is transmitted with length of 9 to 13 bit times.<br>1b - Break character is transmitted with length of 12 to 15 bit times. |
| 25<br>LBKDE | LIN Break Detection Enable<br>LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.<br>0b - LIN break detect is disabled, normal break character can be detected.<br>1b - LIN break detect is enabled. LIN break character is detected at length of 11 bit times (if M = 0) or 12 (if M = 1) or 13 (M10 = 1). |
| 24<br>RAF | Receiver Active Flag<br>RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.<br>0b - LPUART receiver idle waiting for a start bit.<br>1b - LPUART receiver active (RXD input not idle). |
| 23<br>TDRE | Transmit Data Register Empty Flag<br>When the transmit FIFO is enabled, TDRE will set when the number of datawords in the transmit FIFO (LPUART_DATA) is equal to or less than the number indicated by LPUART_WATER[TXWATER]). To clear TDRE, write to the LPUART data register (LPUART_DATA) until the number of words in the transmit FIFO is greater than the number indicated by LPUART_WATER[TXWATER]. When the transmit FIFO is disabled,TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA).<br>TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.<br>0b - Transmit data buffer full.<br>1b - Transmit data buffer empty. |
| 22<br>TC | Transmission Complete Flag<br>TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK].<br>0b - Transmitter active (sending data, a preamble, or a break).<br>1b - Transmitter idle (transmission activity complete). |
| 21<br>RDRF | Receive Data Register Full Flag<br>When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by LPUART_WATER[RXWATER]. To clear RDRF, read LPUART_DATA until the number of datawords in the receive data buffer is equal to or less than the number indicated by LPUART_WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.<br>0b - Receive data buffer empty.<br>1b - Receive data buffer full. |
| 20<br>IDLE | Idle Line Flag<br>IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period.<br>0b - No idle line detected.<br>1b - Idle line was detected. |
| 19<br>OR | Receiver Overrun Flag<br>OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.<br>While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.<br>0b - No overrun.<br>1b - Receive overrun (new LPUART data lost). |
| 18<br>NF | Noise Flag<br>The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.<br>0b - No noise detected.<br>1b - Noise detected in the received character in LPUART_DATA. |
| 17<br>FE | Framing Error Flag<br>FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear FE, write logic one to the FE.<br>0b - No framing error detected. This does not guarantee the framing is correct.<br>1b - Framing error. |
| 16<br>PF | Parity Error Flag<br>PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.<br>0b - No parity error.<br>1b - Parity error. |
| 15<br>MA1F | Match 1 Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.<br>0b - Received data is not equal to MA1<br>1b - Received data is equal to MA1 |
| 14<br>MA2F | Match 2 Flag<br>MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.<br>0b - Received data is not equal to MA2<br>1b - Received data is equal to MA2 |
| 13-0<br>— | Reserved |

## 56.3.4.8 LPUART Control Register (CTRL)

### 56.3.4.8.1 Offset

| Register | Offset |
|---|---|
| CTRL | 18h |

### 56.3.4.8.2 Function

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

### 56.3.4.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MA1IE | MA2IE | 0 | | M7 | IDLECFG | | | LOOPS | DOZEEN | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 56.3.4.8.4 Fields

| Field | Function |
|---|---|
| 31<br>R8T9 | Receive Bit 8 / Transmit Bit 9<br>R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA.<br>T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R8 is a read-only bit and T9 is a write-only bit, the value read is different from the value written. |
| 30<br>R9T8 | Receive Bit 9 / Transmit Bit 8<br>R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA<br>T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.<br>**NOTE:** R9 is a read-only bit and T8 is a write-only bit, the value read is different from the value written. |
| 29<br>TXDIR | TXD Pin Direction in Single-Wire Mode<br>When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the TXD pin.<br>0b - TXD pin is an input in single-wire mode.<br>1b - TXD pin is an output in single-wire mode. |
| 28<br>TXINV | Transmit Data Inversion<br>Setting this bit reverses the polarity of the transmitted data output.<br>**NOTE:** Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.<br>0b - Transmit data not inverted.<br>1b - Transmit data inverted. |
| 27<br>ORIE | Overrun Interrupt Enable<br>This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0b - OR interrupts disabled; use polling.<br>1b - Hardware interrupt requested when OR is set. |
| 26<br>NEIE | Noise Error Interrupt Enable<br>This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0b - NF interrupts disabled; use polling.<br>1b - Hardware interrupt requested when NF is set. |
| 25<br>FEIE | Framing Error Interrupt Enable<br>This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0b - FE interrupts disabled; use polling.<br>1b - Hardware interrupt requested when FE is set. |
| 24<br>PEIE | Parity Error Interrupt Enable<br>This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0b - PF interrupts disabled; use polling).<br>1b - Hardware interrupt requested when PF is set. |
| 23 | Transmit Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIE | Enables STAT[TDRE] to generate interrupt requests.<br>0b - Hardware interrupts from TDRE disabled; use polling.<br>1b - Hardware interrupt requested when TDRE flag is 1. |
| 22<br>TCIE | Transmission Complete Interrupt Enable for<br>TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0b - Hardware interrupts from TC disabled; use polling.<br>1b - Hardware interrupt requested when TC flag is 1. |
| 21<br>RIE | Receiver Interrupt Enable<br>Enables STAT[RDRF] to generate interrupt requests.<br>0b - Hardware interrupts from RDRF disabled; use polling.<br>1b - Hardware interrupt requested when RDRF flag is 1. |
| 20<br>ILIE | Idle Line Interrupt Enable<br>ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.<br>0b - Hardware interrupts from IDLE disabled; use polling.<br>1b - Hardware interrupt requested when IDLE flag is 1. |
| 19<br>TE | Transmitter Enable<br>Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the TXD pin is tristated.<br>A single idle character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] set.<br>0b - Transmitter disabled.<br>1b - Transmitter enabled. |
| 18<br>RE | Receiver Enable<br>Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).<br>0b - Receiver disabled.<br>1b - Receiver enabled. |
| 17<br>RWU | Receiver Wakeup Control<br>This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.<br>**NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.<br>0b - Normal receiver operation.<br>1b - LPUART receiver in standby waiting for wakeup condition. |
| 16<br>SBK | Send Break<br>Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if LPUART_STAT[BRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK.<br>A single break character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear.<br>0b - Normal transmitter operation.<br>1b - Queue break character(s) to be sent. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>MA1IE | Match 1 Interrupt Enable<br>0b - MA1F interrupt disabled<br>1b - MA1F interrupt enabled |
| 14<br>MA2IE | Match 2 Interrupt Enable<br>0b - MA2F interrupt disabled<br>1b - MA2F interrupt enabled |
| 13-12<br>— | Reserved |
| 11<br>M7 | 7-Bit Mode Select<br>This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - Receiver and transmitter use 8-bit to 10-bit data characters.<br>1b - Receiver and transmitter use 7-bit data characters. |
| 10-8<br>IDLECFG | Idle Configuration<br>Configures the number of idle characters that must be received before the IDLE flag is set.<br>000b - 1 idle character<br>001b - 2 idle characters<br>010b - 4 idle characters<br>011b - 8 idle characters<br>100b - 16 idle characters<br>101b - 32 idle characters<br>110b - 64 idle characters<br>111b - 128 idle characters |
| 7<br>LOOPS | Loop Mode Select<br>When LOOPS is set, the RXD pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.<br>0b - Normal operation - RXD and TXD use separate pins.<br>1b - Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit). |
| 6<br>DOZEEN | Doze Enable<br>0b - LPUART is enabled in Doze mode.<br>1b - LPUART is disabled in Doze mode. |
| 5<br>RSRC | Receiver Source Select<br>This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input.<br>0b - Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the RXD pin.<br>1b - Single-wire LPUART mode where the TXD pin is connected to the transmitter output and receiver input. |
| 4<br>M | 9-Bit or 8-Bit Mode Select<br>0b - Receiver and transmitter use 8-bit data characters.<br>1b - Receiver and transmitter use 9-bit data characters. |
| 3<br>WAKE | Receiver Wakeup Method Select<br>Determines which condition wakes the LPUART when RWU=1:<br>• Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character, or<br>• An idle condition on the receive pin input signal.<br><br>0b - Configures RWU for idle-line wakeup.<br>1b - Configures RWU with address-mark wakeup. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>ILT | Idle Line Type Select<br>Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>**NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>0b - Idle character bit count starts after start bit.<br>1b - Idle character bit count starts after stop bit. |
| 1<br>PE | Parity Enable<br>Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.<br>0b - No hardware parity generation or checking.<br>1b - Parity enabled. |
| 0<br>PT | Parity Type<br>Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0b - Even parity.<br>1b - Odd parity. |

## 56.3.4.9 LPUART Data Register (DATA)

### 56.3.4.9.1 Offset

| Register | Offset |
|---|---|
| DATA | 1Ch |

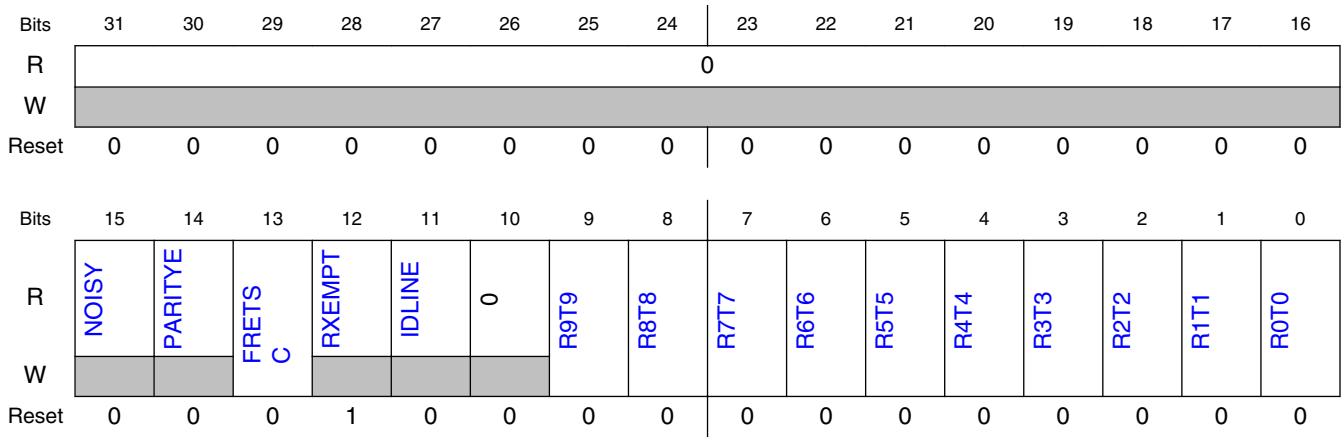
### 56.3.4.9.2 Function

**NOTE**

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

### 56.3.4.9.3 Diagram




### 56.3.4.9.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>NOISY | NOISY<br>The current received dataword contained in DATA[R9:R0] was received with noise.<br>0b - The dataword was received without noise.<br>1b - The data was received with noise. |
| 14<br>PARITYE | PARITYE<br>The current received dataword contained in DATA[R9:R0] was received with a parity error.<br>0b - The dataword was received without a parity error.<br>1b - The dataword was received with a parity error. |
| 13<br>FRETSC | Frame Error / Transmit Special Character<br>For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero.<br>0b - The dataword was received without a frame error on read, or transmit a normal character on write.<br>1b - The dataword was received with a frame error, or transmit an idle or break character on transmit. |
| 12<br>RXEMPT | Receive Buffer Empty<br>Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer contains valid data.<br>1b - Receive buffer is empty, data returned on read is not valid. |
| 11<br>IDLINE | Idle Line<br>Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.<br>0b - Receiver was not idle before receiving this character.<br>1b - Receiver was idle before receiving this character. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>— | Reserved |
| 9<br>R9T9 | R9T9<br>Read receive data buffer 9 or write transmit data buffer 9. |
| 8<br>R8T8 | R8T8<br>Read receive data buffer 8 or write transmit data buffer 8. |
| 7<br>R7T7 | R7T7<br>Read receive data buffer 7 or write transmit data buffer 7. |
| 6<br>R6T6 | R6T6<br>Read receive data buffer 6 or write transmit data buffer 6. |
| 5<br>R5T5 | R5T5<br>Read receive data buffer 5 or write transmit data buffer 5. |
| 4<br>R4T4 | R4T4<br>Read receive data buffer 4 or write transmit data buffer 4. |
| 3<br>R3T3 | R3T3<br>Read receive data buffer 3 or write transmit data buffer 3. |
| 2<br>R2T2 | R2T2<br>Read receive data buffer 2 or write transmit data buffer 2. |
| 1<br>R1T1 | R1T1<br>Read receive data buffer 1 or write transmit data buffer 1. |
| 0<br>R0T0 | R0T0<br>Read receive data buffer 0 or write transmit data buffer 0. |

## 56.3.4.10 LPUART Match Address Register (MATCH)

### 56.3.4.10.1 Offset

| Register | Offset |
|---|---|
| MATCH | 20h |

### 56.3.4.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MA2 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MA1 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.4.10.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>MA2 | Match Address 2<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |
| 15-10<br>— | Reserved |
| 9-0<br>MA1 | Match Address 1<br>The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear. |

## 56.3.4.11 LPUART Modem IrDA Register (MODIR)

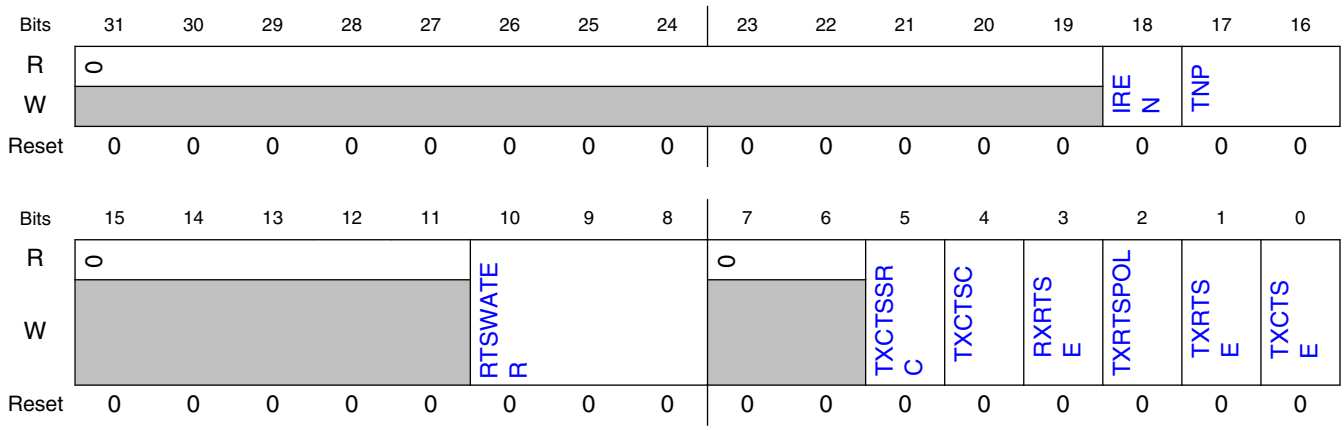
### 56.3.4.11.1 Offset

| Register | Offset |
|---|---|
| MODIR | 24h |

### 56.3.4.11.2 Function

The MODEM register controls options for setting the modem configuration.

## 56.3.4.11.3 Diagram




## 56.3.4.11.4 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>IREN | Infrared enable<br>Enables/disables the infrared modulation/demodulation. This bit should only be changed when the transmitter and receiver are both disabled.<br>0b - IR disabled.<br>1b - IR enabled. |
| 17-16<br>TNP | Transmitter narrow pulse<br>Configures whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse when IR is enabled. This bit should only be changed when the transmitter and receiver are both disabled.<br>The IR pulse width should be configured to less than half of the oversampling ratio. Common pulse widths are 3/16, 1/16, 1/32 or 1/4 of the bit length. These can be configured by selecting the appropriate oversample ratio and pulse width.<br>00b - 1/OSR.<br>01b - 2/OSR.<br>10b - 3/OSR.<br>11b - 4/OSR. |
| 15-11<br>— | Reserved |
| 10-8<br>RTSWATER | Receive RTS Configuration<br>Configures the assertion and negation of the RX RTS_B output. The RX RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to RTSWATER. If RTSWATER is configured to zero, RTS_B negates when the receive FIFO is full. For the purpose of RX RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both RX RTS_B and address/data matching is enabled, then RTS_B could assert at the end of a character if there is not a match. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This field should only be changed when the receiver is disabled. |
| 7-6<br>— | Reserved |
| 5<br>TXCTSSRC | Transmit CTS Source<br>Configures the source of the CTS input.<br>0b - CTS input is the CTS_B pin.<br>1b - CTS input is the inverted Receiver Match result. |
| 4<br>TXCTSC | Transmit CTS Configuration<br>Configures if the CTS state is checked at the start of each character or only when the transmitter is idle.<br>0b - CTS input is sampled at the start of each character.<br>1b - CTS input is sampled when the transmitter is idle. |
| 3<br>RXRTSE | Receiver request-to-send enable<br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. This bit should only be changed when the receiver is disabled.<br>**NOTE:** Do not set both RXRTSE and TXRTSE.<br>0b - The receiver has no effect on RTS.<br>1b - RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full. |
| 2<br>TXRTSPOL | Transmitter request-to-send polarity<br>Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set. This bit should only be changed when the transmitter is disabled.<br>0b - Transmitter RTS is active low.<br>1b - Transmitter RTS is active high. |
| 1<br>TXRTSE | Transmitter request-to-send enable<br>Controls RTS before and after a transmission. This bit should only be changed when the transmitter is disabled.<br>0b - The transmitter has no effect on RTS.<br>1b - When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. |
| 0<br>TXCTSE | Transmitter clear-to-send enable<br>TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE.<br>0b - CTS has no effect on the transmitter.<br>1b - Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission. |

## 56.3.4.12 LPUART FIFO Register (FIFO)

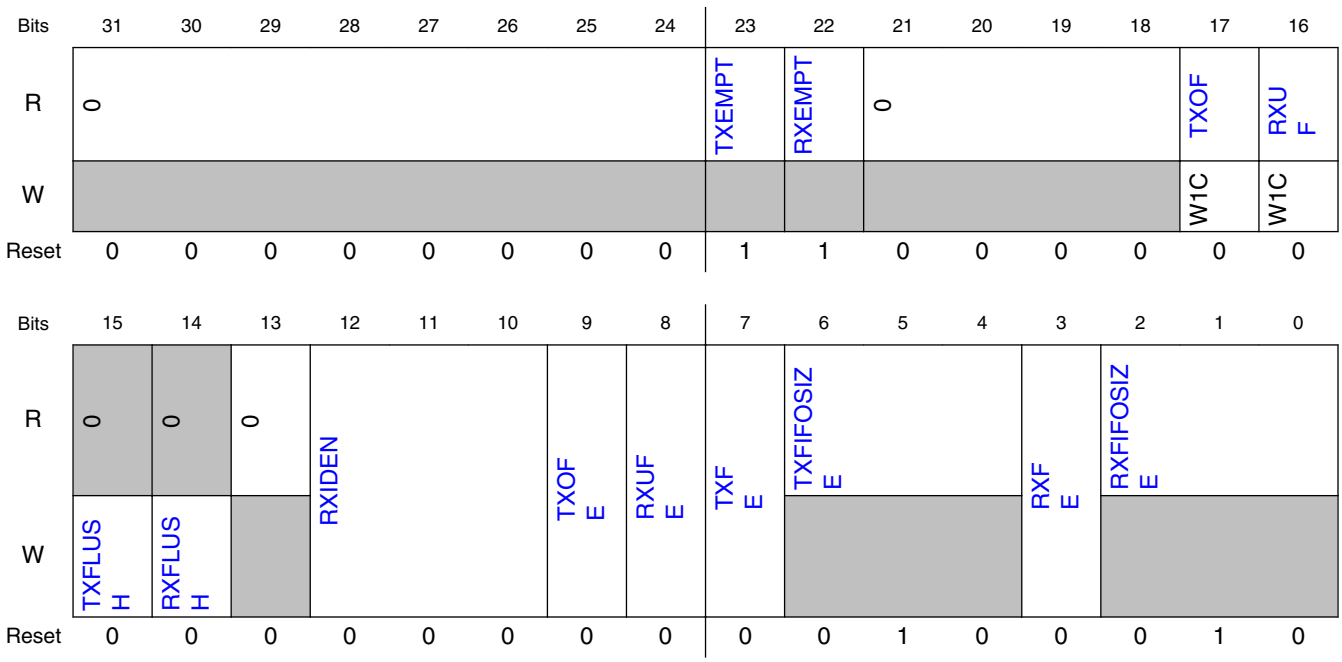
### 56.3.4.12.1 Offset

| Register | Offset |
|---|---|
| FIFO | 28h |

### 56.3.4.12.2 Function

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when CTRL[RE] and CTRL[TE] are cleared/not set and when the data buffer/FIFO is empty.

### 56.3.4.12.3 Diagram




### 56.3.4.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>TXEMPT | Transmit Buffer/FIFO Empty<br>Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register.<br>0b - Transmit buffer is not empty. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Transmit buffer is empty. |
| 22<br>RXEMPT | Receive Buffer/FIFO Empty<br>Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register.<br>0b - Receive buffer is not empty.<br>1b - Receive buffer is empty. |
| 21-18<br>— | Reserved |
| 17<br>TXOF | Transmitter Buffer Overflow Flag<br>Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1.<br>0b - No transmit buffer overflow has occurred since the last time the flag was cleared.<br>1b - At least one transmit buffer overflow has occurred since the last time the flag was cleared. |
| 16<br>RXUF | Receiver Buffer Underflow Flag<br>Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1.<br>0b - No receive buffer underflow has occurred since the last time the flag was cleared.<br>1b - At least one receive buffer underflow has occurred since the last time the flag was cleared. |
| 15<br>TXFLUSH | Transmit FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register.<br>0b - No flush operation occurs.<br>1b - All data in the transmit FIFO/Buffer is cleared out. |
| 14<br>RXFLUSH | Receive FIFO/Buffer Flush<br>Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register.<br>0b - No flush operation occurs.<br>1b - All data in the receive FIFO/buffer is cleared out. |
| 13<br>— | Reserved |
| 12-10<br>RXIDEN | Receiver Idle Empty Enable<br>When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty.<br>000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle.<br>001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character.<br>010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters.<br>011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters.<br>100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters.<br>101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters.<br>110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters.<br>111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters. |
| 9<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br>When this field is set, the TXOF flag generates an interrupt to the host.<br>0b - TXOF flag does not generate an interrupt to the host.<br>1b - TXOF flag generates an interrupt to the host. |
| 8 | Receive FIFO Underflow Interrupt Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXUFE | When this field is set, the RXUF flag generates an interrupt to the host.<br>0b - RXUF flag does not generate an interrupt to the host.<br>1b - RXUF flag generates an interrupt to the host. |
| 7<br>TXFE | Transmit FIFO Enable<br>When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).<br>1b - Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE. |
| 6-4<br>TXFIFOSIZE | Transmit FIFO. Buffer Depth<br>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.<br>000b - Transmit FIFO/Buffer depth = 1 dataword.<br>001b - Transmit FIFO/Buffer depth = 4 datawords.<br>010b - Transmit FIFO/Buffer depth = 8 datawords.<br>011b - Transmit FIFO/Buffer depth = 16 datawords.<br>100b - Transmit FIFO/Buffer depth = 32 datawords.<br>101b - Transmit FIFO/Buffer depth = 64 datawords.<br>110b - Transmit FIFO/Buffer depth = 128 datawords.<br>111b - Transmit FIFO/Buffer depth = 256 datawords |
| 3<br>RXFE | Receive FIFO Enable<br>When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field.<br>0b - Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)<br>1b - Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE. |
| 2-0<br>RXFIFOSIZE | Receive FIFO. Buffer Depth<br>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.<br>000b - Receive FIFO/Buffer depth = 1 dataword.<br>001b - Receive FIFO/Buffer depth = 4 datawords.<br>010b - Receive FIFO/Buffer depth = 8 datawords.<br>011b - Receive FIFO/Buffer depth = 16 datawords.<br>100b - Receive FIFO/Buffer depth = 32 datawords.<br>101b - Receive FIFO/Buffer depth = 64 datawords.<br>110b - Receive FIFO/Buffer depth = 128 datawords.<br>111b - Receive FIFO/Buffer depth = 256 datawords. |

## 56.3.4.13 LPUART Watermark Register (WATER)

### 56.3.4.13.1 Offset

| Register | Offset |
|---|---|
| WATER | 2Ch |

### 56.3.4.13.2 Function

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when CTRL[TE] is not set.

### 56.3.4.13.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RXCOUNT | | | | 0 | | | | | RXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TXCOUNT | | | | 0 | | | | | TXWATER | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 56.3.4.13.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>RXCOUNT | Receive Counter<br>The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer. |
| 23-19<br>— | Reserved |
| 18-16<br>RXWATER | Receive Watermark<br>When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE] and must be greater than 0. |
| 15-12<br>— | Reserved |
| 11-8<br>TXCOUNT | Transmit Counter<br>The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer. |
| 7-3 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 2-0<br>TXWATER | Transmit Watermark<br>When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE]. |

## 56.4 Functional description

The LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following describes each of the blocks of the LPUART.

### 56.4.1 Baud rate generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to BAUD[SBR] determines the baud clock divisor for the asynchronous LPUART baud clock. The baud rate clock drives the receiver, while the transmitter is driven by a bit clock which is generated from baud rate clock divided by the over sampling ratio. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.

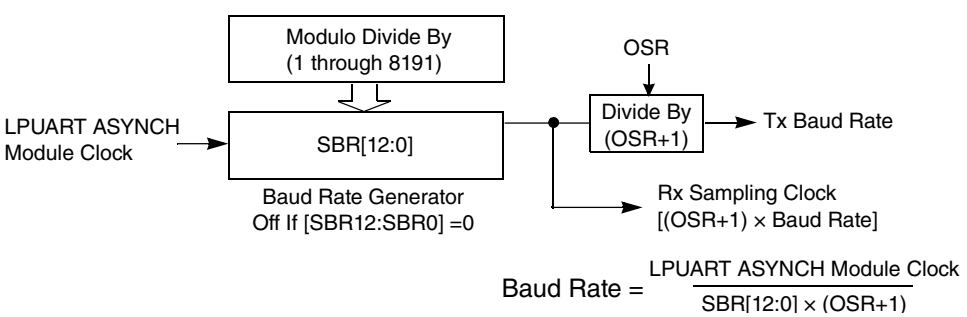



**Figure 56-3. LPUART baud rate generation**

Baud rate generation is subject to two sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can cause phase shift.

The baud rate generation is a free-running counter that continues whenever the transmitter or receiver is enabled. The transmitter bit clock continues whenever the transmitter is enabled, each transmitted character will align to the next edge of the transmit bit clock.

## 56.4.2 Transmitter functional description

This section describes the overall block diagram for the LPUART transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (TXD) idle state defaults to logic high, CTRL[TXINV] is cleared following reset. The transmitter output is inverted by setting CTRL[TXINV]. The transmitter is enabled by setting the CTRL[TE] bit. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the DATA register.

The central element of the LPUART transmitter is the transmit shift register that is 9-bit to 13 bits long depending on the setting in the CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SBNS] control bits. For the remainder of this section, assume CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SBNS] are cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock, and the transmit data register empty (STAT[TDRE]) status flag is set to indicate another character may be written to the transmit data buffer at the DATA register.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TXD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TXD high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character or break character), although the transmitter will not start transmitting another character.

### 56.4.2.1 Send break and queued idle

The CTRL[SBK] bit sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 9-bit to 12-bit times including the start and stop bits. A longer break of 13-bit times can be enabled by setting STAT[BRK13]. Normally, a program would wait for STAT[TDRE] to become set

to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to the CTRL[SBK] bit. This action queues a break character to be sent as soon as the shifter is available. If CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized to the baud rate clock, an additional break character is queued. When the LPUART is the receiving device, a break character is received as 0s in all data bits and a framing error (STAT[FE] = 1) is detected.

A break character can also be transmitted by writing to the DATA register with bit 13 set and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows the DMA to transmit a break character.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the CTRL[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while CTRL[TE] is cleared, the LPUART transmitter never actually releases control of the TXD pin.

An idle character can also be transmitted by writing to the DATA register with bit 13 set and the data bits also set. This supports transmitting the idle character as part of the normal data stream and also allows the DMA to transmit a idle character.

The length of the break character is affected by the STAT[BRK13], CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SNBS] bits as shown below.

**Table 56-2. Break character length**

| BRK13 | M | M10 | M7 | SBNS | Break character length |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 10 bit times |
| 0 | 0 | 0 | 0 | 1 | 11 bit times |
| 0 | 0 | 0 | 1 | 0 | 9 bit times |
| 0 | 0 | 0 | 1 | 1 | 10 bit times |
| 0 | 1 | 0 | X | 0 | 11 bit times |
| 0 | 1 | 0 | X | 1 | 12 bit times |
| 0 | X | 1 | X | 0 | 12 bit times |
| 0 | X | 1 | X | 1 | 13 bit times |
| 1 | 0 | 0 | 0 | 0 | 13 bit times |
| 1 | 0 | 0 | 0 | 1 | 13 bit times |
| 1 | 0 | 0 | 1 | 0 | 12 bit times |
| 1 | 0 | 0 | 1 | 1 | 12 bit times |
| 1 | 1 | 0 | X | 0 | 14 bit times |
| 1 | 1 | 0 | X | 1 | 14 bit times |

*Table continues on the next page...*

**Table 56-2. Break character length (continued)**

| BRK13 | M | M10 | M7 | SBNS | Break character length |
|---|---|---|---|---|---|
| 1 | X | 1 | X | 0 | 15 bit times |
| 1 | X | 1 | X | 1 | 15 bit times |

### 56.4.2.2 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS_B. If the clear-to-send operation is enabled, the character is transmitted when CTS_B is asserted. If CTS_B is deasserted in the middle of a transmission with characters remaining in the transmitter data buffer, the character in the shift register is sent and TXD remains in the mark state until CTS_B is reasserted. The CTS_B pin must assert for longer than one bit period to guarantee a new transmission is started when the transmitter is idle with data to send.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS_B.

The transmitter's CTS_B signal can also be enabled even if the same LPUART receiver's RTS_B signal is disabled.

### 56.4.2.3 Transceiver driver enable

The transmitter can use RTS_B as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS_B for details. If the request-to-send operation is enabled, when a character is placed into an empty transmitter data buffer, RTS_B asserts one bit time before the start bit is transmitted. RTS_B remains asserted for the whole time that the transmitter data buffer has any characters. RTS_B deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. Transmitting a break character also asserts RTS_B, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

The transmitter's RTS_B signal asserts only when the transmitter is enabled. However, the transmitter's RTS_B signal is unaffected by its CTS_B signal. RTS_B will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

### 56.4.2.4 Transceiver driver enable using RTS_B

RS-485 is a multiple drop communication protocol in which the LPUART transceiver's driver is 3-stated unless the LPUART is driving. The RTS_B signal can be used by the transmitter to enable the driver of a transceiver. The polarity of RTS_B can be matched to the polarity of the transceiver's driver enable signal.

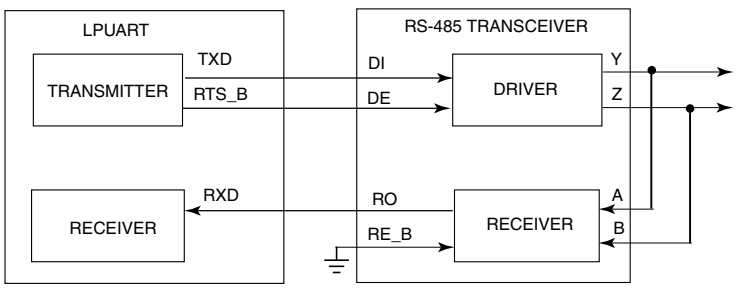



**Figure 56-4. Transceiver driver enable using RTS_B**

In the figure, the receiver enable signal is asserted. Another option for this connection is to connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled while driving. A pullup can pull RXD to a non-floating value during this time. This option can be refined further by operating the LPUART in single wire mode, freeing the RXD pin for other uses.

## 56.4.3 Receiver functional description

In this section, the receiver block diagram is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, different variations of the receiver wakeup function are explained.

The receiver input is inverted by setting STAT[RXINV]. The receiver is enabled by setting the CTRL[RE] bit. Character frames consist of a start bit of logic 0, seven to ten data bits (msb or lsb first), and one or two stop bits of logic 1. For information about 7-bit, 9-bit or 10-bit data mode, refer to Data Modes. For the remainder of this discussion, assume the LPUART is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (STAT[RDRF]) status flag is set. If [RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the LPUART receiver is double-buffered, the program has one full character time after [RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the receive data register by reading the DATA register. Refer to Interrupts and status flags for details about flag clearing.

### 56.4.3.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between 4× and 32× of the baud rate clock for sampling. The receiver starts by taking logic level samples at the oversampling rate times the baud rate to search for a falling edge on the RXD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts the sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (STAT[NF]) is set when the received character is transferred to the receive data buffer.

When the LPUART receiver is configured to sample on both edges of the baud rate clock, the number of segments in each received bit is effectively doubled (from 1 to OSR×2). The start and data bits are then sampled at OSR, OSR+1 and OSR+2. Sampling on both edges of the clock must be enabled for oversampling rates of 4× to 7× and is optional for higher oversampling rates.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times (unless resynchronization has been disabled). This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

### 56.4.3.2 Receiver wakeup operation

Receiver wakeup and receiver address matching is a hardware mechanism that allows an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wakeup, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control bit (CTRL[RWU]). When CTRL[RWU] and STAT[RWUID] bit are set, the status flags associated with the receiver, with the exception of the idle bit, STAT[IDLE], are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force CTRL[RWU] to 0 so all receivers wake up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver will ignore all characters that do not meet the address match requirements.

**Table 56-3. Receiver Wakeup Options**

| RWU | MA1 \| MA2 | MATCFG | WAKE:RWUID | Receiver Wakeup |
|---|---|---|---|---|
| 0 | 0 | X | X | Normal operation |
| 1 | 0 | 00 | 00 | Receiver wakeup on idle line, IDLE flag not set |
| 1 | 0 | 00 | 01 | Receiver wakeup on idle line, IDLE flag set |
| 1 | 0 | 00 | 10 | Receiver wakeup on address mark |
| 1 | 1 | 11 | X0 | Receiver wakeup on data match |
| 0 | 1 | 00 | X0 | Address mark address match, IDLE flag not set for discarded characters |
| 0 | 1 | 00 | X1 | Address mark address match, IDLE flag set for discarded characters |
| 0 | 1 | 01 | X0 | Idle line address match |
| 0 | 1 | 10 | X0 | Address match on and address match off, IDLE flag not set for discarded characters |
| 0 | 1 | 10 | X1 | Address match on and address match off, IDLE flag set for discarded characters |

#### 56.4.3.2.1 Idle-line wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The CTRL[M], CTRL[M7] and BAUD[M10] control bit selects 7-bit to 10-bit data mode and the BAUD[SBNS] bit selects 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 9 to 13 bit times because of the start and stop bits.

When CTRL[RWU] is one and STAT[RWUID] is 0, the idle condition that wakes up the receiver does not set the STAT[IDLE] flag. The receiver wakes up and waits for the first data character of the next message that sets the STAT[RDRF] flag and generates an interrupt if enabled. When STAT[RWUID] is 1, any idle condition sets the STAT[IDLE] flag and generates an interrupt if enabled, regardless of whether CTRL[RWU] is 0 or 1.

The idle-line type (CTRL[ILT]) control bit selects one of two ways to detect an idle line. When CTRL[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When CTRL[ILT] is set, the idle bit counter does not start until after the stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### 56.4.3.2.2 Address-mark wakeup

When CTRL[WAKE] is set, the receiver is configured for address-mark wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a logic 1 in the most significant bit of the received character. When parity is enabled, the second most significant bit is used for address-mark wakeup.

Address-mark wakeup allows messages to contain idle characters, but requires one bit be reserved for use in address frames. The logic 1 in the most significant bit (or second most significant bit when parity is enabled) of an address frame clears the CTRL[RWU] bit and sets the STAT[RDRF] flag. In this case, the character with the address-mark bit is received even though the receiver was sleeping during most of this character time.

#### 56.4.3.2.3 Data match wakeup

When CTRL[RWU] is set and BAUD[MATCFG] equals 11, the receiver is configured for data match wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a character that matches MATCH[MA1] field when BAUD[MAEN1] is set, or that matches MATCH[MA2] when BAUD[MAEN2] is set.

#### 56.4.3.2.4 Address Match operation

Address match operation is enabled when the BAUD[MAEN1] or BAUD[MAEN2] bit is set and BAUD[MATCFG] is equal to 00. In this function, a character received by the RXD pin with a logic 1 in the most significant bit (or second most significant bit when parity is enabled) is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and STAT[RDRF] is set, if the comparison matches. All subsequent characters received with a logic 0 in the most significant bit (or second most significant bit when parity is enabled) are considered to be data associated with the address and are transferred to the receive data buffer. If no marked address match occurs then no transfer is made to the receive data buffer, and all following characters with logic zero in the most significant bit (or second most significant bit when parity is enabled) are also discarded. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2] fields.

- If only one of BAUD[MAEN1] and BAUD[MAEN2] is asserted, a marked address is compared only with the associated match register field and data is transferred to the receive data buffer only on a match.
- If BAUD[MAEN1] and BAUD[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either of the MATCH[MA1] or MATCH[MA2] fields.

#### 56.4.3.2.5 Idle Match operation

Idle match operation is enabled when the BAUD[MAEN1] or BAUD[MAEN2] bit is set and BAUD[MATCFG] is equal to 01. In this function, the first character received by the RXD pin after an idle line condition is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and STAT[RDRF] is set, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive data buffer until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive data buffer, and all following frames until the next idle condition are also discarded. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Idle match operation functions in the same way for both MATCH[MA1] or MATCH[MA2] fields.

- If only one of BAUD[MAEN1] and BAUD[MAEN2] is asserted, the first character after an idle line is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If BAUD[MAEN1] and BAUD[MAEN2] are asserted, the first character after an idle line is compared with both MATCH[MA1] or MATCH[MA2] fields and data is transferred only on a match with either of the fields.

#### 56.4.3.2.6 Match On Match Off operation

Match on, match off operation is enabled when both BAUD[MAEN1] and BAUD[MAEN2] are set and BAUD[MATCFG] is equal to 10. In this function, a character received by the RXD pin that matches MATCH[MA1] is received and transferred to the receive buffer, and STAT[RDRF] is set. All subsequent characters are considered to be data and are also transferred to the receive data buffer, until a character is received that matches MATCH[MA2] field. The character that matches MATCH[MA2] and all following characters are discarded; this continues until another character that matches MATCH[MA1] is received. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

**NOTE**

Match on, match off operation requires both BAUD[MAEN1] and BAUD[MAEN2] to be asserted.

### 56.4.3.3 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert RTS_B.

- RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS_B for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts RTS_B if the number of characters in the receiver data register is full or a start bit is detected that will cause the receiver data register to be full.
- The receiver asserts RTS_B when the number of characters in the receiver data register is not full and has not detected a start bit that will cause the receiver data register to be full. It is not affected if STAT[RDRF] is asserted.

- Even if RTS_B is deasserted, the receiver continues to receive characters until the receiver data buffer is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS_B remains deasserted.

#### 56.4.3.4 Infrared decoder

The infrared decoder converts the received character from the IrDA format to the NRZ format used by the receiver. It also has a OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received.

##### 56.4.3.4.1 Start bit detection

When STAT[RXINV] is cleared, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

##### 56.4.3.4.2 Noise filtering

Any further rising edges detected during the first half of the infrared decoder counter are ignored by the decoder. Any pulses less than one oversampling baud clock can be undetected by it regardless of whether it is seen in the first or second half of the count.

##### 56.4.3.4.3 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as a 0, which is sent to the receiver. The decoder counter is also reset.

##### 56.4.3.4.4 High-bit detection

At OSR oversampling baud rate clocks after the previous rising edge, if a rising edge is not seen, then the decoder sends a 1 to the receiver.

If the next bit is a 0, which arrives late, then a low-bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to a 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, then the delay of a 0 is not recorded as noise.

## 56.4.4 Additional LPUART functions

The following sections describe additional LPUART functions.

### 56.4.4.1 Data Modes

The LPUART transmitter and receiver can be configured to operate in 7-bit data mode by setting CTRL[M7], 9-bit data mode by setting the CTRL[M] or 10-bit data mode by setting BAUD[M10]. In 9-bit mode, there is a ninth data bit and in 10-bit mode, there is a tenth data bit. For the transmit data buffer, these bits are stored in CTRL[T8] and CTRL[T9]. For the receiver, these bits are held in CTRL[R8] and CTRL[R9]. They are also accessible via 16-bit or 32-bit accesses to the DATA register.

For coherent 8-bit writes to the transmit data buffer, write to CTRL[T8] and CTRL[T9] before writing to DATA[7:0]. For 16-bit and 32-bit writes to the DATA register, all 10 transmit bits are written to the transmit data buffer at the same time.

If the bit values to be transmitted as the ninth and tenth bit of a new character are the same as for the previous character, it is not necessary to write to CTRL[T8] and CTRL[T9] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in CTRL[T8] and CTRL[T9] is copied at the same time data is transferred from DATA[7:0] to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. The 10-bit data mode is typically used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

### 56.4.4.2 Idle length

An idle character is a character where the start bit, all data bits and stop bits are in the mark position. The CTRL[ILT] bit can be configured to start detecting an idle character from the previous start bit (any data bits and stop bits count towards the idle character detection) or from the previous stop bit.

The number of idle characters that must be received before an idle line condition is detected can also be configured using the CTRL[IDLECFG] field. This field configures the number of idle characters that must be received before the STAT[IDLE] flag is set, the STAT[RAF] flag is cleared and the DATA[IDLINE] flag is set with the next received character.

Idle-line wakeup and idle match operation are also affected by the CTRL[IDLECFG] field. When address match or match on/off operation is enabled, setting the STAT[RWUID] bit will cause any discarded characters to be treated as if they were idle characters.

#### 56.4.4.3 Loop mode

When CTRL[LOOPS] is set, the CTRL[RSRC] bit chooses between loop mode (CTRL[RSRC] = 0) or single-wire mode (CTRL[RSRC] = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RXD pin is not used by the LPUART.

#### 56.4.4.4 Single-wire operation

When CTRL[LOOPS] is set, the CTRL[RSRC] bit chooses between loop mode (CTRL[RSRC] = 0) or single-wire mode (CTRL[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TXD pin (the RXD pin is not used).

In single-wire mode, the CTRL[TXDIR] bit controls the direction of serial data on the TXD pin. When CTRL[TXDIR] is cleared, the TXD pin is an input to the receiver and the transmitter is temporarily disconnected from the TXD pin so an external device can send serial data to the receiver. When CTRL[TXDIR] is set, the TXD pin is an output driven by the transmitter, the internal loop back connection is disabled, and as a result the receiver cannot receive characters that are sent out by the transmitter.

### 56.4.5 Infrared interface

The LPUART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the LPUART. The IrDA physical layer specification defines a half-duplex infrared communication link for exchanging data. The full standard includes data rates up to 16 Mbits/s. This module covers data rates only between 2.4 kbits/s and 115.2 kbits/s.

The LPUART has an infrared transmit encoder and receive decoder. The LPUART transmits serial bits of data that are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses are detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder, external from the LPUART. The narrow pulses are then stretched by the infrared receive decoder to get back to a serial bit stream to be received by the LPUART. The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active high pulses.

The infrared submodule receives its clock sources from the LPUART. One of these two clocks are selected in the infrared submodule to generate either 1/OSR, 2/OSR, 3/OSR, or 4/OSR narrow pulses during transmission.

#### 56.4.5.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD signal. A narrow pulse is transmitted for a 0 bit and no pulse for a 1 bit. The narrow pulse is sent at the start of the bit with a duration of 1/OSR, 2/OSR, 3/OSR, or 4/OSR of a bit time. A narrow low pulse is transmitted for a 0 bit when CTRL[TXINV] is cleared, while a narrow high pulse is transmitted for a 0 bit when CTRL[TXINV] is set.

#### 56.4.5.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each 0 received and no pulse is expected for each 1 received. A narrow low pulse is expected for a 0 bit when STAT[RXINV] is cleared, while a narrow high pulse is expected for a 0 bit when STAT[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

### 56.4.6 Interrupts and status flags

The LPUART transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty STAT[TDRE]) indicates when there is room in the transmit data buffer to write another transmit character to the DATA register. If the transmit interrupt enable CTRL[TIE]) bit is set, a hardware interrupt is requested when STAT[TDRE] is set. Transmit complete (STAT[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TXD at

the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (CTRL[TCIE]) bit is set, a hardware interrupt is requested when STAT[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the STAT[TDRE] and STAT[TC] status flags if the corresponding CTRL[TIE] or CTRL[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the receive data register by reading the DATA register. The STAT[RDRF] flag is cleared by reading the DATA register.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RXD line remains idle for an extended period of time. IDLE is cleared by writing 1 to the STAT[IDLE] flag. After STAT[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set STAT[RDRF].

If the associated error was detected in the received character that caused STAT[RDRF] to be set, the error flags - noise flag (STAT[NF]), framing error (STAT[FE]), and parity error flag (STAT[PF]) - are set at the same time as STAT[RDRF]. These flags are not set in overrun cases.

If STAT[RDRF] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (STAT[OR]) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2] then the STAT[MA1F] and/or STAT[MA2F] flags are set at the same time that STAT[RDRF] is set.

At any time, an active edge on the RXD serial data input pin causes the STAT[RXEDGIF] flag to set. The STAT[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (CTRL[RE] = 1).

## 56.4.7 Peripheral Triggers

The connection of the LPUART peripheral triggers with other peripherals are device specific.

### 56.4.7.1 Output Triggers

The LPUART generates three output triggers that can be connected to other peripherals on the device.

- The transmit word trigger asserts at the end of each transmitted word, it negates after one bit period.
- The receive word trigger asserts at the end of each received word that is written to the receive FIFO, for one oversampling clock period.
- The receive idle trigger asserts at when the idle flag would set, for one oversampling clock period.

### 56.4.7.2 Input Trigger

The LPUART supports one peripheral input trigger, that can be configured in one of the following ways.

- The input trigger can be connected in place of the CTS_B pin input. The input trigger must assert for longer than one bit clock period when the transmitter is idle with data to send to guarantee a new transmission.
- The input trigger can modulate the transmit data output (trigger is logically ANDed with the TXD output). The input trigger is expected to be generated from a PWM source with a period that is less than the bit clock frequency.
- The input trigger can be connected in place of the RXD pin input. The input trigger is expected to be generated from a receive data source, such as analog comparator or external pin.

# Chapter 57
# Low Power Serial Peripheral Interface (LPSPI)

## 57.1 Chip-specific LPSPI information

### 57.1.1 LPSPI instantiations

The following list describes the LPSPI instantions on this device:

- Instantiations: 4, LPSPI0-2 belong to CM4F domain, LPSPI3 belongs to CM0+ domain
- Receive FIFO size: 4 entries
- Transmit FIFO size: 4 entries

## 57.2 Introduction

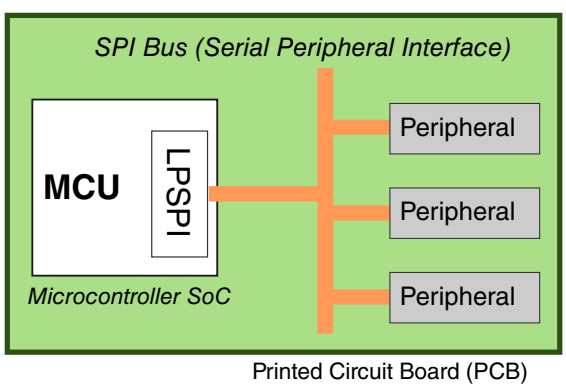



**Figure 57-1. SPI bus connects MCU to peripherals on PCB**

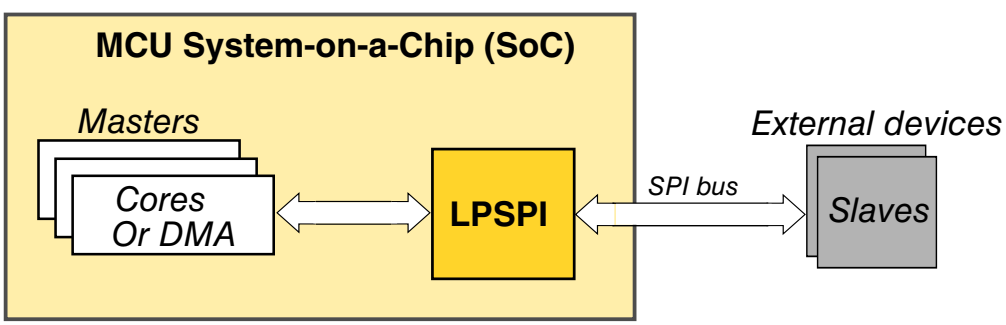
The LPSPI is a low power Serial Peripheral Interface (SPI) module that supports an efficient interface to an SPI bus, either as a master and/or as a slave.

- The LPSPI is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The LPSPI can continue operating in stop modes, if an appropriate clock is available.

**NOTE**

The Serial Peripheral Interface bus (SPI) is a synchronous serial communication interface used in embedded systems, typically to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing to Secure Digital cards and LCD displays.

- SPI devices communicate in full duplex mode using a master-slave scheme with a single master. This enables communication in both directions to happen simultaneously.
- Multiple slave devices are selected using individual slave select (SS) lines.
- The master device originates the frame for reading and writing.




*Cores or DMA in an MCU SoC use LPSPI to communicate with external devices over a serial bus.*

**Figure 57-2. LPSPI connects masters in MCU to external slaves on PCB**

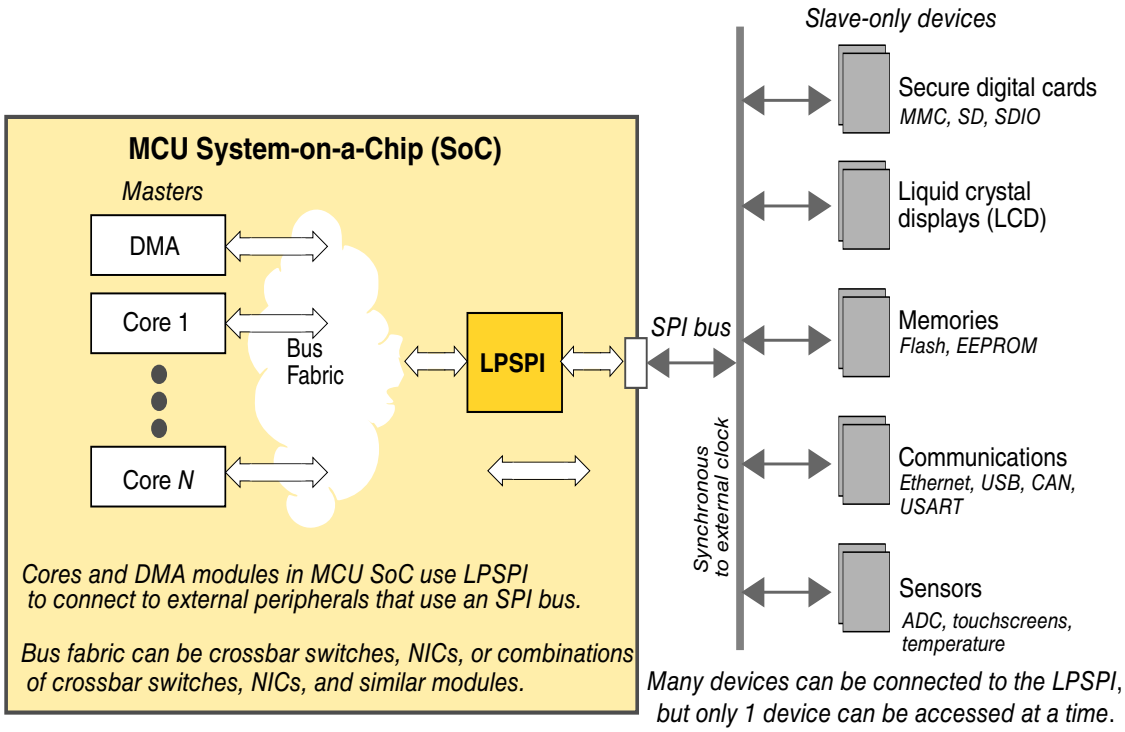



**Figure 57-3. Typical LPSPI connection scheme**

- SPI accesses are always synchronous to the external clock.
- Each SPI can only assert one chip select at a time, but technically there can be multiple SPI slaves sharing the same chip select (in the form of a larger shift register). This requires:
  - the SDO of the master to connect to the SDI of the first slave,
  - the SDO of first slave to connect to the SDI of the next slave,
  - and the SDO of last slave to connect to the SDI of the LPSPI master.

  Note that some SPI modules use MISO/MOSI for the input/output data.
- The LPSPI supports DMA accesses and generates a DMA request.

## 57.2.1 Features

The LPSPI supports:

- Word size = 32 bits
- Configurable clock polarity and clock phase
- Master operation supporting up to 4 peripheral chip select
- Slave operation
- Command/transmit FIFO of 4 words
- Receive FIFO of 4 words

- Flexible timing parameters in master mode, including SCK frequency and delays between PCS and SCK edges
- Support for full duplex transfers supporting 1-bit transmit and receive on each clock edge
- Support for half duplex transfers supporting 1-bit transmit or receive on each clock edge
- Support for half duplex transfers supporting 2-bit or 4-bit transmit or receive on each clock edge (master only)
- Host request input can be used to control the start time of an SPI bus transfer (master only)
- Receive data match logic supporting wakeup on data match

## 57.2.2 Block Diagram

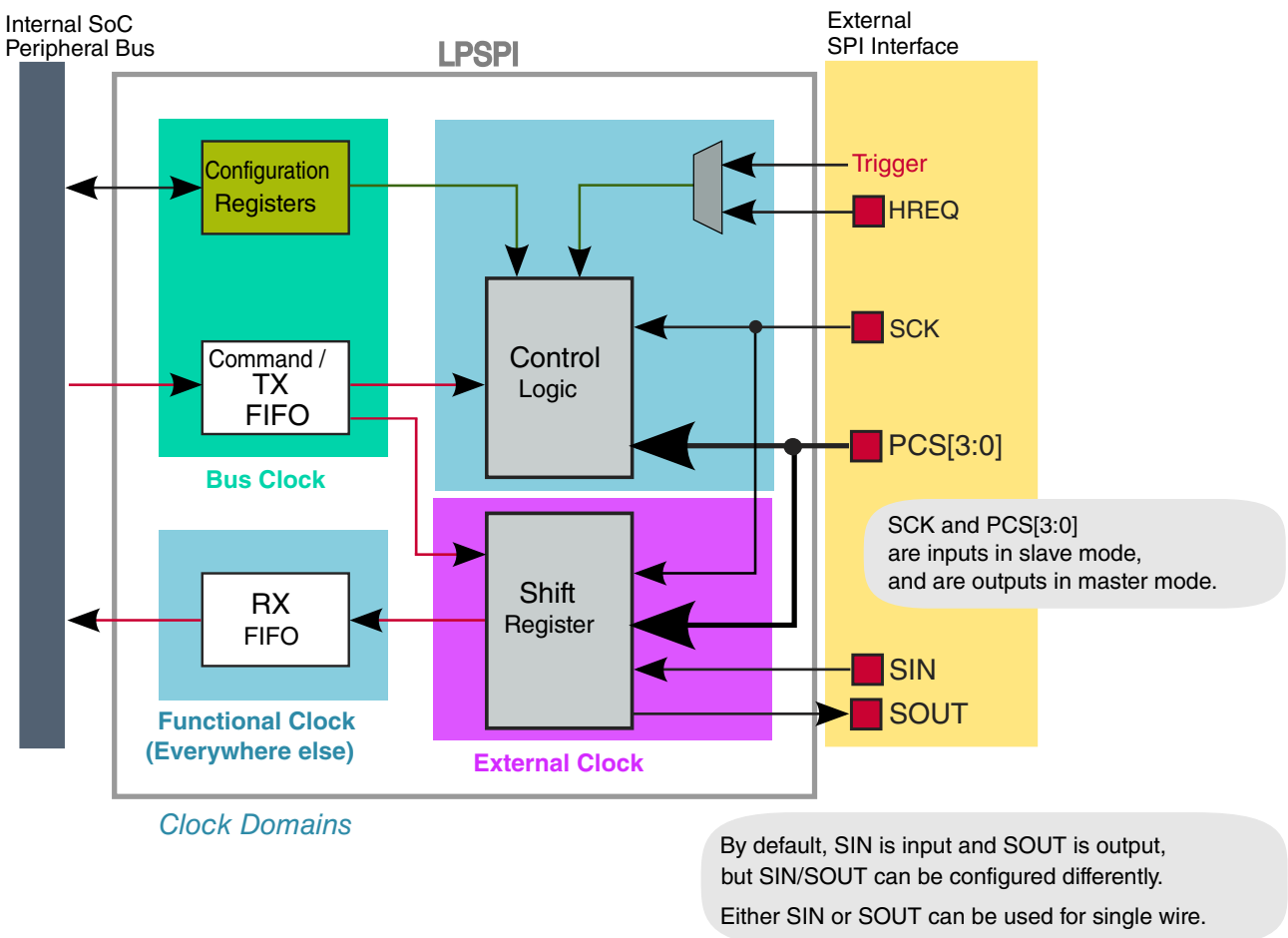



**Figure 57-4. Block Diagram**

## 57.2.3 Modes of operation

**Table 57-1. Chip modes supported by the LPSPI module**

| Chip mode | LPSPI Operation |
|---|---|
| Run | Normal operation |
| Stop/Wait | Can continue operating in stop mode if the Doze Enable bit (CR[DOZEN]) is clear and the LPSPI is using an external or internal clock source that remains operating during stop/wait modes. |
| Low Power Stop (also called Low Leakage Stop or LLS) | Before entering low power stop mode, the LPSPI will wait for the current transfer to finish any pending operation, temporarily ignoring the Doze Enable (CR[DOZEN]) bit. |
| Debug (the core is in Debug/Halted mode) | Can continue operating in debug mode, if the Debug Enable bit (CR[DBGEN]) is set. |

## 57.2.4 Signal Descriptions

**Table 57-2. Signals**

| Signal | Name | Description | I/O |
|---|---|---|---|
| SCK | Serial clock | • Input in slave mode<br>• Output in master mode | I/O |
| PCS[0] | Peripheral Chip Select | • Input in slave mode<br>• Output in master mode | I/O |
| PCS[1] / HREQ | Peripheral Chip Select<br>or<br>Host Request | Host Request pin is selected when HREN=1 and HRSEL=0<br>• Input in either slave mode or when used as Host Request<br>• Output in master mode | I/O |
| PCS[2] / DATA[2] | Peripheral Chip Select<br>or<br>data pin 2 during quad-data transfers | • Input in slave mode<br>• Output in master mode<br>• Input in quad-data receive transfers<br>• Output in quad-data transmit transfers | I/O |
| PCS[3] / DATA[3] | Peripheral Chip Select<br>or<br>data pin 3 during quad-data transfers | • Input in slave mode<br>• Output in master mode<br>• Input in quad-data receive transfers<br>• Output in quad-data transmit transfers | I/O |
| SOUT / DATA[0] | Serial Data Output | Can be configured as serial data input signal<br>• Used as data pin 0 in quad-data transfers<br>• Used as data pin 0 in dual-data transfers | I/O |
| SIN / DATA[1] | Serial Data Input | Can be configured as serial data output signal<br>• Used as data pin 1 in quad-data transfers<br>• Used as data pin 1 in dual-data transfers | I/O |

## 57.2.5 Wiring options

LPSPI can be used to implement typical (dedicated slave select lines) or daisy-chain (shared slave select lines) SPI busses.

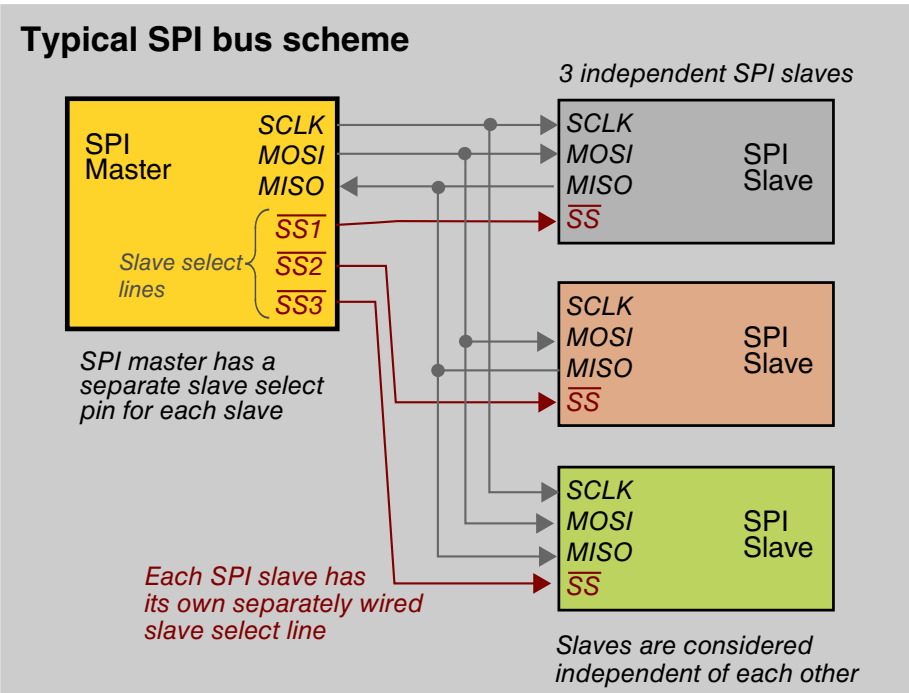



**Figure 57-5. Dedicated slave select lines scheme**

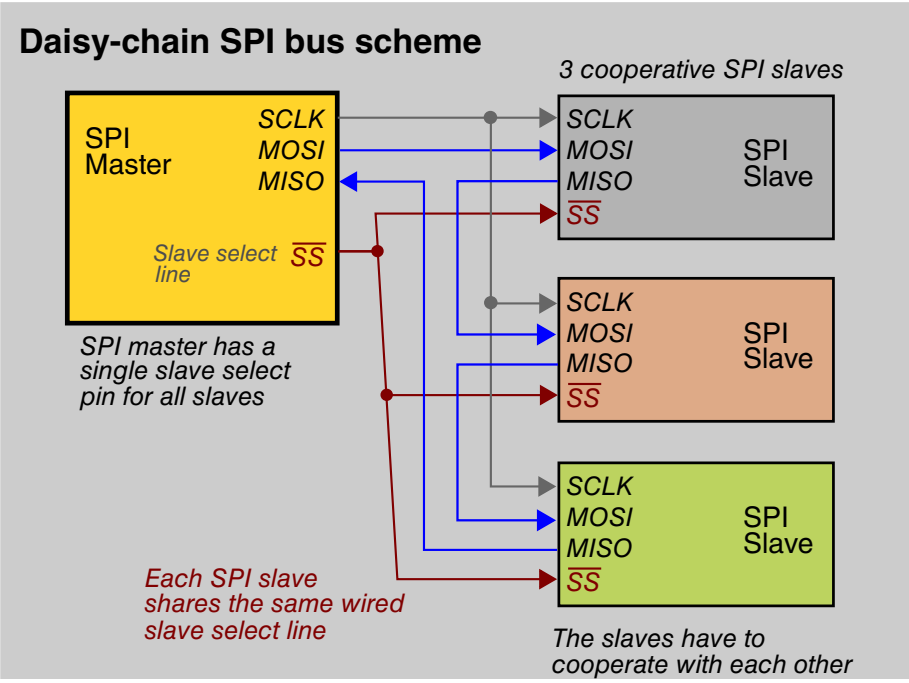



**Figure 57-6. Shared slave select lines (daisy-chain) scheme**

**Table 57-3. Wiring Scheme Differences**

| Item | Dedicated slave select lines | Shared slave select lines (daisy-chain) |
|---|---|---|
| slave select lines (pins on the master device) | • 1 slave select line (pin) for each slave; the scheme is limited by how many lines (pins) that the master device can support | • 1 slave select line (pin) for multiple slaves |
| data flow from master device to slave devices | • Master sends data directly to the slave that the data is for; the slaves for whom the data is not intended, simply ignore the data; this implies that those slaves could be asleep (if desired) | • Master sends data and multiple slaves have to forward the data through the daisy chain; this implies that all slaves must be on and awake when the master sends data to any slave |

# 57.3 Memory Map and Registers

## 57.3.1 LPSPI0 register descriptions

### 57.3.1.1 LPSPI0 Memory map

LPSPI0 base address: 4003_F000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0004h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Control Register (CR) | 32 | RW | 0000_0000h |
| 14h | Status Register (SR) | 32 | W1C | 0000_0001h |
| 18h | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable Register (DER) | 32 | RW | 0000_0000h |
| 20h | Configuration Register 0 (CFGR0) | 32 | RW | 0000_0000h |
| 24h | Configuration Register 1 (CFGR1) | 32 | RW | 0000_0000h |
| 30h | Data Match Register 0 (DMR0) | 32 | RW | 0000_0000h |
| 34h | Data Match Register 1 (DMR1) | 32 | RW | 0000_0000h |
| 40h | Clock Configuration Register (CCR) | 32 | RW | 0000_0000h |
| 58h | FIFO Control Register (FCR) | 32 | RW | 0000_0000h |
| 5Ch | FIFO Status Register (FSR) | 32 | RO | 0000_0000h |
| 60h | Transmit Command Register (TCR) | 32 | RW | 0000_001Fh |
| 64h | Transmit Data Register (TDR) | 32 | WO | 0000_0000h |
| 70h | Receive Status Register (RSR) | 32 | RO | 0000_0002h |
| 74h | Receive Data Register (RDR) | 32 | RO | 0000_0000h |

## 57.3.1.2 Version ID Register (VERID)

### 57.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 57.3.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 57.3.1.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module specification. Read-only field. |
| 15-0<br>FEATURE | Module Identification Number<br>Returns the feature set number. Read-only field.<br>0000000000000100b - Standard feature set supporting a 32-bit shift register. |

### 57.3.1.3 Parameter Register (PARAM)

#### 57.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 57.3.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 57.3.1.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Sets the maximum number of words in the receive FIFO, which is $2^{RXFIFO}$ |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Sets the maximum number of words in the transmit FIFO, which is $2^{TXFIFO}$ |

## 57.3.1.4 Control Register (CR)

### 57.3.1.4.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 57.3.1.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | DBGEN | DOZEN | RST | MEN |
| W | | | | | | | RRF | RTF | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Module is disabled in debug mode<br>1b - Module is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the module in Doze mode<br>0b - Module is enabled in Doze mode<br>1b - Module is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |
| 0<br>MEN | Module Enable<br>0b - Module is disabled<br>1b - Module is enabled |

## 57.3.1.5 Status Register (SR)

### 57.3.1.5.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 57.3.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMF | REF | TEF | TCF | FCF | WCF | 0 | | | | | | RDF | TDF |
| W | | | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 57.3.1.5.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MBF | Module Busy Flag<br>0b - LPSPI is idle<br>1b - LPSPI is busy |
| 23-14<br>— | Reserved |
| 13<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by CFGR1[MATCFG], Configuration Register 1).<br>0b - Have not received matching data<br>1b - Have received matching data |
| 12<br>REF | Receive Error Flag<br>The Receive Error Flag will set when the Receiver FIFO overflows. When the Receive Error Flag is set, it is recommended to first end the transfer, empty the Receive FIFO, clear the Receive Error Flag and then restart the transfer from the beginning.<br>0b - Receive FIFO has not overflowed<br>1b - Receive FIFO has overflowed |
| 11<br>TEF | Transmit Error Flag<br>The Transmit Error Flag will set when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning.<br>0b - Transmit FIFO underrun has not occurred<br>1b - Transmit FIFO underrun has occurred |
| 10<br>TCF | Transfer Complete Flag<br>In master mode when the LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag will set.<br>0b - All transfers have not completed<br>1b - All transfers have completed |
| 9<br>FCF | Frame Complete Flag<br>The Frame Complete Flag will set at the end of each frame transfer, when the PCS negates.<br>0b - Frame transfer has not completed<br>1b - Frame transfer has completed |
| 8<br>WCF | Word Complete Flag<br>The Word Complete Flag will set when the last bit of a received word is sampled.<br>0b - Transfer of a received word has not yet completed<br>1b - Transfer of a received word has completed |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than FCR[RXWATER] (FIFO Control Register)<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0 | Transmit Data Flag |

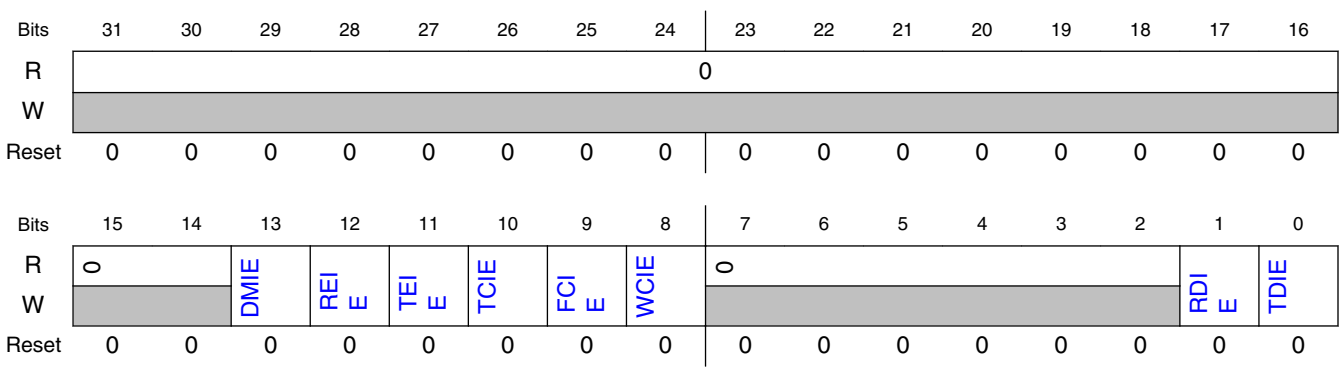
| Field | Function |
|---|---|
| TDF | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than FCR[TXWATER] (FIFO Control Register)<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 57.3.1.6 Interrupt Enable Register (IER)

### 57.3.1.6.1 Offset

| Register | Offset |
|---|---|
| IER | 18h |

### 57.3.1.6.2 Diagram




### 57.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13<br>DMIE | Data Match Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>REIE | Receive Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 11<br>TEIE | Transmit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

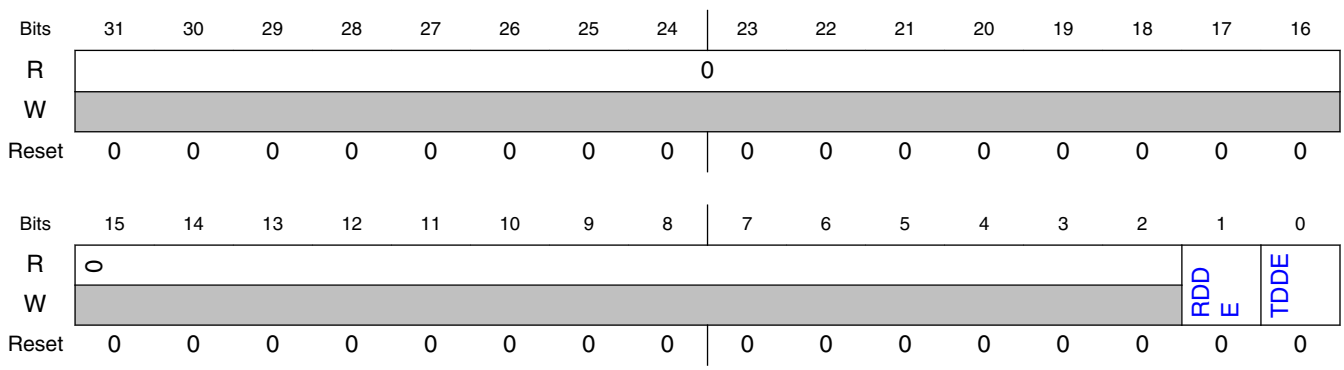
| Field | Function |
|---|---|
| 10<br>TCIE | Transfer Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>FCIE | Frame Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>WCIE | Word Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 57.3.1.7 DMA Enable Register (DER)

### 57.3.1.7.1 Offset

| Register | Offset |
|---|---|
| DER | 1Ch |

### 57.3.1.7.2 Diagram



#### 57.3.1.7.3 Fields

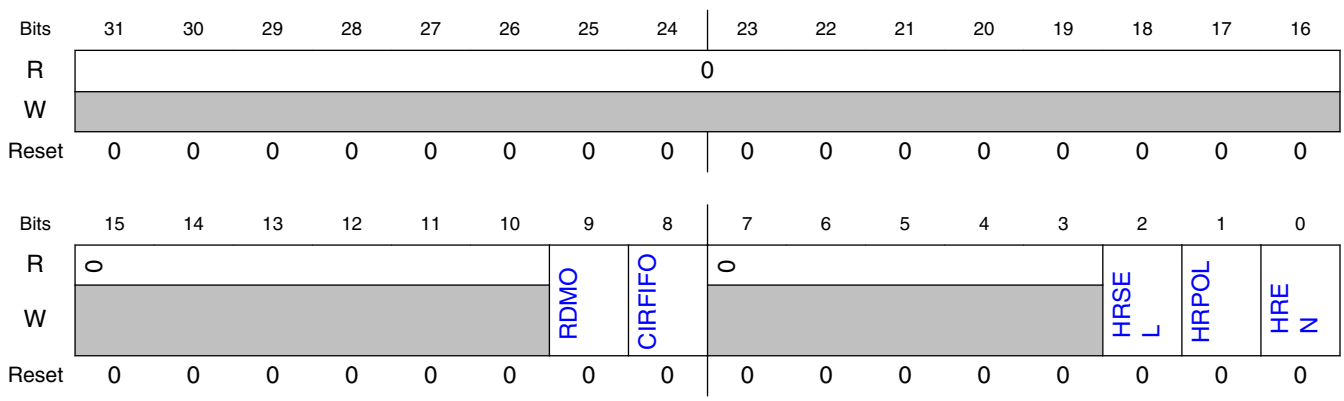
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

### 57.3.1.8 Configuration Register 0 (CFGR0)

#### 57.3.1.8.1 Offset

| Register | Offset |
|---|---|
| CFGR0 | 20h |

#### 57.3.1.8.2 Diagram




#### 57.3.1.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9 | Receive Data Match Only |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RDMO | When Receive Data Match Only is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded.<br>• Receive Data Match Only bit should be set when the LPSPI is idle and the Data Match Flag is clear<br>• After the Data Match Flag (DMF) is set, the Receive Data Match Only bit configuration is ignored<br>• When disabling RDMO and to ensure that no receive data is lost, before clearing the Data Match Flag, first clear Receive Data Match Only (RDMO)<br><br>0b - Received data is stored in the receive FIFO as in normal operations<br>1b - Received data is discarded unless the Data Match Flag (DMF) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as it normally is, but after the LPSPI is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This restoring of the read pointer will cause the contents of the transmit FIFO to be cycled through repeatedly.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When the host request function is enabled with the LPSPI_HREQ pin, the LPSPI_PCS[1] function is disabled.<br>0b - Host request input is the LPSPI_HREQ pin<br>1b - Host request input is the input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request pin.<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable<br>When enabled in master mode, the LPSPI will only start a new SPI bus transfer if the host request input is asserted. When the LPSPI is busy, the host request input is ignored.<br>0b - Host request is disabled<br>1b - Host request is enabled |

## 57.3.1.9 Configuration Register 1 (CFGR1)

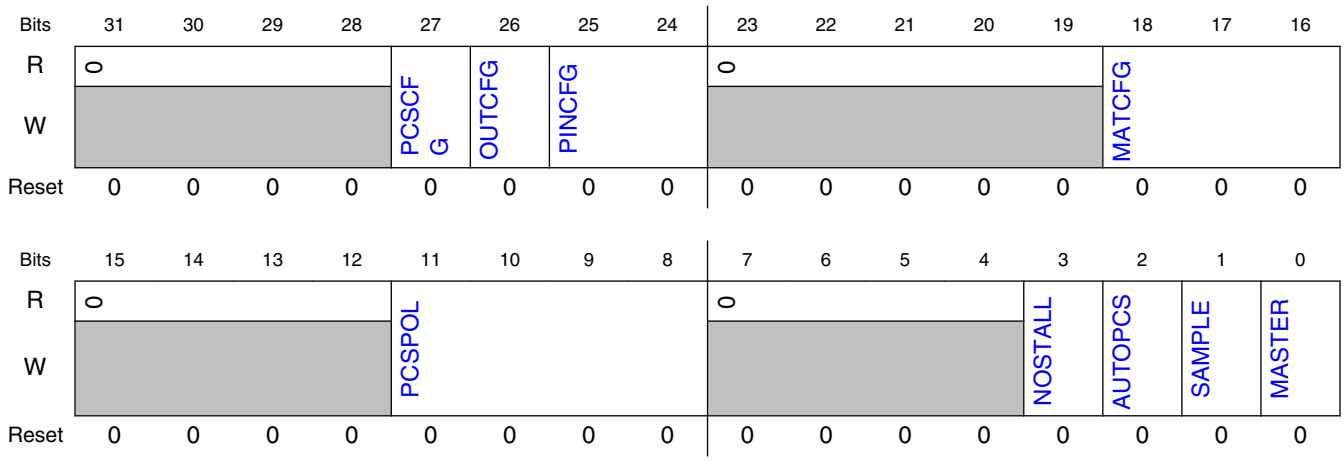
### 57.3.1.9.1 Offset

| Register | Offset |
|---|---|
| CFGR1 | 24h |

### 57.3.1.9.2 Function

The CFGR1 should only be written when the LPSPI is disabled.

## 57.3.1.9.3 Diagram




## 57.3.1.9.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>PCSCFG | Peripheral Chip Select Configuration<br>If performing 4-bit transfers, the Peripheral Chip Select Configuration bit must be set.<br>0b - PCS[3:2] are enabled<br>1b - PCS[3:2] are disabled |
| 26<br>OUTCFG | Output Config<br>Configures if the output data is tristated between accesses (LPSPI_PCS is negated).<br>0b - Output data retains last value when chip select is negated<br>1b - Output data is tristated when chip select is negated |
| 25-24<br>PINCFG | Pin Configuration<br>Configures which pins are used for input and output data during single bit transfers.<br>00b - SIN is used for input data and SOUT is used for output data<br>01b - SIN is used for both input and output data<br>10b - SOUT is used for both input and output data<br>11b - SOUT is used for input data and SIN is used for output data |
| 23-19<br>— | Reserved |
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that will cause the DMF to set.<br>**NOTE:** *Syntax:* * is boolean AND, + is boolean OR<br>000b - Match is disabled<br>001b - Reserved<br>010b - 010b - Match is enabled, if 1st data word equals MATCH0 OR MATCH1, i.e., (1st data word = MATCH0 + MATCH1) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 011b - 011b - Match is enabled, if any data word equals MATCH0 OR MATCH1, i.e., (any data word = MATCH0 + MATCH1)<br>100b - 100b - Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, i.e., [(1st data word = MATCH0) * (2nd data word = MATCH1)]<br>101b - 101b - Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, i.e., [(any data word = MATCH0) * (next data word = MATCH1)]<br>110b - 110b - Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(1st data word * MATCH1) = (MATCH0 * MATCH1)]<br>111b - 111b - Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(any data word * MATCH1) = (MATCH0 * MATCH1)] |
| 15-12<br>— | Reserved |
| 11-8<br>PCSPOL | Peripheral Chip Select Polarity<br>Configures the polarity of each Peripheral Chip Select pin.<br>0000b - The Peripheral Chip Select pin PCSx is active low<br>0001b - The Peripheral Chip Select pin PCSx is active high |
| 7-4<br>— | Reserved |
| 3<br>NOSTALL | No Stall<br>In master mode, the LPSPI will stall transfers when the transmit FIFO is empty or when the receive FIFO is full, ensuring that no transmit FIFO underrun or receive FIFO overrun can occur. Setting the No Stall bit will disable this functionality.<br>0b - Transfers will stall when the transmit FIFO is empty or the receive FIFO is full<br>1b - Transfers will not stall, allowing transmit FIFO underruns or receive FIFO overruns to occur |
| 2<br>AUTOPCS | Automatic PCS<br>For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the Automatic PCS bit will cause the LPSPI to generate an internal PCS signal at the end of each transfer word when the Clock Phase bit TCR[CPHA] = 1.<br>• When the Automatic PCS bit is set, the SCK must remain idle for at least 4 LPSPI functional clock cycles (divided by the Prescaler Value TCR[PRESCALE] configuration) between each word, to ensure correct operations<br>• In master mode, the Automatic PCS bit is ignored<br>0b - Automatic PCS generation is disabled<br>1b - Automatic PCS generation is enabled |
| 1<br>SAMPLE | Sample Point<br>When set, the LPSPI master will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data.<br>• The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode<br>• In slave mode, the SAMPLE bit is ignored<br>0b - Input data is sampled on SCK edge<br>1b - Input data is sampled on delayed SCK edge |
| 0<br>MASTER | Master Mode<br>Configures the LPSPI in master or slave mode. The Master Mode bit directly controls the direction of the LPSPI_SCK and LPCPI_PCS pins.<br>0b - Slave mode<br>1b - Master mode |

## 57.3.1.10 Data Match Register 0 (DMR0)

### 57.3.1.10.1 Offset

| Register | Offset |
|---|---|
| DMR0 | 30h |

### 57.3.1.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.1.10.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH0 | Match 0 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 0 Value is compared against the received data. |

## 57.3.1.11 Data Match Register 1 (DMR1)

### 57.3.1.11.1 Offset

| Register | Offset |
|---|---|
| DMR1 | 34h |

### 57.3.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.1.11.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH1 | Match 1 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 1 Value is compared against the received data. |

## 57.3.1.12 Clock Configuration Register (CCR)

### 57.3.1.12.1 Offset

| Register | Offset |
|---|---|
| CCR | 40h |

### 57.3.1.12.2 Function

The Clock Configuration Register is only used in master mode, and the Clock Configuration Register cannot be changed when the LPSPI is enabled.

## 57.3.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBT | | | | | | | | SCKDIV | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 57.3.1.12.4 Fields

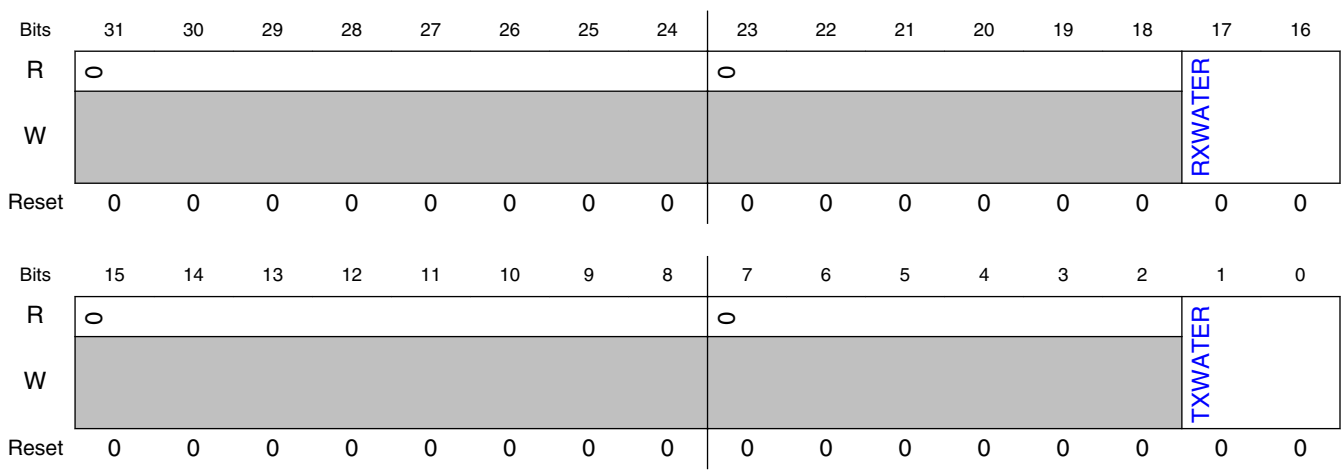
| Field | Function |
|---|---|
| 31-24<br>SCKPCS | SCK-to-PCS Delay<br>In master mode: configures the delay from the last SCK edge to the PCS negation.<br>• The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 23-16<br>PCSSCK | PCS-to-SCK Delay<br>In master mode: configures the delay from the PCS assertion to the first SCK edge.<br>• The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 15-8<br>DBT | Delay Between Transfers<br>In master mode:<br>• Configures the delay from the PCS negation to the next PCS assertion.<br>  • The delay is equal to (DBT + 2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 2 cycles.<br>  • Half of the delay occurs before PCS assertion and the other half of the delay occurs after PCS negation. If the command word is updated between 2 transfers, then the command word is updated half-way between the PCS negation of the last transfer and PCS assertion of the next transfer.<br>  • The command word sets which PCS signal is used (of PCS[3:0]}, the polarity/phase of the SCK signal, and the Prescaler Value.<br>• Configures the delay from the last SCK edge of a transfer word and the first SCK edge of the next transfer word, in a continuous transfer.<br>  • The delay is equal to (DBT + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 1 cycle. |
| 7-0<br>SCKDIV | SCK Divider<br>In master mode, the SCK Divider configures the divide ratio of the SCK pin.<br>• The SCK period is equal to (SCKDIV+2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum SCK period is 2 cycles.<br>• If the SCK period is an odd number of cycles, then the 1st half of the SCK period will be 1 cycle longer than the 2nd half of the SCK period. |

## 57.3.1.13 FIFO Control Register (FCR)

### 57.3.1.13.1 Offset

| Register | Offset |
|---|---|
| FCR | 58h |

### 57.3.1.13.2 Diagram




### 57.3.1.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal or greater than the FIFO size will be truncated. |
| 15-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1-0 | Transmit FIFO Watermark |

| Field | Function |
|---|---|
| TXWATER | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal or greater than the FIFO size will be truncated. |

## 57.3.1.14 FIFO Status Register (FSR)

### 57.3.1.14.1 Offset

| Register | Offset |
|---|---|
| FSR | 5Ch |

### 57.3.1.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words currently stored in the receive FIFO. |
| 15-8<br>— | Reserved |
| 7-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words currently stored in the transmit FIFO. |

## 57.3.1.15 Transmit Command Register (TCR)

### 57.3.1.15.1 Offset

| Register | Offset |
|---|---|
| TCR | 60h |

### 57.3.1.15.2 Function

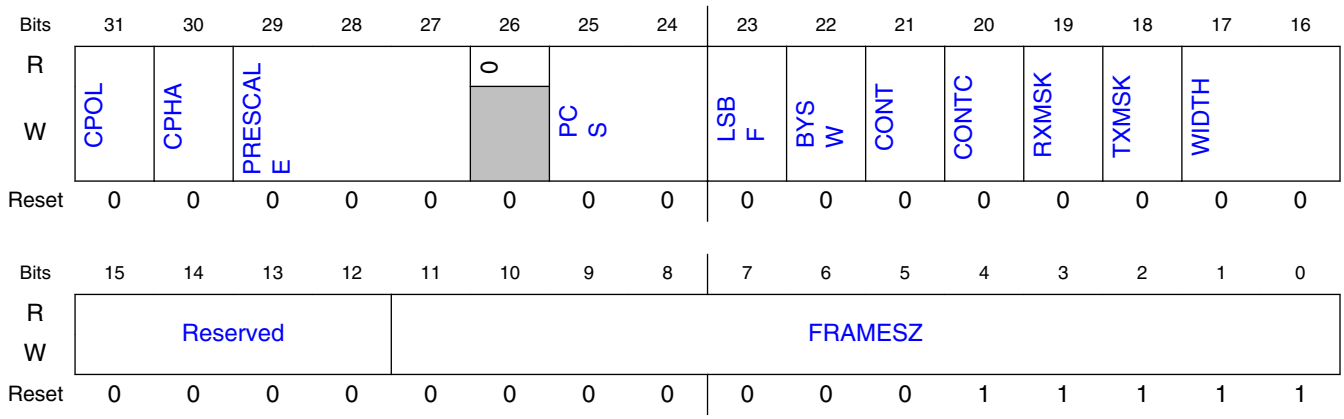
Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that the data are written. Command Register writes will be tagged and cause the command register to update, after that entry reaches the top of the FIFO. This allows changes to the command word and the transmit data itself to be interleaved. Changing the command word will cause all subsequent SPI bus transfers to be performed using the new command word.

- **In master mode,** writing a new command word does not initiate a new transfer, unless TXMSK is set. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK set). Hardware will clear TXMSK when the LPSPI_PCS negates.
- **In master mode,** if the command word is changed before an existing frame has completed, then the existing frame will terminate and the command word will then update. The command word can be changed during a continuous transfer, if CONTC of the new command word is set and the command word is written on a frame size boundary.
- **In slave mode,** the command word should be changed only when the LPSPI is idle and there is no SPI bus transfer.

**Avoid register reading problems:** Reading the Transmit Command Register will return the current state of the command register. Reading the Transmit Command Register at the same time that the Transmit Command Register is loaded from the transmit FIFO, can return an incorrect Transmit Command Register value. It is recommended:

- to either read the Transmit Command Register when the transmit FIFO is empty,
- or to read the Transmit Command Register more than once and then compare the returned values.

### 57.3.1.15.3 Diagram




### 57.3.1.15.4 Fields

| Field | Function |
|---|---|
| 31<br>CPOL | Clock Polarity<br>The Clock Polarity field is only updated between frames.<br>0b - The inactive state value of SCK is low<br>1b - The inactive state value of SCK is high |
| 30<br>CPHA | Clock Phase<br>The Clock Phase field is only updated between frames.<br>0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK<br>1b - Data is changed on the leading edge of SCK and captured on the following edge of SCK |
| 29-27<br>PRESCALE | Prescaler Value<br>For all SPI bus transfers, the Prescaler value applied to the clock configuration register. The Prescaler Value field is only updated between frames.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |
| 26<br>— | Reserved |
| 25-24<br>PCS | Peripheral Chip Select<br>Configures the peripheral chip select used for the transfer. The Peripheral Chip Select field is only updated between frames.<br>00b - Transfer using LPSPI_PCS[0]<br>01b - Transfer using LPSPI_PCS[1]<br>10b - Transfer using LPSPI_PCS[2]<br>11b - Transfer using LPSPI_PCS[3] |
| 23 | LSB First |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LSBF | 0b - Data is transferred MSB first<br>1b - Data is transferred LSB first |
| 22<br>BYSW | Byte Swap<br>Byte swap will swap the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers).<br>0b - Byte swap is disabled<br>1b - Byte swap is enabled |
| 21<br>CONT | Continuous Transfer<br>• In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.<br>• In slave mode, when continuous transfer is enabled, the LPSPI will only transmit the first FRAMESZ bits; after which the LPSPI will transmit received data (assuming a 32-bit shift register).<br><br>0b - Continuous transfer is disabled<br>1b - Continuous transfer is enabled |
| 20<br>CONTC | Continuing Command<br>In master mode, the Continuing Command bit allows the command word to be changed within a continuous transfer.<br>• The initial command word must enable continuous transfer (CONT=1),<br>• the continuing command must set this bit (CONTC=1),<br>• and the continuing command word must be loaded on a frame size boundary.<br><br>For example, if the continuous transfer has a frame size of 64-bits, then a continuing command word must be loaded on a 64-bit boundary.<br><br>0b - Command word for start of new transfer<br>1b - Command word for continuing transfer |
| 19<br>RXMSK | Receive Data Mask<br>When set, receive data is masked (receive data is not stored in receive FIFO).<br>0b - Normal transfer<br>1b - Receive data is masked |
| 18<br>TXMSK | Transmit Data Mask<br>When set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer.<br>0b - Normal transfer<br>1b - Mask transmit data |
| 17-16<br>WIDTH | Transfer Width<br>For 2-bit or 4-bit transfers, either Receive Data Mask (RXMSK) or Transmit Data Mask (TXMSK) must be set.<br>00b - 1 bit transfer<br>01b - 2 bit transfer<br>10b - 4 bit transfer<br>11b - Reserved |
| 15-12<br>— | Reserved. Software should only write zero to this field. |
| 11-0<br>FRAMESZ | Frame Size<br>Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. |

| Field | Function |
|---|---|
| | • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32 bits, and the 3rd word is 8 bits. |

## 57.3.1.16 Transmit Data Register (TDR)

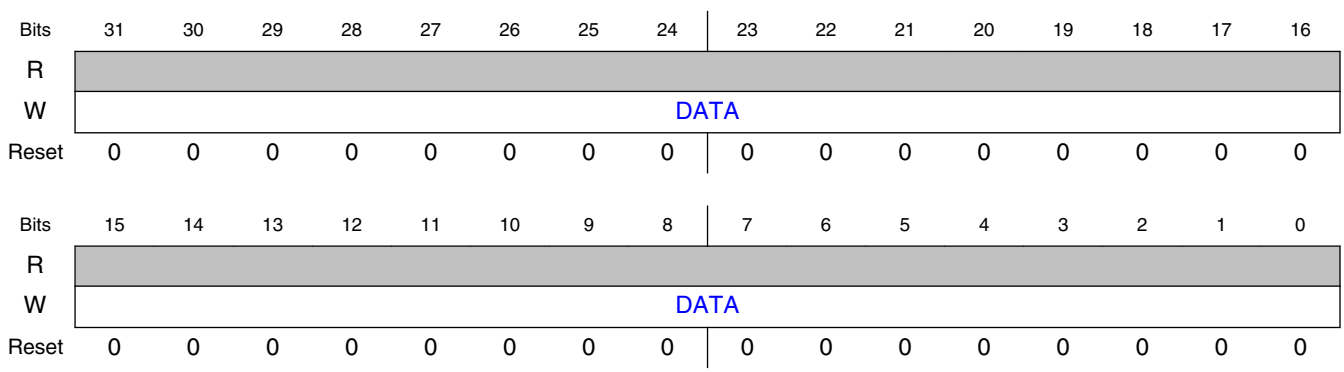
### 57.3.1.16.1 Offset

| Register | Offset |
|---|---|
| TDR | 64h |

### 57.3.1.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that it (the data) was written.

### 57.3.1.16.3 Diagram




### 57.3.1.16.4 Fields

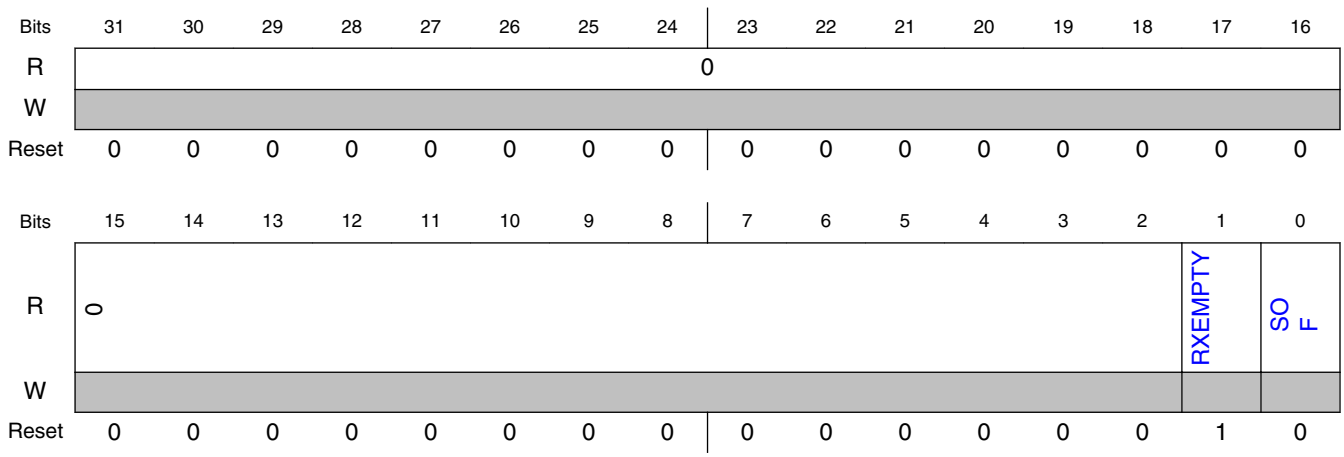
| Field | Function |
|---|---|
| 31-0<br>DATA | Transmit Data<br>Both 8-bit and 16-bit writes of transmit data will zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes. |

# 57.3.1.17 Receive Status Register (RSR)

## 57.3.1.17.1 Offset

| Register | Offset |
|---|---|
| RSR | 70h |

## 57.3.1.17.2 Diagram




## 57.3.1.17.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RXEMPTY | RX FIFO Empty<br>0b - RX FIFO is not empty<br>1b - RX FIFO is empty |
| 0<br>SOF | Start Of Frame<br>Indicates that this is the first data word received after LPSPI_PCS assertion.<br>0b - Subsequent data word received after LPSPI_PCS assertion<br>1b - First data word received after LPSPI_PCS assertion |

# 57.3.1.18 Receive Data Register (RDR)

#### 57.3.1.18.1 Offset

| Register | Offset |
|---|---|
| RDR | 74h |

#### 57.3.1.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 57.3.1.18.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

## 57.3.2 LPSPI1 register descriptions

### 57.3.2.1 LPSPI1 Memory map

LPSPI1 base address: 4004_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0004h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Control Register (CR) | 32 | RW | 0000_0000h |
| 14h | Status Register (SR) | 32 | W1C | 0000_0001h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 18h | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable Register (DER) | 32 | RW | 0000_0000h |
| 20h | Configuration Register 0 (CFGR0) | 32 | RW | 0000_0000h |
| 24h | Configuration Register 1 (CFGR1) | 32 | RW | 0000_0000h |
| 30h | Data Match Register 0 (DMR0) | 32 | RW | 0000_0000h |
| 34h | Data Match Register 1 (DMR1) | 32 | RW | 0000_0000h |
| 40h | Clock Configuration Register (CCR) | 32 | RW | 0000_0000h |
| 58h | FIFO Control Register (FCR) | 32 | RW | 0000_0000h |
| 5Ch | FIFO Status Register (FSR) | 32 | RO | 0000_0000h |
| 60h | Transmit Command Register (TCR) | 32 | RW | 0000_001Fh |
| 64h | Transmit Data Register (TDR) | 32 | WO | 0000_0000h |
| 70h | Receive Status Register (RSR) | 32 | RO | 0000_0002h |
| 74h | Receive Data Register (RDR) | 32 | RO | 0000_0000h |

## 57.3.2.2 Version ID Register (VERID)

### 57.3.2.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 57.3.2.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 57.3.2.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module specification. Read-only field. |
| 15-0<br>FEATURE | Module Identification Number<br>Returns the feature set number. Read-only field.<br>0000000000000100b - Standard feature set supporting a 32-bit shift register. |

### 57.3.2.3 Parameter Register (PARAM)

#### 57.3.2.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 57.3.2.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 57.3.2.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Sets the maximum number of words in the receive FIFO, which is $2^{RXFIFO}$ |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Sets the maximum number of words in the transmit FIFO, which is $2^{TXFIFO}$ |

## 57.3.2.4 Control Register (CR)

### 57.3.2.4.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 57.3.2.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | DBGEN | DOZEN | RST | MEN |
| W | | | | | | | RRF | RTF | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.2.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Module is disabled in debug mode<br>1b - Module is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the module in Doze mode<br>0b - Module is enabled in Doze mode<br>1b - Module is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |
| 0<br>MEN | Module Enable<br>0b - Module is disabled<br>1b - Module is enabled |

## 57.3.2.5 Status Register (SR)

### 57.3.2.5.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 57.3.2.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMF | REF | TEF | TCF | FCF | WCF | 0 | | | | | | RDF | TDF |
| W | | | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 57.3.2.5.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MBF | Module Busy Flag<br>0b - LPSPI is idle<br>1b - LPSPI is busy |
| 23-14<br>— | Reserved |
| 13<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by CFGR1[MATCFG], Configuration Register 1).<br>0b - Have not received matching data<br>1b - Have received matching data |
| 12<br>REF | Receive Error Flag<br>The Receive Error Flag will set when the Receiver FIFO overflows. When the Receive Error Flag is set, it is recommended to first end the transfer, empty the Receive FIFO, clear the Receive Error Flag and then restart the transfer from the beginning.<br>0b - Receive FIFO has not overflowed<br>1b - Receive FIFO has overflowed |
| 11<br>TEF | Transmit Error Flag<br>The Transmit Error Flag will set when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning.<br>0b - Transmit FIFO underrun has not occurred<br>1b - Transmit FIFO underrun has occurred |
| 10<br>TCF | Transfer Complete Flag<br>In master mode when the LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag will set.<br>0b - All transfers have not completed<br>1b - All transfers have completed |
| 9<br>FCF | Frame Complete Flag<br>The Frame Complete Flag will set at the end of each frame transfer, when the PCS negates.<br>0b - Frame transfer has not completed<br>1b - Frame transfer has completed |
| 8<br>WCF | Word Complete Flag<br>The Word Complete Flag will set when the last bit of a received word is sampled.<br>0b - Transfer of a received word has not yet completed<br>1b - Transfer of a received word has completed |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than FCR[RXWATER] (FIFO Control Register)<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0 | Transmit Data Flag |

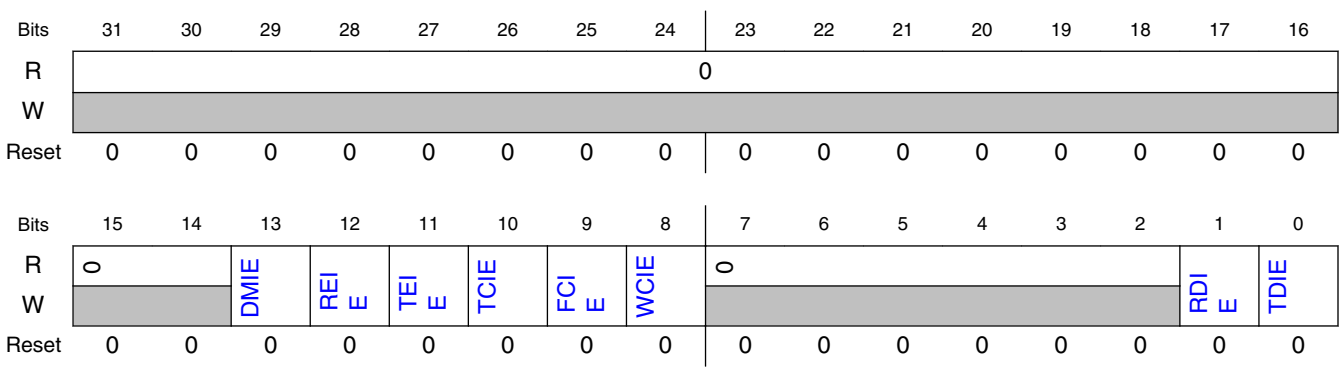
| Field | Function |
|---|---|
| TDF | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than FCR[TXWATER] (FIFO Control Register)<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 57.3.2.6 Interrupt Enable Register (IER)

### 57.3.2.6.1 Offset

| Register | Offset |
|---|---|
| IER | 18h |

### 57.3.2.6.2 Diagram




### 57.3.2.6.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13<br>DMIE | Data Match Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>REIE | Receive Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 11<br>TEIE | Transmit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>TCIE | Transfer Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>FCIE | Frame Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>WCIE | Word Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 57.3.2.7 DMA Enable Register (DER)

### 57.3.2.7.1 Offset

| Register | Offset |
|---|---|
| DER | 1Ch |

### 57.3.2.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RDDE | TDDE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 57.3.2.7.3 Fields

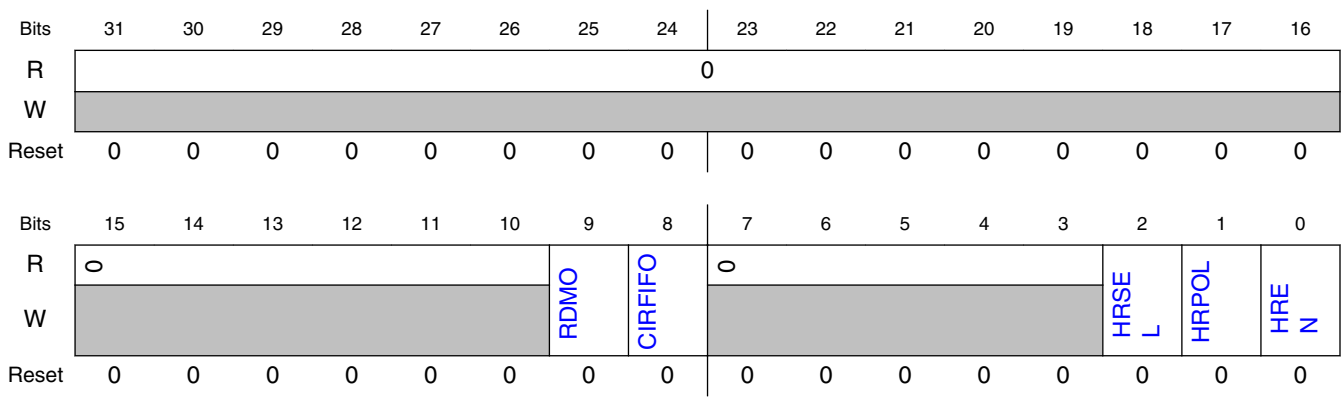
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

### 57.3.2.8 Configuration Register 0 (CFGR0)

#### 57.3.2.8.1 Offset

| Register | Offset |
|---|---|
| CFGR0 | 20h |

#### 57.3.2.8.2 Diagram




#### 57.3.2.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9 | Receive Data Match Only |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RDMO | When Receive Data Match Only is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded.<br>• Receive Data Match Only bit should be set when the LPSPI is idle and the Data Match Flag is clear<br>• After the Data Match Flag (DMF) is set, the Receive Data Match Only bit configuration is ignored<br>• When disabling RDMO and to ensure that no receive data is lost, before clearing the Data Match Flag, first clear Receive Data Match Only (RDMO)<br><br>0b - Received data is stored in the receive FIFO as in normal operations<br>1b - Received data is discarded unless the Data Match Flag (DMF) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as it normally is, but after the LPSPI is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This restoring of the read pointer will cause the contents of the transmit FIFO to be cycled through repeatedly.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When the host request function is enabled with the LPSPI_HREQ pin, the LPSPI_PCS[1] function is disabled.<br>0b - Host request input is the LPSPI_HREQ pin<br>1b - Host request input is the input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request pin.<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable<br>When enabled in master mode, the LPSPI will only start a new SPI bus transfer if the host request input is asserted. When the LPSPI is busy, the host request input is ignored.<br>0b - Host request is disabled<br>1b - Host request is enabled |

## 57.3.2.9 Configuration Register 1 (CFGR1)

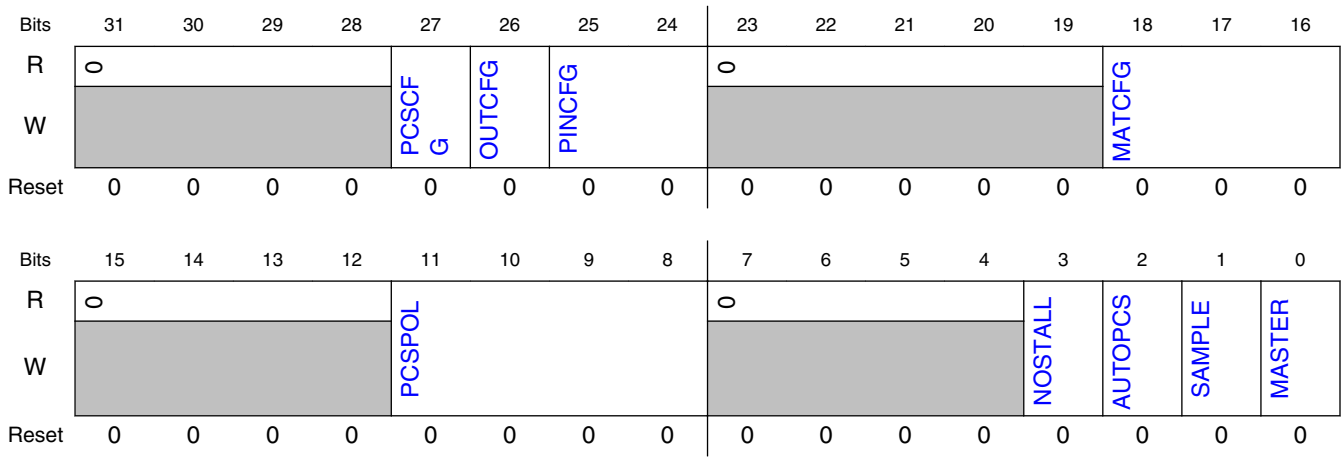
### 57.3.2.9.1 Offset

| Register | Offset |
|---|---|
| CFGR1 | 24h |

### 57.3.2.9.2 Function

The CFGR1 should only be written when the LPSPI is disabled.

### 57.3.2.9.3 Diagram




### 57.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>PCSCFG | Peripheral Chip Select Configuration<br>If performing 4-bit transfers, the Peripheral Chip Select Configuration bit must be set.<br>0b - PCS[3:2] are enabled<br>1b - PCS[3:2] are disabled |
| 26<br>OUTCFG | Output Config<br>Configures if the output data is tristated between accesses (LPSPI_PCS is negated).<br>0b - Output data retains last value when chip select is negated<br>1b - Output data is tristated when chip select is negated |
| 25-24<br>PINCFG | Pin Configuration<br>Configures which pins are used for input and output data during single bit transfers.<br>00b - SIN is used for input data and SOUT is used for output data<br>01b - SIN is used for both input and output data<br>10b - SOUT is used for both input and output data<br>11b - SOUT is used for input data and SIN is used for output data |
| 23-19<br>— | Reserved |
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that will cause the DMF to set.<br>**NOTE:** *Syntax:* * is boolean AND, + is boolean OR<br>000b - Match is disabled<br>001b - Reserved<br>010b - 010b - Match is enabled, if 1st data word equals MATCH0 OR MATCH1, i.e., (1st data word = MATCH0 + MATCH1) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 011b - 011b - Match is enabled, if any data word equals MATCH0 OR MATCH1, i.e., (any data word = MATCH0 + MATCH1)<br>100b - 100b - Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, i.e., [(1st data word = MATCH0) * (2nd data word = MATCH1)]<br>101b - 101b - Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, i.e., [(any data word = MATCH0) * (next data word = MATCH1)]<br>110b - 110b - Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(1st data word * MATCH1) = (MATCH0 * MATCH1)]<br>111b - 111b - Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(any data word * MATCH1) = (MATCH0 * MATCH1)] |
| 15-12<br>— | Reserved |
| 11-8<br>PCSPOL | Peripheral Chip Select Polarity<br>Configures the polarity of each Peripheral Chip Select pin.<br>0000b - The Peripheral Chip Select pin PCSx is active low<br>0001b - The Peripheral Chip Select pin PCSx is active high |
| 7-4<br>— | Reserved |
| 3<br>NOSTALL | No Stall<br>In master mode, the LPSPI will stall transfers when the transmit FIFO is empty or when the receive FIFO is full, ensuring that no transmit FIFO underrun or receive FIFO overrun can occur. Setting the No Stall bit will disable this functionality.<br>0b - Transfers will stall when the transmit FIFO is empty or the receive FIFO is full<br>1b - Transfers will not stall, allowing transmit FIFO underruns or receive FIFO overruns to occur |
| 2<br>AUTOPCS | Automatic PCS<br>For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the Automatic PCS bit will cause the LPSPI to generate an internal PCS signal at the end of each transfer word when the Clock Phase bit TCR[CPHA] = 1.<br>• When the Automatic PCS bit is set, the SCK must remain idle for at least 4 LPSPI functional clock cycles (divided by the Prescaler Value TCR[PRESCALE] configuration) between each word, to ensure correct operations<br>• In master mode, the Automatic PCS bit is ignored<br>0b - Automatic PCS generation is disabled<br>1b - Automatic PCS generation is enabled |
| 1<br>SAMPLE | Sample Point<br>When set, the LPSPI master will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data.<br>• The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode<br>• In slave mode, the SAMPLE bit is ignored<br>0b - Input data is sampled on SCK edge<br>1b - Input data is sampled on delayed SCK edge |
| 0<br>MASTER | Master Mode<br>Configures the LPSPI in master or slave mode. The Master Mode bit directly controls the direction of the LPSPI_SCK and LPCPI_PCS pins.<br>0b - Slave mode<br>1b - Master mode |

## 57.3.2.10 Data Match Register 0 (DMR0)

### 57.3.2.10.1 Offset

| Register | Offset |
|---|---|
| DMR0 | 30h |

### 57.3.2.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.2.10.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH0 | Match 0 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 0 Value is compared against the received data. |

## 57.3.2.11 Data Match Register 1 (DMR1)

### 57.3.2.11.1 Offset

| Register | Offset |
|---|---|
| DMR1 | 34h |

### 57.3.2.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.2.11.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH1 | Match 1 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 1 Value is compared against the received data. |

## 57.3.2.12 Clock Configuration Register (CCR)

### 57.3.2.12.1 Offset

| Register | Offset |
|---|---|
| CCR | 40h |

### 57.3.2.12.2 Function

The Clock Configuration Register is only used in master mode, and the Clock Configuration Register cannot be changed when the LPSPI is enabled.

### 57.3.2.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBT | | | | | | | | SCKDIV | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.2.12.4 Fields

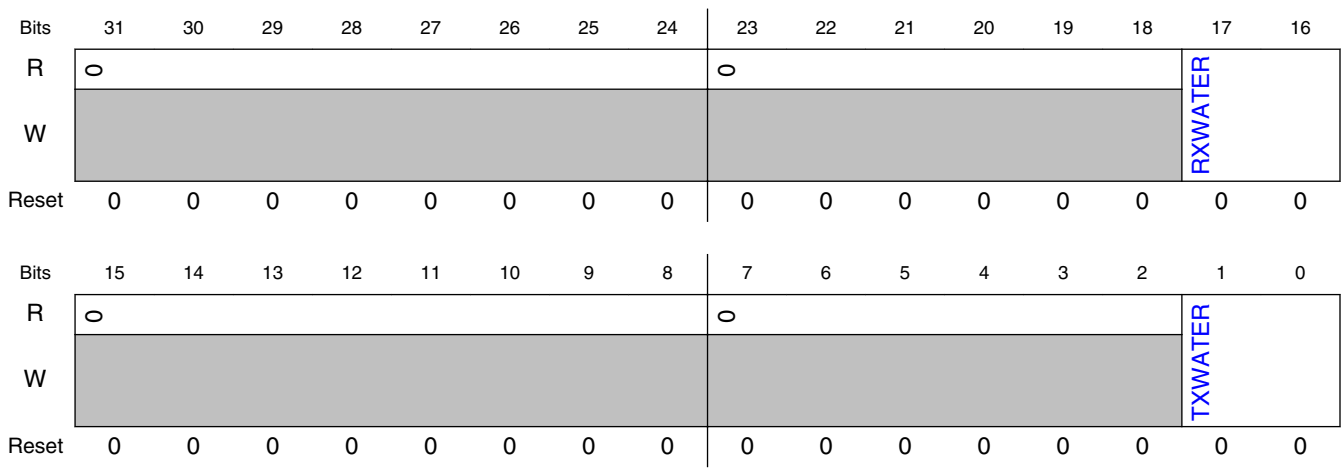
| Field | Function |
|---|---|
| 31-24<br>SCKPCS | SCK-to-PCS Delay<br>In master mode: configures the delay from the last SCK edge to the PCS negation.<br>• The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 23-16<br>PCSSCK | PCS-to-SCK Delay<br>In master mode: configures the delay from the PCS assertion to the first SCK edge.<br>• The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 15-8<br>DBT | Delay Between Transfers<br>In master mode:<br>• Configures the delay from the PCS negation to the next PCS assertion.<br>  • The delay is equal to (DBT + 2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 2 cycles.<br>  • Half of the delay occurs before PCS assertion and the other half of the delay occurs after PCS negation. If the command word is updated between 2 transfers, then the command word is updated half-way between the PCS negation of the last transfer and PCS assertion of the next transfer.<br>  • The command word sets which PCS signal is used (of PCS[3:0]}, the polarity/phase of the SCK signal, and the Prescaler Value.<br>• Configures the delay from the last SCK edge of a transfer word and the first SCK edge of the next transfer word, in a continuous transfer.<br>  • The delay is equal to (DBT + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 1 cycle. |
| 7-0<br>SCKDIV | SCK Divider<br>In master mode, the SCK Divider configures the divide ratio of the SCK pin.<br>• The SCK period is equal to (SCKDIV+2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum SCK period is 2 cycles.<br>• If the SCK period is an odd number of cycles, then the 1st half of the SCK period will be 1 cycle longer than the 2nd half of the SCK period. |

# 57.3.2.13 FIFO Control Register (FCR)

## 57.3.2.13.1 Offset

| Register | Offset |
|---|---|
| FCR | 58h |

## 57.3.2.13.2 Diagram




## 57.3.2.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal or greater than the FIFO size will be truncated. |
| 15-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1-0 | Transmit FIFO Watermark |

| Field | Function |
|---|---|
| TXWATER | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal or greater than the FIFO size will be truncated. |

## 57.3.2.14 FIFO Status Register (FSR)

### 57.3.2.14.1 Offset

| Register | Offset |
|---|---|
| FSR | 5Ch |

### 57.3.2.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.2.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words currently stored in the receive FIFO. |
| 15-8<br>— | Reserved |
| 7-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words currently stored in the transmit FIFO. |

## 57.3.2.15 Transmit Command Register (TCR)

### 57.3.2.15.1 Offset

| Register | Offset |
|---|---|
| TCR | 60h |

### 57.3.2.15.2 Function

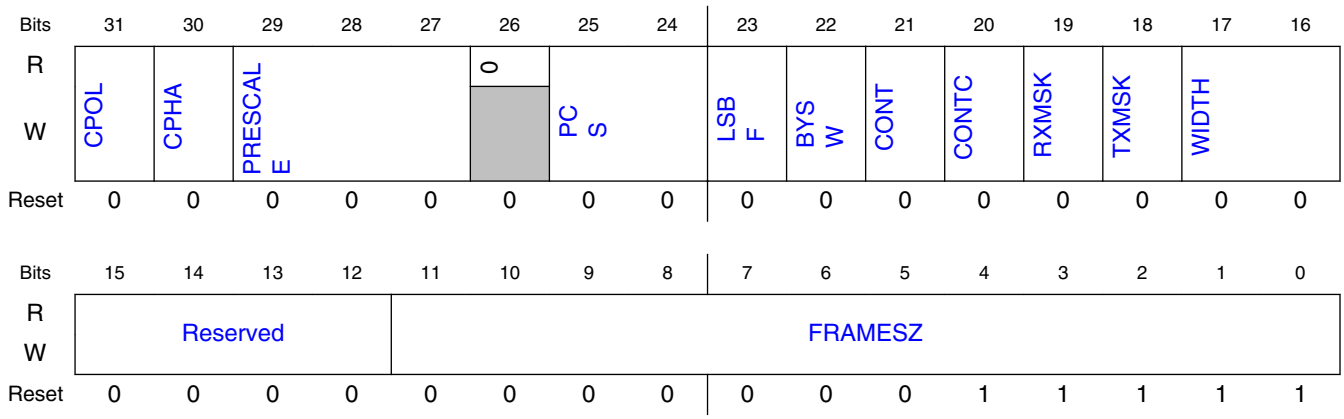
Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that the data are written. Command Register writes will be tagged and cause the command register to update, after that entry reaches the top of the FIFO. This allows changes to the command word and the transmit data itself to be interleaved. Changing the command word will cause all subsequent SPI bus transfers to be performed using the new command word.

- **In master mode,** writing a new command word does not initiate a new transfer, unless TXMSK is set. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK set). Hardware will clear TXMSK when the LPSPI_PCS negates.
- **In master mode,** if the command word is changed before an existing frame has completed, then the existing frame will terminate and the command word will then update. The command word can be changed during a continuous transfer, if CONTC of the new command word is set and the command word is written on a frame size boundary.
- **In slave mode,** the command word should be changed only when the LPSPI is idle and there is no SPI bus transfer.

**Avoid register reading problems:** Reading the Transmit Command Register will return the current state of the command register. Reading the Transmit Command Register at the same time that the Transmit Command Register is loaded from the transmit FIFO, can return an incorrect Transmit Command Register value. It is recommended:

- to either read the Transmit Command Register when the transmit FIFO is empty,
- or to read the Transmit Command Register more than once and then compare the returned values.

### 57.3.2.15.3 Diagram




### 57.3.2.15.4 Fields

| Field | Function |
|---|---|
| 31<br>CPOL | Clock Polarity<br>The Clock Polarity field is only updated between frames.<br>0b - The inactive state value of SCK is low<br>1b - The inactive state value of SCK is high |
| 30<br>CPHA | Clock Phase<br>The Clock Phase field is only updated between frames.<br>0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK<br>1b - Data is changed on the leading edge of SCK and captured on the following edge of SCK |
| 29-27<br>PRESCALE | Prescaler Value<br>For all SPI bus transfers, the Prescaler value applied to the clock configuration register. The Prescaler Value field is only updated between frames.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |
| 26<br>— | Reserved |
| 25-24<br>PCS | Peripheral Chip Select<br>Configures the peripheral chip select used for the transfer. The Peripheral Chip Select field is only updated between frames.<br>00b - Transfer using LPSPI_PCS[0]<br>01b - Transfer using LPSPI_PCS[1]<br>10b - Transfer using LPSPI_PCS[2]<br>11b - Transfer using LPSPI_PCS[3] |
| 23 | LSB First |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LSBF | 0b - Data is transferred MSB first<br>1b - Data is transferred LSB first |
| 22<br>BYSW | Byte Swap<br>Byte swap will swap the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers).<br>0b - Byte swap is disabled<br>1b - Byte swap is enabled |
| 21<br>CONT | Continuous Transfer<br>• In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.<br>• In slave mode, when continuous transfer is enabled, the LPSPI will only transmit the first FRAMESZ bits; after which the LPSPI will transmit received data (assuming a 32-bit shift register).<br><br>0b - Continuous transfer is disabled<br>1b - Continuous transfer is enabled |
| 20<br>CONTC | Continuing Command<br>In master mode, the Continuing Command bit allows the command word to be changed within a continuous transfer.<br>• The initial command word must enable continuous transfer (CONT=1),<br>• the continuing command must set this bit (CONTC=1),<br>• and the continuing command word must be loaded on a frame size boundary.<br><br>For example, if the continuous transfer has a frame size of 64-bits, then a continuing command word must be loaded on a 64-bit boundary.<br><br>0b - Command word for start of new transfer<br>1b - Command word for continuing transfer |
| 19<br>RXMSK | Receive Data Mask<br>When set, receive data is masked (receive data is not stored in receive FIFO).<br>0b - Normal transfer<br>1b - Receive data is masked |
| 18<br>TXMSK | Transmit Data Mask<br>When set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer.<br>0b - Normal transfer<br>1b - Mask transmit data |
| 17-16<br>WIDTH | Transfer Width<br>For 2-bit or 4-bit transfers, either Receive Data Mask (RXMSK) or Transmit Data Mask (TXMSK) must be set.<br>00b - 1 bit transfer<br>01b - 2 bit transfer<br>10b - 4 bit transfer<br>11b - Reserved |
| 15-12<br>— | Reserved. Software should only write zero to this field. |
| 11-0<br>FRAMESZ | Frame Size<br>Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. |

| Field | Function |
|---|---|
| | • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32 bits, and the 3rd word is 8 bits. |

## 57.3.2.16 Transmit Data Register (TDR)

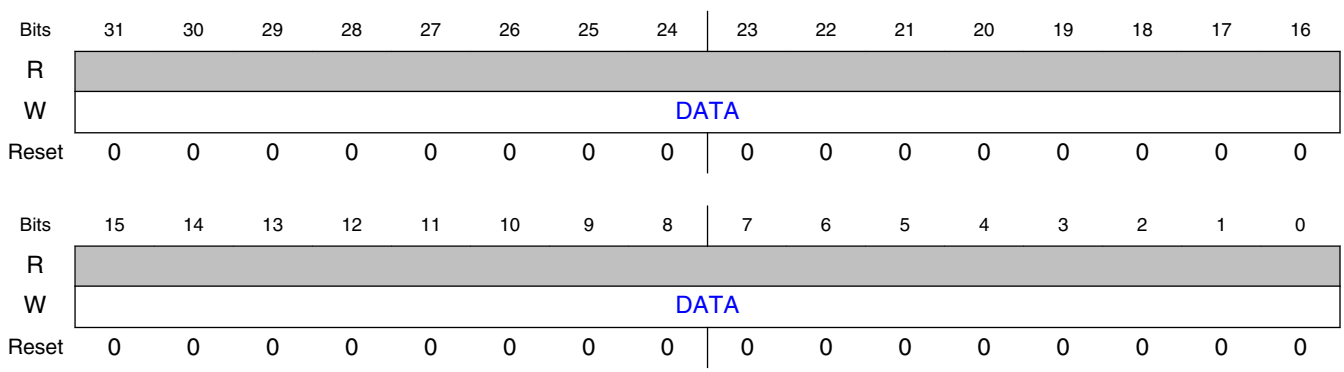
### 57.3.2.16.1 Offset

| Register | Offset |
|---|---|
| TDR | 64h |

### 57.3.2.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that it (the data) was written.

### 57.3.2.16.3 Diagram




### 57.3.2.16.4 Fields

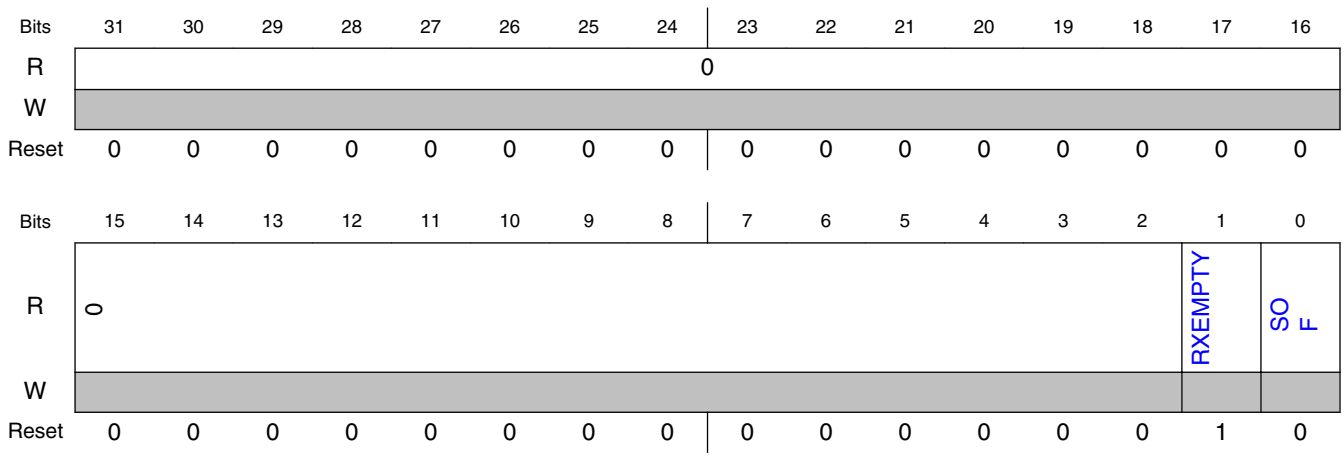
| Field | Function |
|---|---|
| 31-0<br>DATA | Transmit Data<br>Both 8-bit and 16-bit writes of transmit data will zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes. |

## 57.3.2.17 Receive Status Register (RSR)

### 57.3.2.17.1 Offset

| Register | Offset |
|---|---|
| RSR | 70h |

### 57.3.2.17.2 Diagram




### 57.3.2.17.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RXEMPTY | RX FIFO Empty<br>0b - RX FIFO is not empty<br>1b - RX FIFO is empty |
| 0<br>SOF | Start Of Frame<br>Indicates that this is the first data word received after LPSPI_PCS assertion.<br>0b - Subsequent data word received after LPSPI_PCS assertion<br>1b - First data word received after LPSPI_PCS assertion |

## 57.3.2.18 Receive Data Register (RDR)

#### 57.3.2.18.1 Offset

| Register | Offset |
|---|---|
| RDR | 74h |

#### 57.3.2.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 57.3.2.18.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

## 57.3.3 LPSPI2 register descriptions

### 57.3.3.1 LPSPI2 Memory map

LPSPI2 base address: 4004_1000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0004h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Control Register (CR) | 32 | RW | 0000_0000h |
| 14h | Status Register (SR) | 32 | W1C | 0000_0001h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 18h | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable Register (DER) | 32 | RW | 0000_0000h |
| 20h | Configuration Register 0 (CFGR0) | 32 | RW | 0000_0000h |
| 24h | Configuration Register 1 (CFGR1) | 32 | RW | 0000_0000h |
| 30h | Data Match Register 0 (DMR0) | 32 | RW | 0000_0000h |
| 34h | Data Match Register 1 (DMR1) | 32 | RW | 0000_0000h |
| 40h | Clock Configuration Register (CCR) | 32 | RW | 0000_0000h |
| 58h | FIFO Control Register (FCR) | 32 | RW | 0000_0000h |
| 5Ch | FIFO Status Register (FSR) | 32 | RO | 0000_0000h |
| 60h | Transmit Command Register (TCR) | 32 | RW | 0000_001Fh |
| 64h | Transmit Data Register (TDR) | 32 | WO | 0000_0000h |
| 70h | Receive Status Register (RSR) | 32 | RO | 0000_0002h |
| 74h | Receive Data Register (RDR) | 32 | RO | 0000_0000h |

## 57.3.3.2 Version ID Register (VERID)

### 57.3.3.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 57.3.3.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 57.3.3.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module specification. Read-only field. |
| 15-0<br>FEATURE | Module Identification Number<br>Returns the feature set number. Read-only field.<br>0000000000000100b - Standard feature set supporting a 32-bit shift register. |

### 57.3.3.3 Parameter Register (PARAM)

#### 57.3.3.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 57.3.3.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 57.3.3.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Sets the maximum number of words in the receive FIFO, which is $2^{RXFIFO}$ |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Sets the maximum number of words in the transmit FIFO, which is $2^{TXFIFO}$ |

## 57.3.3.4 Control Register (CR)

### 57.3.3.4.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 57.3.3.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | DBGEN | DOZEN | RST | MEN |
| W | | | | | | | RRF | RTF | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.3.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Module is disabled in debug mode<br>1b - Module is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the module in Doze mode<br>0b - Module is enabled in Doze mode<br>1b - Module is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |
| 0<br>MEN | Module Enable<br>0b - Module is disabled<br>1b - Module is enabled |

## 57.3.3.5 Status Register (SR)

### 57.3.3.5.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 57.3.3.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMF | REF | TEF | TCF | FCF | WCF | 0 | | | | | | RDF | TDF |
| W | | | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 57.3.3.5.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MBF | Module Busy Flag<br>0b - LPSPI is idle<br>1b - LPSPI is busy |
| 23-14<br>— | Reserved |
| 13<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by CFGR1[MATCFG], Configuration Register 1).<br>0b - Have not received matching data<br>1b - Have received matching data |
| 12<br>REF | Receive Error Flag<br>The Receive Error Flag will set when the Receiver FIFO overflows. When the Receive Error Flag is set, it is recommended to first end the transfer, empty the Receive FIFO, clear the Receive Error Flag and then restart the transfer from the beginning.<br>0b - Receive FIFO has not overflowed<br>1b - Receive FIFO has overflowed |
| 11<br>TEF | Transmit Error Flag<br>The Transmit Error Flag will set when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning.<br>0b - Transmit FIFO underrun has not occurred<br>1b - Transmit FIFO underrun has occurred |
| 10<br>TCF | Transfer Complete Flag<br>In master mode when the LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag will set.<br>0b - All transfers have not completed<br>1b - All transfers have completed |
| 9<br>FCF | Frame Complete Flag<br>The Frame Complete Flag will set at the end of each frame transfer, when the PCS negates.<br>0b - Frame transfer has not completed<br>1b - Frame transfer has completed |
| 8<br>WCF | Word Complete Flag<br>The Word Complete Flag will set when the last bit of a received word is sampled.<br>0b - Transfer of a received word has not yet completed<br>1b - Transfer of a received word has completed |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than FCR[RXWATER] (FIFO Control Register)<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0 | Transmit Data Flag |

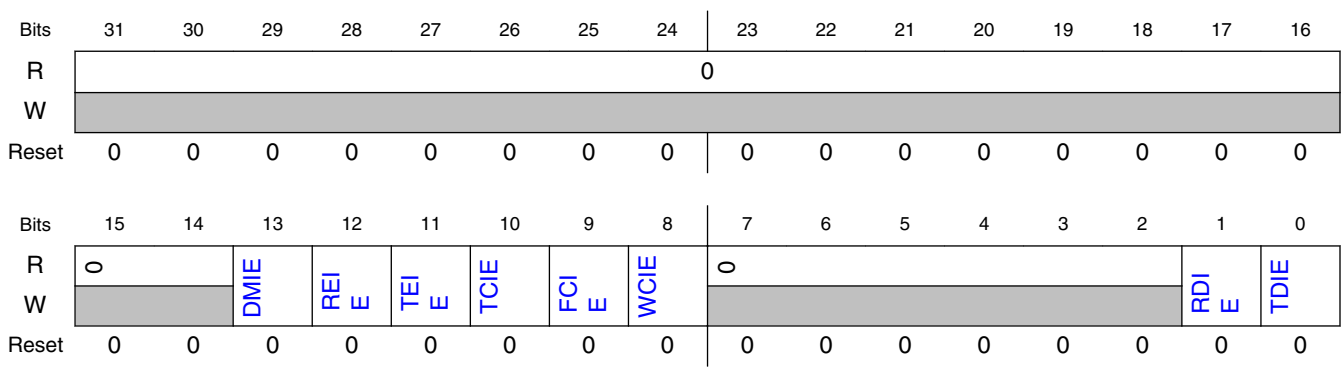
| Field | Function |
|---|---|
| TDF | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than FCR[TXWATER] (FIFO Control Register)<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 57.3.3.6 Interrupt Enable Register (IER)

### 57.3.3.6.1 Offset

| Register | Offset |
|---|---|
| IER | 18h |

### 57.3.3.6.2 Diagram




### 57.3.3.6.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13<br>DMIE | Data Match Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>REIE | Receive Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 11<br>TEIE | Transmit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

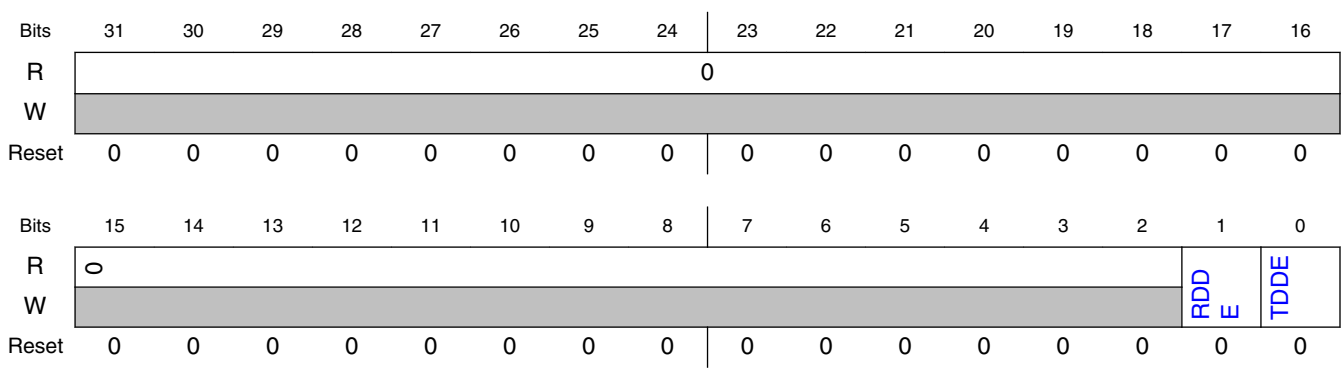
| Field | Function |
|---|---|
| 10<br>TCIE | Transfer Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>FCIE | Frame Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>WCIE | Word Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 57.3.3.7 DMA Enable Register (DER)

### 57.3.3.7.1 Offset

| Register | Offset |
|---|---|
| DER | 1Ch |

### 57.3.3.7.2 Diagram



#### 57.3.3.7.3 Fields

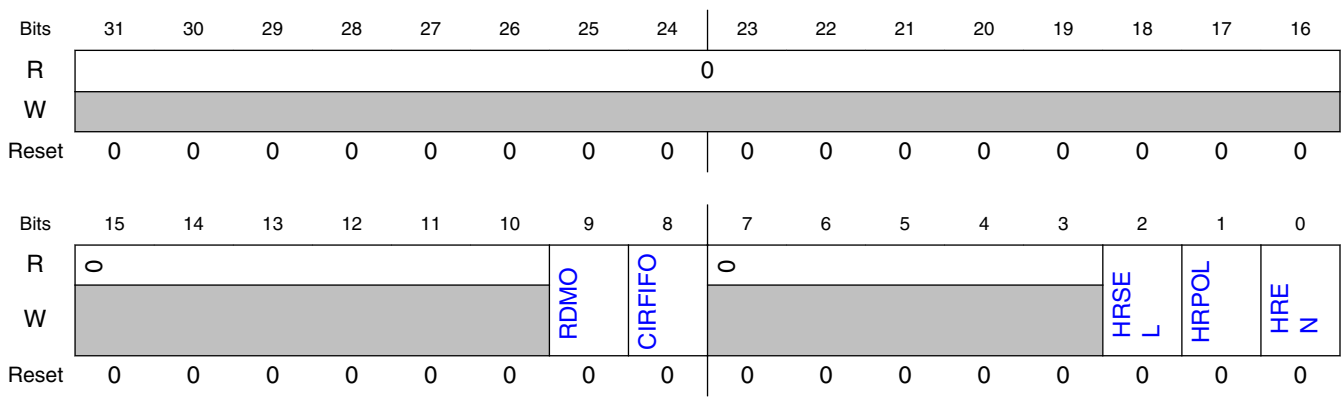
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

### 57.3.3.8 Configuration Register 0 (CFGR0)

#### 57.3.3.8.1 Offset

| Register | Offset |
|---|---|
| CFGR0 | 20h |

#### 57.3.3.8.2 Diagram




#### 57.3.3.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9 | Receive Data Match Only |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RDMO | When Receive Data Match Only is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded.<br>• Receive Data Match Only bit should be set when the LPSPI is idle and the Data Match Flag is clear<br>• After the Data Match Flag (DMF) is set, the Receive Data Match Only bit configuration is ignored<br>• When disabling RDMO and to ensure that no receive data is lost, before clearing the Data Match Flag, first clear Receive Data Match Only (RDMO)<br><br>0b - Received data is stored in the receive FIFO as in normal operations<br>1b - Received data is discarded unless the Data Match Flag (DMF) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as it normally is, but after the LPSPI is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This restoring of the read pointer will cause the contents of the transmit FIFO to be cycled through repeatedly.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When the host request function is enabled with the LPSPI_HREQ pin, the LPSPI_PCS[1] function is disabled.<br>0b - Host request input is the LPSPI_HREQ pin<br>1b - Host request input is the input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request pin.<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable<br>When enabled in master mode, the LPSPI will only start a new SPI bus transfer if the host request input is asserted. When the LPSPI is busy, the host request input is ignored.<br>0b - Host request is disabled<br>1b - Host request is enabled |

## 57.3.3.9 Configuration Register 1 (CFGR1)

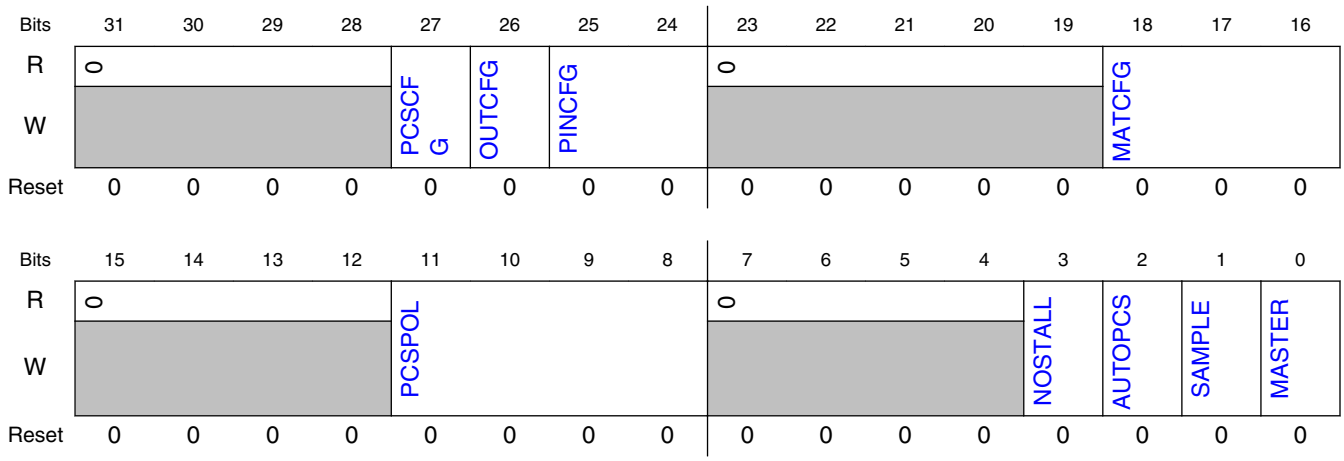
### 57.3.3.9.1 Offset

| Register | Offset |
|---|---|
| CFGR1 | 24h |

### 57.3.3.9.2 Function

The CFGR1 should only be written when the LPSPI is disabled.

## 57.3.3.9.3 Diagram




## 57.3.3.9.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>PCSCFG | Peripheral Chip Select Configuration<br>If performing 4-bit transfers, the Peripheral Chip Select Configuration bit must be set.<br>0b - PCS[3:2] are enabled<br>1b - PCS[3:2] are disabled |
| 26<br>OUTCFG | Output Config<br>Configures if the output data is tristated between accesses (LPSPI_PCS is negated).<br>0b - Output data retains last value when chip select is negated<br>1b - Output data is tristated when chip select is negated |
| 25-24<br>PINCFG | Pin Configuration<br>Configures which pins are used for input and output data during single bit transfers.<br>00b - SIN is used for input data and SOUT is used for output data<br>01b - SIN is used for both input and output data<br>10b - SOUT is used for both input and output data<br>11b - SOUT is used for input data and SIN is used for output data |
| 23-19<br>— | Reserved |
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that will cause the DMF to set.<br>**NOTE:** *Syntax:* * is boolean AND, + is boolean OR<br>000b - Match is disabled<br>001b - Reserved<br>010b - 010b - Match is enabled, if 1st data word equals MATCH0 OR MATCH1, i.e., (1st data word = MATCH0 + MATCH1) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 011b - 011b - Match is enabled, if any data word equals MATCH0 OR MATCH1, i.e., (any data word = MATCH0 + MATCH1)<br>100b - 100b - Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, i.e., [(1st data word = MATCH0) * (2nd data word = MATCH1)]<br>101b - 101b - Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, i.e., [(any data word = MATCH0) * (next data word = MATCH1)]<br>110b - 110b - Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(1st data word * MATCH1) = (MATCH0 * MATCH1)]<br>111b - 111b - Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(any data word * MATCH1) = (MATCH0 * MATCH1)] |
| 15-12<br>— | Reserved |
| 11-8<br>PCSPOL | Peripheral Chip Select Polarity<br>Configures the polarity of each Peripheral Chip Select pin.<br>0000b - The Peripheral Chip Select pin PCSx is active low<br>0001b - The Peripheral Chip Select pin PCSx is active high |
| 7-4<br>— | Reserved |
| 3<br>NOSTALL | No Stall<br>In master mode, the LPSPI will stall transfers when the transmit FIFO is empty or when the receive FIFO is full, ensuring that no transmit FIFO underrun or receive FIFO overrun can occur. Setting the No Stall bit will disable this functionality.<br>0b - Transfers will stall when the transmit FIFO is empty or the receive FIFO is full<br>1b - Transfers will not stall, allowing transmit FIFO underruns or receive FIFO overruns to occur |
| 2<br>AUTOPCS | Automatic PCS<br>For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the Automatic PCS bit will cause the LPSPI to generate an internal PCS signal at the end of each transfer word when the Clock Phase bit TCR[CPHA] = 1.<br>• When the Automatic PCS bit is set, the SCK must remain idle for at least 4 LPSPI functional clock cycles (divided by the Prescaler Value TCR[PRESCALE] configuration) between each word, to ensure correct operations<br>• In master mode, the Automatic PCS bit is ignored<br>0b - Automatic PCS generation is disabled<br>1b - Automatic PCS generation is enabled |
| 1<br>SAMPLE | Sample Point<br>When set, the LPSPI master will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data.<br>• The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode<br>• In slave mode, the SAMPLE bit is ignored<br>0b - Input data is sampled on SCK edge<br>1b - Input data is sampled on delayed SCK edge |
| 0<br>MASTER | Master Mode<br>Configures the LPSPI in master or slave mode. The Master Mode bit directly controls the direction of the LPSPI_SCK and LPCPI_PCS pins.<br>0b - Slave mode<br>1b - Master mode |

## 57.3.3.10 Data Match Register 0 (DMR0)

### 57.3.3.10.1 Offset

| Register | Offset |
|---|---|
| DMR0 | 30h |

### 57.3.3.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.3.10.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH0 | Match 0 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 0 Value is compared against the received data. |

## 57.3.3.11 Data Match Register 1 (DMR1)

### 57.3.3.11.1 Offset

| Register | Offset |
|---|---|
| DMR1 | 34h |

### 57.3.3.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.3.11.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH1 | Match 1 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 1 Value is compared against the received data. |

## 57.3.3.12 Clock Configuration Register (CCR)

### 57.3.3.12.1 Offset

| Register | Offset |
|---|---|
| CCR | 40h |

### 57.3.3.12.2 Function

The Clock Configuration Register is only used in master mode, and the Clock Configuration Register cannot be changed when the LPSPI is enabled.

## 57.3.3.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBT | | | | | | | | SCKDIV | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 57.3.3.12.4 Fields

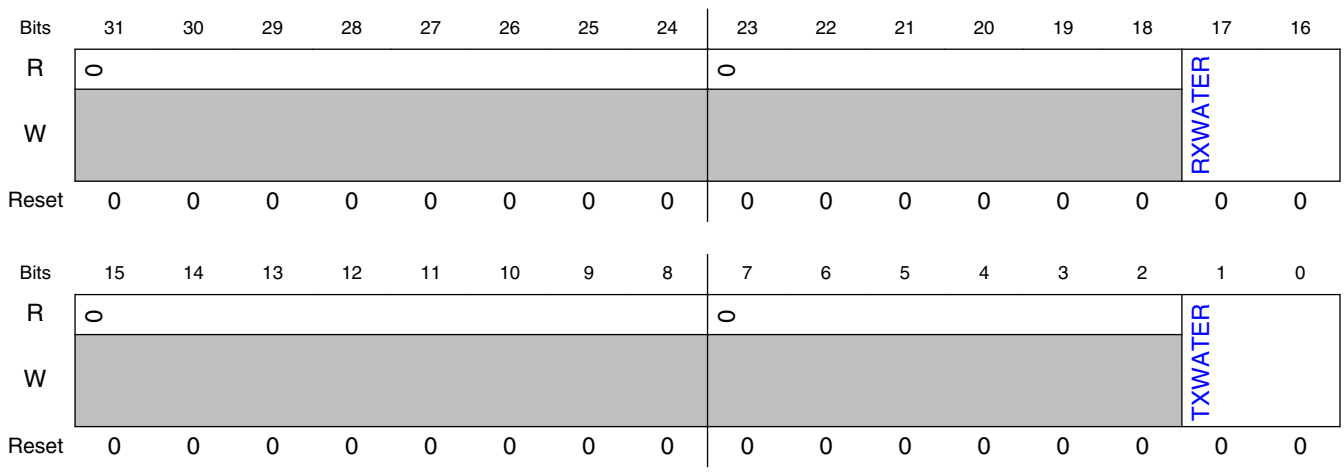
| Field | Function |
|---|---|
| 31-24<br>SCKPCS | SCK-to-PCS Delay<br>In master mode: configures the delay from the last SCK edge to the PCS negation.<br>• The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 23-16<br>PCSSCK | PCS-to-SCK Delay<br>In master mode: configures the delay from the PCS assertion to the first SCK edge.<br>• The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 15-8<br>DBT | Delay Between Transfers<br>In master mode:<br>• Configures the delay from the PCS negation to the next PCS assertion.<br>  • The delay is equal to (DBT + 2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 2 cycles.<br>  • Half of the delay occurs before PCS assertion and the other half of the delay occurs after PCS negation. If the command word is updated between 2 transfers, then the command word is updated half-way between the PCS negation of the last transfer and PCS assertion of the next transfer.<br>  • The command word sets which PCS signal is used (of PCS[3:0]}, the polarity/phase of the SCK signal, and the Prescaler Value.<br>• Configures the delay from the last SCK edge of a transfer word and the first SCK edge of the next transfer word, in a continuous transfer.<br>  • The delay is equal to (DBT + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 1 cycle. |
| 7-0<br>SCKDIV | SCK Divider<br>In master mode, the SCK Divider configures the divide ratio of the SCK pin.<br>• The SCK period is equal to (SCKDIV+2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum SCK period is 2 cycles.<br>• If the SCK period is an odd number of cycles, then the 1st half of the SCK period will be 1 cycle longer than the 2nd half of the SCK period. |

# 57.3.3.13 FIFO Control Register (FCR)

## 57.3.3.13.1 Offset

| Register | Offset |
|---|---|
| FCR | 58h |

## 57.3.3.13.2 Diagram




## 57.3.3.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal or greater than the FIFO size will be truncated. |
| 15-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1-0 | Transmit FIFO Watermark |

| Field | Function |
|---|---|
| TXWATER | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal or greater than the FIFO size will be truncated. |

## 57.3.3.14 FIFO Status Register (FSR)

### 57.3.3.14.1 Offset

| Register | Offset |
|---|---|
| FSR | 5Ch |

### 57.3.3.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.3.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words currently stored in the receive FIFO. |
| 15-8<br>— | Reserved |
| 7-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words currently stored in the transmit FIFO. |

## 57.3.3.15 Transmit Command Register (TCR)

### 57.3.3.15.1 Offset

| Register | Offset |
|---|---|
| TCR | 60h |

### 57.3.3.15.2 Function

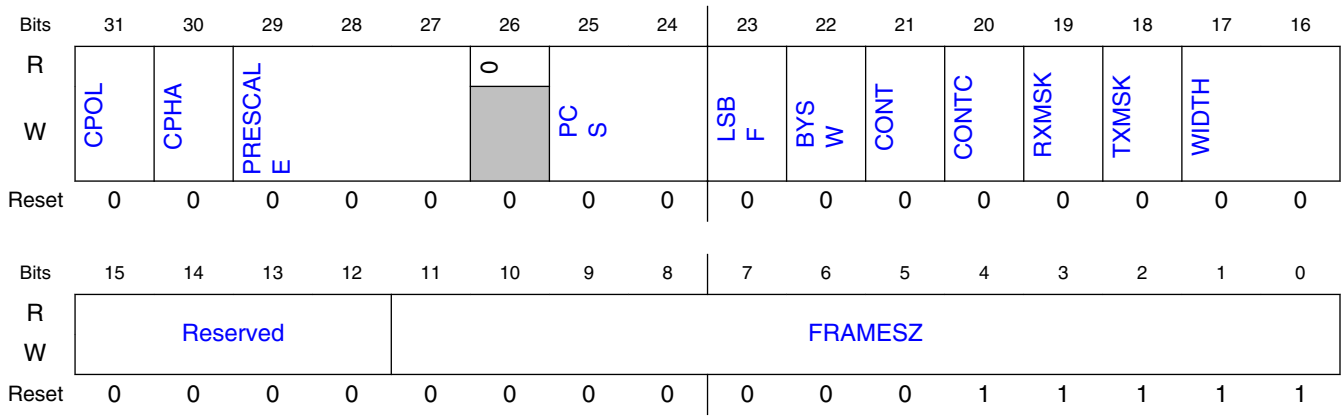
Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that the data are written. Command Register writes will be tagged and cause the command register to update, after that entry reaches the top of the FIFO. This allows changes to the command word and the transmit data itself to be interleaved. Changing the command word will cause all subsequent SPI bus transfers to be performed using the new command word.

- **In master mode,** writing a new command word does not initiate a new transfer, unless TXMSK is set. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK set). Hardware will clear TXMSK when the LPSPI_PCS negates.
- **In master mode,** if the command word is changed before an existing frame has completed, then the existing frame will terminate and the command word will then update. The command word can be changed during a continuous transfer, if CONTC of the new command word is set and the command word is written on a frame size boundary.
- **In slave mode,** the command word should be changed only when the LPSPI is idle and there is no SPI bus transfer.

**Avoid register reading problems:** Reading the Transmit Command Register will return the current state of the command register. Reading the Transmit Command Register at the same time that the Transmit Command Register is loaded from the transmit FIFO, can return an incorrect Transmit Command Register value. It is recommended:

- to either read the Transmit Command Register when the transmit FIFO is empty,
- or to read the Transmit Command Register more than once and then compare the returned values.

## 57.3.3.15.3 Diagram




## 57.3.3.15.4 Fields

| Field | Function |
|---|---|
| 31<br>CPOL | Clock Polarity<br>The Clock Polarity field is only updated between frames.<br>0b - The inactive state value of SCK is low<br>1b - The inactive state value of SCK is high |
| 30<br>CPHA | Clock Phase<br>The Clock Phase field is only updated between frames.<br>0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK<br>1b - Data is changed on the leading edge of SCK and captured on the following edge of SCK |
| 29-27<br>PRESCALE | Prescaler Value<br>For all SPI bus transfers, the Prescaler value applied to the clock configuration register. The Prescaler Value field is only updated between frames.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |
| 26<br>— | Reserved |
| 25-24<br>PCS | Peripheral Chip Select<br>Configures the peripheral chip select used for the transfer. The Peripheral Chip Select field is only updated between frames.<br>00b - Transfer using LPSPI_PCS[0]<br>01b - Transfer using LPSPI_PCS[1]<br>10b - Transfer using LPSPI_PCS[2]<br>11b - Transfer using LPSPI_PCS[3] |
| 23 | LSB First |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LSBF | 0b - Data is transferred MSB first<br>1b - Data is transferred LSB first |
| 22<br>BYSW | Byte Swap<br>Byte swap will swap the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers).<br>0b - Byte swap is disabled<br>1b - Byte swap is enabled |
| 21<br>CONT | Continuous Transfer<br>• In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.<br>• In slave mode, when continuous transfer is enabled, the LPSPI will only transmit the first FRAMESZ bits; after which the LPSPI will transmit received data (assuming a 32-bit shift register).<br><br>0b - Continuous transfer is disabled<br>1b - Continuous transfer is enabled |
| 20<br>CONTC | Continuing Command<br>In master mode, the Continuing Command bit allows the command word to be changed within a continuous transfer.<br>• The initial command word must enable continuous transfer (CONT=1),<br>• the continuing command must set this bit (CONTC=1),<br>• and the continuing command word must be loaded on a frame size boundary.<br><br>For example, if the continuous transfer has a frame size of 64-bits, then a continuing command word must be loaded on a 64-bit boundary.<br>0b - Command word for start of new transfer<br>1b - Command word for continuing transfer |
| 19<br>RXMSK | Receive Data Mask<br>When set, receive data is masked (receive data is not stored in receive FIFO).<br>0b - Normal transfer<br>1b - Receive data is masked |
| 18<br>TXMSK | Transmit Data Mask<br>When set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer.<br>0b - Normal transfer<br>1b - Mask transmit data |
| 17-16<br>WIDTH | Transfer Width<br>For 2-bit or 4-bit transfers, either Receive Data Mask (RXMSK) or Transmit Data Mask (TXMSK) must be set.<br>00b - 1 bit transfer<br>01b - 2 bit transfer<br>10b - 4 bit transfer<br>11b - Reserved |
| 15-12<br>— | Reserved. Software should only write zero to this field. |
| 11-0<br>FRAMESZ | Frame Size<br>Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. |

| Field | Function |
|---|---|
| | • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32 bits, and the 3rd word is 8 bits. |

## 57.3.3.16 Transmit Data Register (TDR)

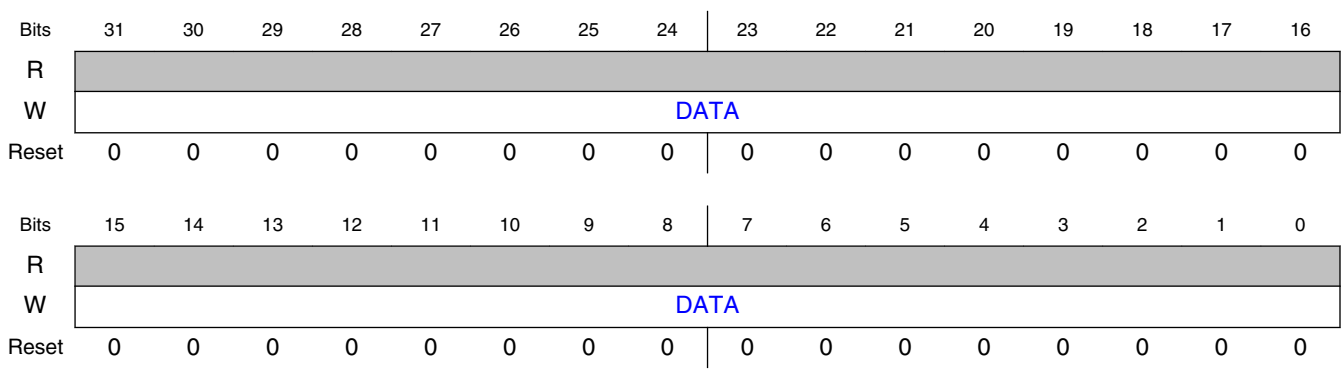
### 57.3.3.16.1 Offset

| Register | Offset |
|---|---|
| TDR | 64h |

### 57.3.3.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that it (the data) was written.

### 57.3.3.16.3 Diagram




### 57.3.3.16.4 Fields

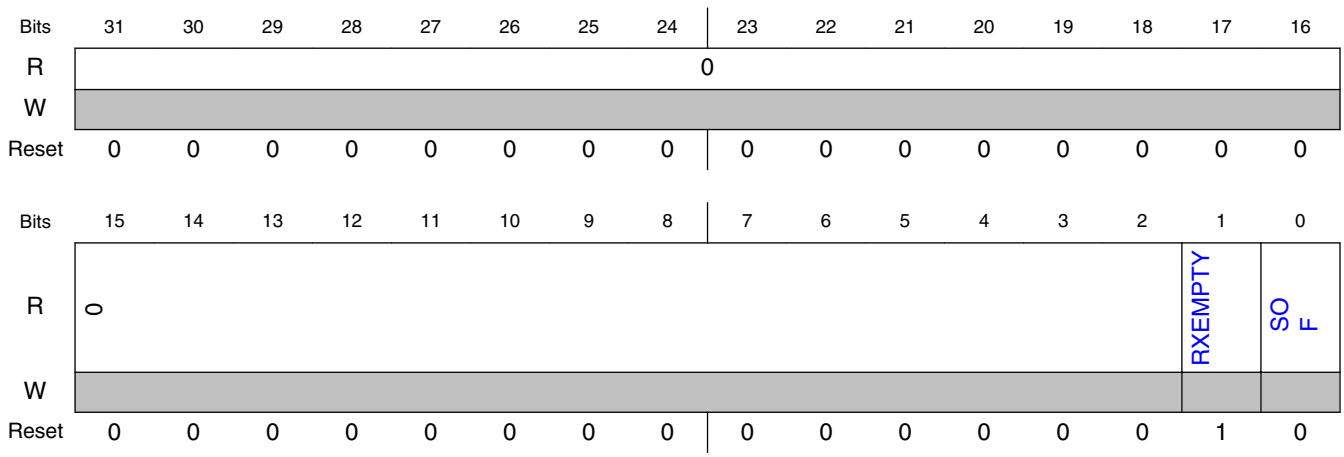
| Field | Function |
|---|---|
| 31-0<br>DATA | Transmit Data<br>Both 8-bit and 16-bit writes of transmit data will zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes. |

# 57.3.3.17 Receive Status Register (RSR)

## 57.3.3.17.1 Offset

| Register | Offset |
|---|---|
| RSR | 70h |

## 57.3.3.17.2 Diagram




## 57.3.3.17.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RXEMPTY | RX FIFO Empty<br>0b - RX FIFO is not empty<br>1b - RX FIFO is empty |
| 0<br>SOF | Start Of Frame<br>Indicates that this is the first data word received after LPSPI_PCS assertion.<br>0b - Subsequent data word received after LPSPI_PCS assertion<br>1b - First data word received after LPSPI_PCS assertion |

# 57.3.3.18 Receive Data Register (RDR)

#### 57.3.3.18.1 Offset

| Register | Offset |
|---|---|
| RDR | 74h |

#### 57.3.3.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 57.3.3.18.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

## 57.3.4 LPSPI3 register descriptions

### 57.3.4.1 LPSPI3 Memory map

LPSPI3 base address: 4103_5000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0004h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_0202h |
| 10h | Control Register (CR) | 32 | RW | 0000_0000h |
| 14h | Status Register (SR) | 32 | W1C | 0000_0001h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 18h | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable Register (DER) | 32 | RW | 0000_0000h |
| 20h | Configuration Register 0 (CFGR0) | 32 | RW | 0000_0000h |
| 24h | Configuration Register 1 (CFGR1) | 32 | RW | 0000_0000h |
| 30h | Data Match Register 0 (DMR0) | 32 | RW | 0000_0000h |
| 34h | Data Match Register 1 (DMR1) | 32 | RW | 0000_0000h |
| 40h | Clock Configuration Register (CCR) | 32 | RW | 0000_0000h |
| 58h | FIFO Control Register (FCR) | 32 | RW | 0000_0000h |
| 5Ch | FIFO Status Register (FSR) | 32 | RO | 0000_0000h |
| 60h | Transmit Command Register (TCR) | 32 | RW | 0000_001Fh |
| 64h | Transmit Data Register (TDR) | 32 | WO | 0000_0000h |
| 70h | Receive Status Register (RSR) | 32 | RO | 0000_0002h |
| 74h | Receive Data Register (RDR) | 32 | RO | 0000_0000h |

## 57.3.4.2 Version ID Register (VERID)

### 57.3.4.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 57.3.4.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 57.3.4.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module specification. Read-only field. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module specification. Read-only field. |
| 15-0<br>FEATURE | Module Identification Number<br>Returns the feature set number. Read-only field.<br>0000000000000100b - Standard feature set supporting a 32-bit shift register. |

### 57.3.4.3 Parameter Register (PARAM)

#### 57.3.4.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 57.3.4.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFIFO | | | | | | | | TXFIFO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 57.3.4.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Sets the maximum number of words in the receive FIFO, which is $2^{RXFIFO}$ |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Sets the maximum number of words in the transmit FIFO, which is $2^{TXFIFO}$ |

## 57.3.4.4 Control Register (CR)

### 57.3.4.4.1 Offset

| Register | Offset |
|---|---|
| CR | 10h |

### 57.3.4.4.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | RRF | RTF | | | | | DBGEN | DOZEN | RST | MEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.4.4.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>0b - No effect<br>1b - Receive FIFO is reset |
| 8<br>RTF | Reset Transmit FIFO<br>0b - No effect<br>1b - Transmit FIFO is reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>0b - Module is disabled in debug mode<br>1b - Module is enabled in debug mode |
| 2<br>DOZEN | Doze mode enable<br>Enables or disables the module in Doze mode<br>0b - Module is enabled in Doze mode<br>1b - Module is disabled in Doze mode |
| 1<br>RST | Software Reset<br>Reset all internal logic and registers, except the Control Register. RST remains set until cleared by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset |
| 0<br>MEN | Module Enable<br>0b - Module is disabled<br>1b - Module is enabled |

## 57.3.4.5 Status Register (SR)

### 57.3.4.5.1 Offset

| Register | Offset |
|---|---|
| SR | 14h |

### 57.3.4.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMF | REF | TEF | TCF | FCF | WCF | 0 | | | | | | RDF | TDF |
| W | | | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 57.3.4.5.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>MBF | Module Busy Flag<br>0b - LPSPI is idle<br>1b - LPSPI is busy |
| 23-14<br>— | Reserved |
| 13<br>DMF | Data Match Flag<br>Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by CFGR1[MATCFG], Configuration Register 1).<br>0b - Have not received matching data<br>1b - Have received matching data |
| 12<br>REF | Receive Error Flag<br>The Receive Error Flag will set when the Receiver FIFO overflows. When the Receive Error Flag is set, it is recommended to first end the transfer, empty the Receive FIFO, clear the Receive Error Flag and then restart the transfer from the beginning.<br>0b - Receive FIFO has not overflowed<br>1b - Receive FIFO has overflowed |
| 11<br>TEF | Transmit Error Flag<br>The Transmit Error Flag will set when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning.<br>0b - Transmit FIFO underrun has not occurred<br>1b - Transmit FIFO underrun has occurred |
| 10<br>TCF | Transfer Complete Flag<br>In master mode when the LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag will set.<br>0b - All transfers have not completed<br>1b - All transfers have completed |
| 9<br>FCF | Frame Complete Flag<br>The Frame Complete Flag will set at the end of each frame transfer, when the PCS negates.<br>0b - Frame transfer has not completed<br>1b - Frame transfer has completed |
| 8<br>WCF | Word Complete Flag<br>The Word Complete Flag will set when the last bit of a received word is sampled.<br>0b - Transfer of a received word has not yet completed<br>1b - Transfer of a received word has completed |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than FCR[RXWATER] (FIFO Control Register)<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| 0 | Transmit Data Flag |

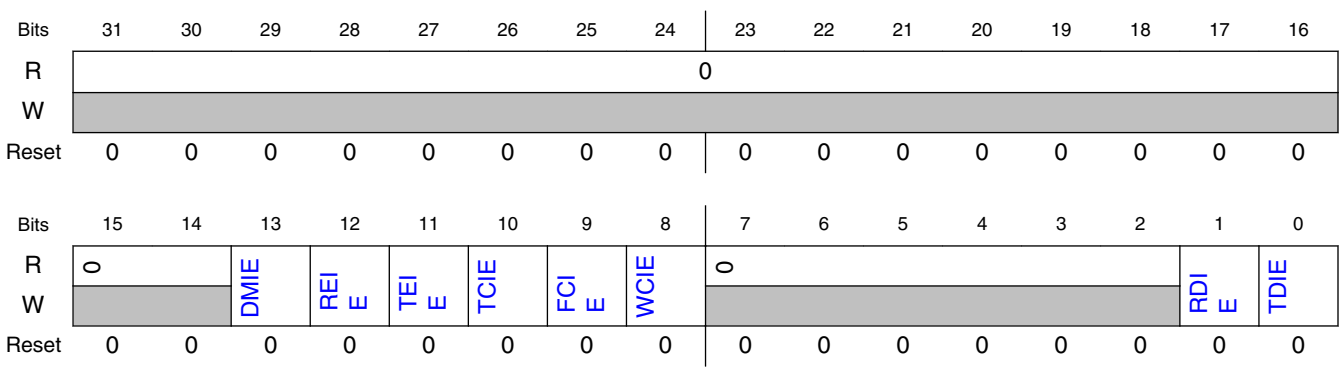
| Field | Function |
|---|---|
| TDF | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than FCR[TXWATER] (FIFO Control Register)<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

## 57.3.4.6 Interrupt Enable Register (IER)

### 57.3.4.6.1 Offset

| Register | Offset |
|---|---|
| IER | 18h |

### 57.3.4.6.2 Diagram




### 57.3.4.6.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13<br>DMIE | Data Match Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 12<br>REIE | Receive Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 11<br>TEIE | Transmit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

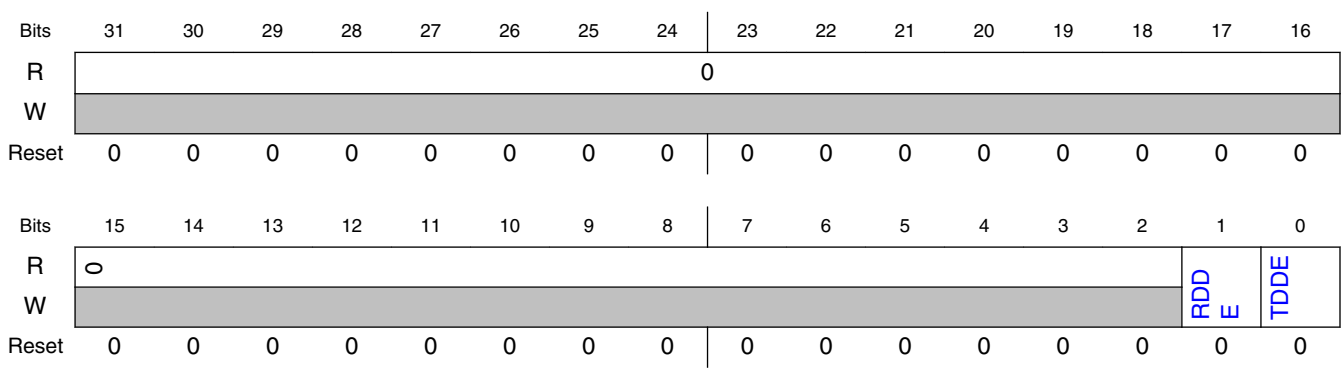
| Field | Function |
|---|---|
| 10<br>TCIE | Transfer Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 9<br>FCIE | Frame Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 8<br>WCIE | Word Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

## 57.3.4.7 DMA Enable Register (DER)

### 57.3.4.7.1 Offset

| Register | Offset |
|---|---|
| DER | 1Ch |

### 57.3.4.7.2 Diagram



#### 57.3.4.7.3 Fields

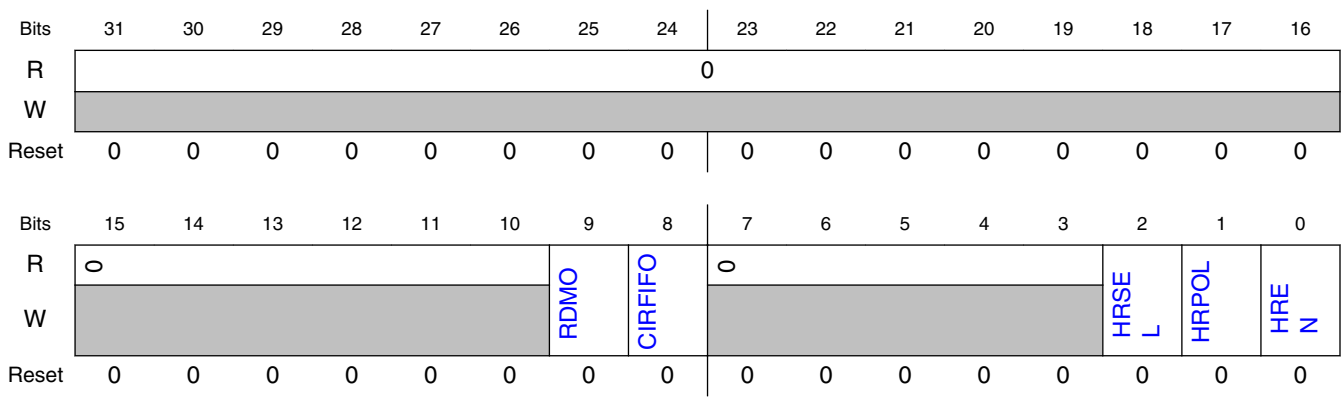
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| 0<br>TDDE | Transmit Data DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

### 57.3.4.8 Configuration Register 0 (CFGR0)

#### 57.3.4.8.1 Offset

| Register | Offset |
|---|---|
| CFGR0 | 20h |

#### 57.3.4.8.2 Diagram




#### 57.3.4.8.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9 | Receive Data Match Only |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RDMO | When Receive Data Match Only is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded.<br>• Receive Data Match Only bit should be set when the LPSPI is idle and the Data Match Flag is clear<br>• After the Data Match Flag (DMF) is set, the Receive Data Match Only bit configuration is ignored<br>• When disabling RDMO and to ensure that no receive data is lost, before clearing the Data Match Flag, first clear Receive Data Match Only (RDMO)<br><br>0b - Received data is stored in the receive FIFO as in normal operations<br>1b - Received data is discarded unless the Data Match Flag (DMF) is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as it normally is, but after the LPSPI is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This restoring of the read pointer will cause the contents of the transmit FIFO to be cycled through repeatedly.<br>0b - Circular FIFO is disabled<br>1b - Circular FIFO is enabled |
| 7-3<br>— | Reserved |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When the host request function is enabled with the LPSPI_HREQ pin, the LPSPI_PCS[1] function is disabled.<br>0b - Host request input is the LPSPI_HREQ pin<br>1b - Host request input is the input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request pin.<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable<br>When enabled in master mode, the LPSPI will only start a new SPI bus transfer if the host request input is asserted. When the LPSPI is busy, the host request input is ignored.<br>0b - Host request is disabled<br>1b - Host request is enabled |

## 57.3.4.9 Configuration Register 1 (CFGR1)

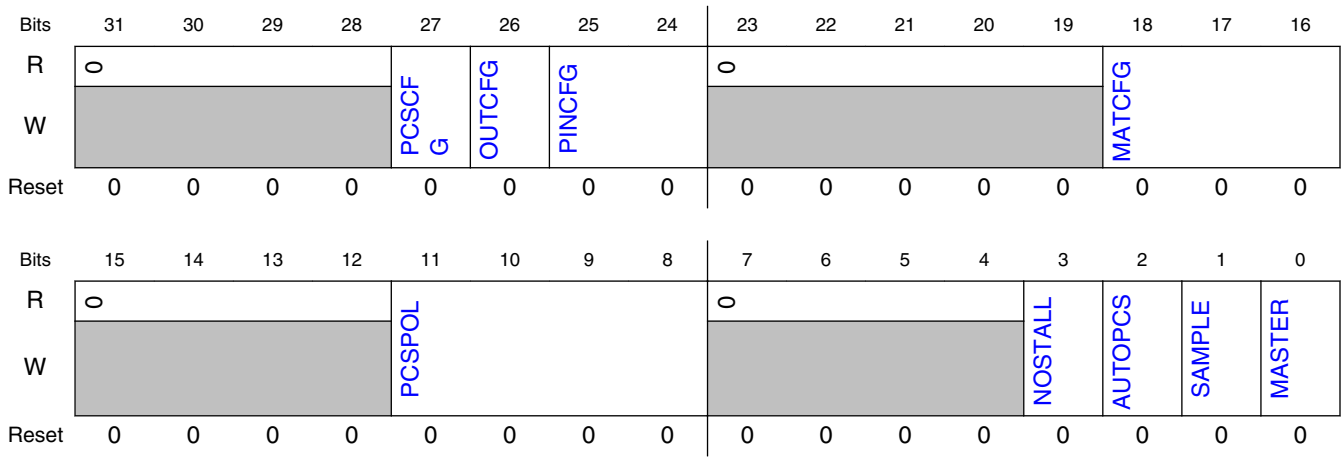
### 57.3.4.9.1 Offset

| Register | Offset |
|---|---|
| CFGR1 | 24h |

### 57.3.4.9.2 Function

The CFGR1 should only be written when the LPSPI is disabled.

### 57.3.4.9.3 Diagram




### 57.3.4.9.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>PCSCFG | Peripheral Chip Select Configuration<br>If performing 4-bit transfers, the Peripheral Chip Select Configuration bit must be set.<br>0b - PCS[3:2] are enabled<br>1b - PCS[3:2] are disabled |
| 26<br>OUTCFG | Output Config<br>Configures if the output data is tristated between accesses (LPSPI_PCS is negated).<br>0b - Output data retains last value when chip select is negated<br>1b - Output data is tristated when chip select is negated |
| 25-24<br>PINCFG | Pin Configuration<br>Configures which pins are used for input and output data during single bit transfers.<br>00b - SIN is used for input data and SOUT is used for output data<br>01b - SIN is used for both input and output data<br>10b - SOUT is used for both input and output data<br>11b - SOUT is used for input data and SIN is used for output data |
| 23-19<br>— | Reserved |
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that will cause the DMF to set.<br>**NOTE:** *Syntax:* * is boolean AND, + is boolean OR<br>000b - Match is disabled<br>001b - Reserved<br>010b - 010b - Match is enabled, if 1st data word equals MATCH0 OR MATCH1, i.e., (1st data word = MATCH0 + MATCH1) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 011b - 011b - Match is enabled, if any data word equals MATCH0 OR MATCH1, i.e., (any data word = MATCH0 + MATCH1)<br>100b - 100b - Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, i.e., [(1st data word = MATCH0) * (2nd data word = MATCH1)]<br>101b - 101b - Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, i.e., [(any data word = MATCH0) * (next data word = MATCH1)]<br>110b - 110b - Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(1st data word * MATCH1) = (MATCH0 * MATCH1)]<br>111b - 111b - Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(any data word * MATCH1) = (MATCH0 * MATCH1)] |
| 15-12<br>— | Reserved |
| 11-8<br>PCSPOL | Peripheral Chip Select Polarity<br>Configures the polarity of each Peripheral Chip Select pin.<br>0000b - The Peripheral Chip Select pin PCSx is active low<br>0001b - The Peripheral Chip Select pin PCSx is active high |
| 7-4<br>— | Reserved |
| 3<br>NOSTALL | No Stall<br>In master mode, the LPSPI will stall transfers when the transmit FIFO is empty or when the receive FIFO is full, ensuring that no transmit FIFO underrun or receive FIFO overrun can occur. Setting the No Stall bit will disable this functionality.<br>0b - Transfers will stall when the transmit FIFO is empty or the receive FIFO is full<br>1b - Transfers will not stall, allowing transmit FIFO underruns or receive FIFO overruns to occur |
| 2<br>AUTOPCS | Automatic PCS<br>For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the Automatic PCS bit will cause the LPSPI to generate an internal PCS signal at the end of each transfer word when the Clock Phase bit TCR[CPHA] = 1.<br>• When the Automatic PCS bit is set, the SCK must remain idle for at least 4 LPSPI functional clock cycles (divided by the Prescaler Value TCR[PRESCALE] configuration) between each word, to ensure correct operations<br>• In master mode, the Automatic PCS bit is ignored<br>0b - Automatic PCS generation is disabled<br>1b - Automatic PCS generation is enabled |
| 1<br>SAMPLE | Sample Point<br>When set, the LPSPI master will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data.<br>• The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode<br>• In slave mode, the SAMPLE bit is ignored<br>0b - Input data is sampled on SCK edge<br>1b - Input data is sampled on delayed SCK edge |
| 0<br>MASTER | Master Mode<br>Configures the LPSPI in master or slave mode. The Master Mode bit directly controls the direction of the LPSPI_SCK and LPCPI_PCS pins.<br>0b - Slave mode<br>1b - Master mode |

## 57.3.4.10 Data Match Register 0 (DMR0)

### 57.3.4.10.1 Offset

| Register | Offset |
|---|---|
| DMR0 | 30h |

### 57.3.4.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.4.10.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH0 | Match 0 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 0 Value is compared against the received data. |

## 57.3.4.11 Data Match Register 1 (DMR1)

### 57.3.4.11.1 Offset

| Register | Offset |
|---|---|
| DMR1 | 34h |

### 57.3.4.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.4.11.3 Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH1 | Match 1 Value<br>When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 1 Value is compared against the received data. |

## 57.3.4.12 Clock Configuration Register (CCR)

### 57.3.4.12.1 Offset

| Register | Offset |
|---|---|
| CCR | 40h |

### 57.3.4.12.2 Function

The Clock Configuration Register is only used in master mode, and the Clock Configuration Register cannot be changed when the LPSPI is enabled.

## 57.3.4.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBT | | | | | | | | SCKDIV | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 57.3.4.12.4 Fields

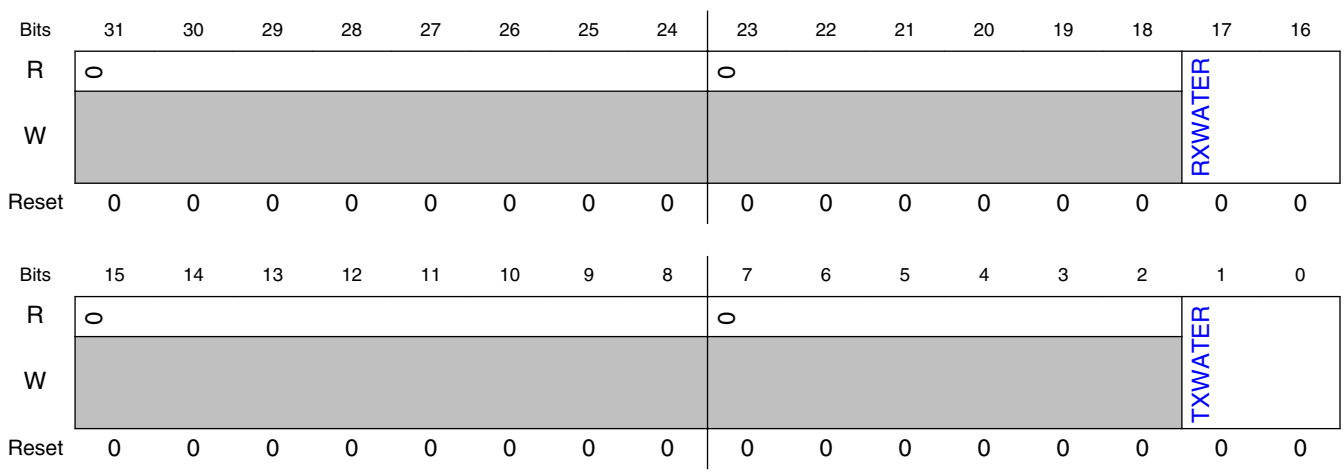
| Field | Function |
|---|---|
| 31-24<br>SCKPCS | SCK-to-PCS Delay<br>In master mode: configures the delay from the last SCK edge to the PCS negation.<br>• The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 23-16<br>PCSSCK | PCS-to-SCK Delay<br>In master mode: configures the delay from the PCS assertion to the first SCK edge.<br>• The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum delay is 1 cycle. |
| 15-8<br>DBT | Delay Between Transfers<br>In master mode:<br>• Configures the delay from the PCS negation to the next PCS assertion.<br>  • The delay is equal to (DBT + 2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 2 cycles.<br>  • Half of the delay occurs before PCS assertion and the other half of the delay occurs after PCS negation. If the command word is updated between 2 transfers, then the command word is updated half-way between the PCS negation of the last transfer and PCS assertion of the next transfer.<br>  • The command word sets which PCS signal is used (of PCS[3:0]}, the polarity/phase of the SCK signal, and the Prescaler Value.<br>• Configures the delay from the last SCK edge of a transfer word and the first SCK edge of the next transfer word, in a continuous transfer.<br>  • The delay is equal to (DBT + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>  • The minimum delay is 1 cycle. |
| 7-0<br>SCKDIV | SCK Divider<br>In master mode, the SCK Divider configures the divide ratio of the SCK pin.<br>• The SCK period is equal to (SCKDIV+2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.<br>• The minimum SCK period is 2 cycles.<br>• If the SCK period is an odd number of cycles, then the 1st half of the SCK period will be 1 cycle longer than the 2nd half of the SCK period. |

# 57.3.4.13 FIFO Control Register (FCR)

## 57.3.4.13.1 Offset

| Register | Offset |
|---|---|
| FCR | 58h |

## 57.3.4.13.2 Diagram




## 57.3.4.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal or greater than the FIFO size will be truncated. |
| 15-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1-0 | Transmit FIFO Watermark |

| Field | Function |
|---|---|
| TXWATER | The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal or greater than the FIFO size will be truncated. |

## 57.3.4.14 FIFO Status Register (FSR)

### 57.3.4.14.1 Offset

| Register | Offset |
|---|---|
| FSR | 5Ch |

### 57.3.4.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | RXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | TXCOUNT | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.4.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Returns the number of words currently stored in the receive FIFO. |
| 15-8<br>— | Reserved |
| 7-3<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Returns the number of words currently stored in the transmit FIFO. |

## 57.3.4.15 Transmit Command Register (TCR)

### 57.3.4.15.1 Offset

| Register | Offset |
|---|---|
| TCR | 60h |

### 57.3.4.15.2 Function

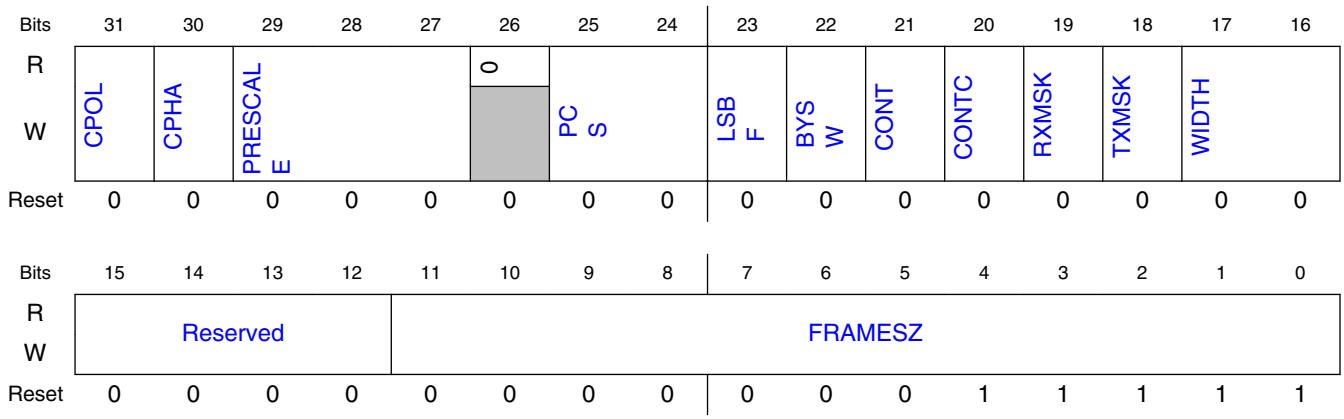
Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that the data are written. Command Register writes will be tagged and cause the command register to update, after that entry reaches the top of the FIFO. This allows changes to the command word and the transmit data itself to be interleaved. Changing the command word will cause all subsequent SPI bus transfers to be performed using the new command word.

- **In master mode,** writing a new command word does not initiate a new transfer, unless TXMSK is set. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK set). Hardware will clear TXMSK when the LPSPI_PCS negates.
- **In master mode,** if the command word is changed before an existing frame has completed, then the existing frame will terminate and the command word will then update. The command word can be changed during a continuous transfer, if CONTC of the new command word is set and the command word is written on a frame size boundary.
- **In slave mode,** the command word should be changed only when the LPSPI is idle and there is no SPI bus transfer.

**Avoid register reading problems:** Reading the Transmit Command Register will return the current state of the command register. Reading the Transmit Command Register at the same time that the Transmit Command Register is loaded from the transmit FIFO, can return an incorrect Transmit Command Register value. It is recommended:

- to either read the Transmit Command Register when the transmit FIFO is empty,
- or to read the Transmit Command Register more than once and then compare the returned values.

## 57.3.4.15.3 Diagram




## 57.3.4.15.4 Fields

| Field | Function |
|---|---|
| 31<br>CPOL | Clock Polarity<br>The Clock Polarity field is only updated between frames.<br>0b - The inactive state value of SCK is low<br>1b - The inactive state value of SCK is high |
| 30<br>CPHA | Clock Phase<br>The Clock Phase field is only updated between frames.<br>0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK<br>1b - Data is changed on the leading edge of SCK and captured on the following edge of SCK |
| 29-27<br>PRESCALE | Prescaler Value<br>For all SPI bus transfers, the Prescaler value applied to the clock configuration register. The Prescaler Value field is only updated between frames.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |
| 26<br>— | Reserved |
| 25-24<br>PCS | Peripheral Chip Select<br>Configures the peripheral chip select used for the transfer. The Peripheral Chip Select field is only updated between frames.<br>00b - Transfer using LPSPI_PCS[0]<br>01b - Transfer using LPSPI_PCS[1]<br>10b - Transfer using LPSPI_PCS[2]<br>11b - Transfer using LPSPI_PCS[3] |
| 23 | LSB First |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LSBF | 0b - Data is transferred MSB first<br>1b - Data is transferred LSB first |
| 22<br>BYSW | Byte Swap<br>Byte swap will swap the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers).<br>0b - Byte swap is disabled<br>1b - Byte swap is enabled |
| 21<br>CONT | Continuous Transfer<br>• In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.<br>• In slave mode, when continuous transfer is enabled, the LPSPI will only transmit the first FRAMESZ bits; after which the LPSPI will transmit received data (assuming a 32-bit shift register).<br><br>0b - Continuous transfer is disabled<br>1b - Continuous transfer is enabled |
| 20<br>CONTC | Continuing Command<br>In master mode, the Continuing Command bit allows the command word to be changed within a continuous transfer.<br>• The initial command word must enable continuous transfer (CONT=1),<br>• the continuing command must set this bit (CONTC=1),<br>• and the continuing command word must be loaded on a frame size boundary.<br><br>For example, if the continuous transfer has a frame size of 64-bits, then a continuing command word must be loaded on a 64-bit boundary.<br><br>0b - Command word for start of new transfer<br>1b - Command word for continuing transfer |
| 19<br>RXMSK | Receive Data Mask<br>When set, receive data is masked (receive data is not stored in receive FIFO).<br>0b - Normal transfer<br>1b - Receive data is masked |
| 18<br>TXMSK | Transmit Data Mask<br>When set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer.<br>0b - Normal transfer<br>1b - Mask transmit data |
| 17-16<br>WIDTH | Transfer Width<br>For 2-bit or 4-bit transfers, either Receive Data Mask (RXMSK) or Transmit Data Mask (TXMSK) must be set.<br>00b - 1 bit transfer<br>01b - 2 bit transfer<br>10b - 4 bit transfer<br>11b - Reserved |
| 15-12<br>— | Reserved. Software should only write zero to this field. |
| 11-0<br>FRAMESZ | Frame Size<br>Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. |

| Field | Function |
|---|---|
| | • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32 bits, and the 3rd word is 8 bits. |

## 57.3.4.16 Transmit Data Register (TDR)

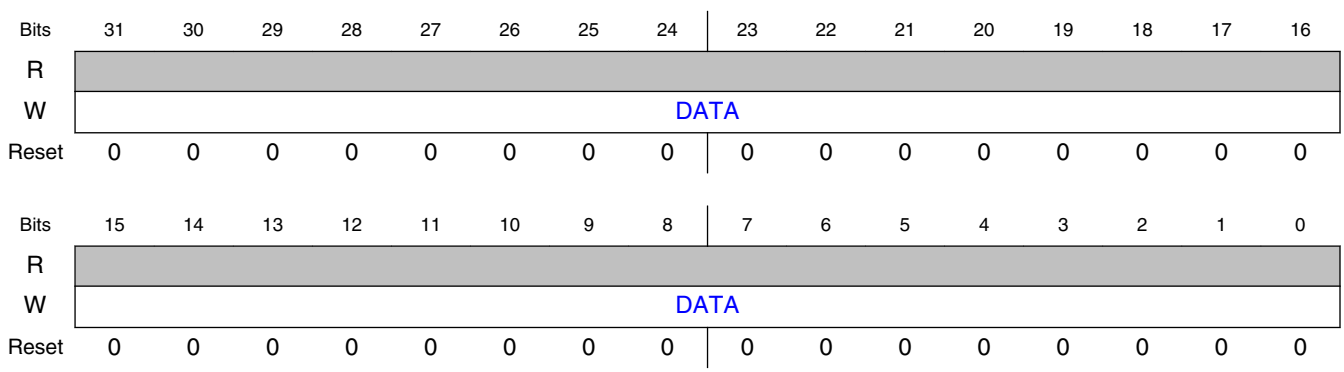
### 57.3.4.16.1 Offset

| Register | Offset |
|---|---|
| TDR | 64h |

### 57.3.4.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that it (the data) was written.

### 57.3.4.16.3 Diagram




### 57.3.4.16.4 Fields

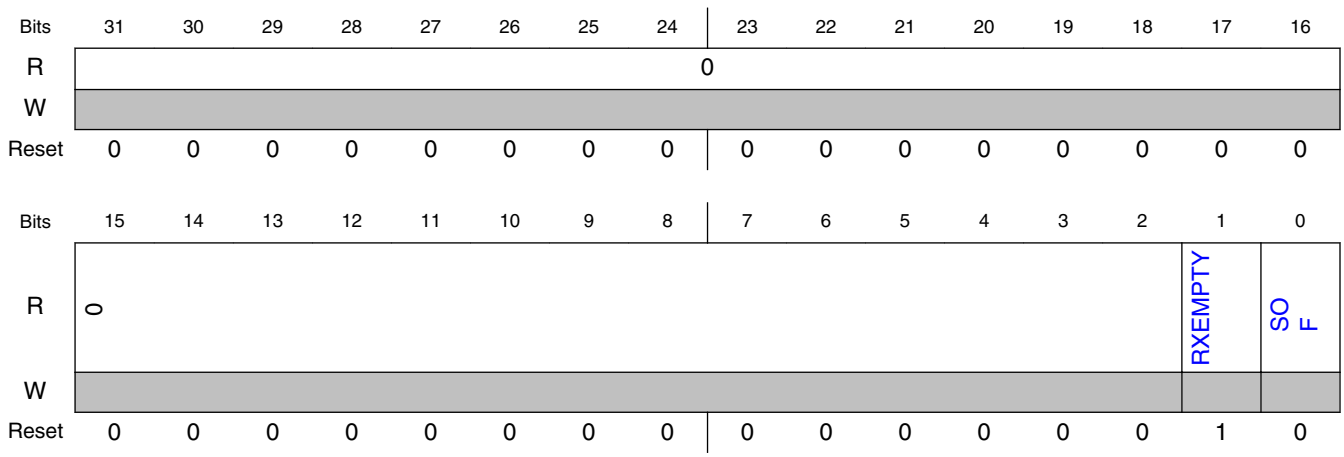
| Field | Function |
|---|---|
| 31-0<br>DATA | Transmit Data<br>Both 8-bit and 16-bit writes of transmit data will zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes. |

# 57.3.4.17 Receive Status Register (RSR)

## 57.3.4.17.1 Offset

| Register | Offset |
|---|---|
| RSR | 70h |

## 57.3.4.17.2 Diagram




## 57.3.4.17.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RXEMPTY | RX FIFO Empty<br>0b - RX FIFO is not empty<br>1b - RX FIFO is empty |
| 0<br>SOF | Start Of Frame<br>Indicates that this is the first data word received after LPSPI_PCS assertion.<br>0b - Subsequent data word received after LPSPI_PCS assertion<br>1b - First data word received after LPSPI_PCS assertion |

# 57.3.4.18 Receive Data Register (RDR)

### 57.3.4.18.1 Offset

| Register | Offset |
|---|---|
| RDR | 74h |

### 57.3.4.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 57.3.4.18.3 Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

# 57.4 Functional description

## 57.4.1 Clocking and resets

**Table 57-4. Clocks**

| | |
|---|---|
| LPSPI Functional clock | • The LPSPI functional clock is asynchronous to the bus clock.<br>• If the LPSPI functional clock remains enabled in low power modes, then LPSPI can perform SPI bus transfers and low power wakeups, in both master and slave modes.<br>• The LPSPI divides the functional clock by a prescaler; the resulting frequency must be at least 2 times faster than the SPI external clock frequency (LPSPI_SCK). |
| External clock | • The LPSPI shift register is clocked directly by the LPSPI_SCK clock.<br>• How the LPSPI_SCK clock is generated or supplied depends upon the mode (master or slave):<br>  • in master mode: the LPSPI_SCK clock is generated internally.<br>  • in slave mode: the LPSPI_SCK clock is supplied externally. |

*Table continues on the next page...*

**Table 57-4. Clocks (continued)**

| | |
|---|---|
| Bus clock | The bus clock is only used for bus accesses to the LPSPI control and configuration registers. The bus clock frequency must be high enough to support the data bandwidth requirements of the LPSPI registers, including the FIFOs. |

**Table 57-5. Resets**

| | |
|---|---|
| Chip reset | Resets the LPSPI logic and registers to their default state. |
| Software reset | • Resets the LPSPI logic and registers to their default state, except for the Control Register.<br>• The LPSPI software reset is in the Control Register CR[RST]. |
| FIFO resets | • Resets the transmit/command FIFO and the receive FIFO.<br>• Control Register CR[RTF](Reset Transmit FIFO) and CR[RRF] (Reset Receive FIFO) are write-only bits.<br>• After being reset,a FIFO is empty. |

## 57.4.2 Master Mode

### 57.4.2.1 Transmit and Command FIFO commands

The transmit and command FIFO is a combined FIFO that includes both transmit data words and command words.

- Transmit data words are stored to the transmit/command FIFO, by writing the Transmit Data Register (TDR).
- Command words are stored to the transmit/command FIFO, by writing the Transmit Command Register (TCR).

When a command word is at the top of the transmit/command FIFO, the actions that can occur depend upon whether the LPSPI module is either busy or between frames, the Continuous Transfer bit (TCR[CONT]), and the Continuing Command bit (TCR[CONTC]).

**Table 57-6. Possible actions when a command word is at the top of the transmit/command FIFO**

| Condition | Action |
|---|---|
| If the LPSPI is between frames | then the command word is pulled from the FIFO, and that command word controls all subsequent transfers. |
| If LPSPI is busy and the Continuous Transfer bit (TCR[CONT]) is set or cleared and the Continuing Command bit (TCR[CONTC]) is cleared | then the SPI frame will complete at the end of the existing word, ignoring the FRAMESZ configuration. The command word is then pulled from the FIFO and that command word controls all subsequent transfers (or until the next update to the command word). Note that a command word with CONTC=0 will always terminate (stop) the existing transfer regardless of the previous CONT value. |

*Table continues on the next page...*

**Table 57-6. Possible actions when a command word is at the top of the transmit/command FIFO (continued)**

| Condition | Action |
|---|---|
| If the LPSPI is busy and the existing Continuous Transfer bit (TCR[CONT]) is set or cleared and the new Continuing Command bit (TCR[CONTC]) value is set | then the command word must be updated at the frame boundary. The command word is pulled from the FIFO during the last LPSPI_SCK pulse of the existing frame (based on the FRAMESZ configuration), and the frame continues using the new command value for the rest of the frame (or until the next update to the command word). When the Continuing Command bit (TCR[CONTC]) is set, only the lower 24-bits of the command word are updated. If the command word is updated at a word boundary, then the transfer halts (stops) after that word. |

**NOTE**

About the Continuous Transfer bit (CONT) and Continuing Command bit (CONTC):

- Continuous Transfer bit (CONT)=1 is used to keep PCS asserted at end of frame, allowing the transfer to continue.
- Continuing Command bit (CONTC)=1 is used to indicate that this command word should not terminate the existing frame, and the transfer can continue using the new command word.

The Continuing Command bit (CONTC)=1 is restricted in the sense that the new command must load on a frame boundary, and the only way for a transfer to continue from a frame boundary, is when the previous command has the Continuous Transfer bit (CONT)=1.

The current state of the existing command word can be read by reading the Transmit Command Register (TCR). It requires at least 3 LPSPI functional clock cycles for the transmit command register to update after the transmit command register is written (assuming an empty FIFO) and the LPSPI must be enabled (Module Enable CR[MEN] bit is set).

Writing the transmit command register does not initiate a SPI bus transfer, unless the Transmit Data Mask (TCR[TXMSK]) bit is set. When the Transmit Data Mask bit is set, a new command word will not be loaded until the end of the existing frame (based on FRAMESZ configuration); at the end of the transfer, the TXMSK bit will be cleared.

In master mode, the LPSPI command word in the Transmit Command Register (TCR) controls SPI attributes (using bits and fields in registers).

## Table 57-7. LPSPI Command Word in Master Mode

| Transmit Command Register (TCR) | | Description | Can this bit/ field be modified during a data transfer? |
|---|---|---|---|
| Bit/Field | Name | | |
| CPOL | Clock Polarity | Configures the polarity of the LPSPI_SCK pin. Any change of CPOL value will cause a transition on the LPSPI_SCK pin. | N |
| CPHA | Clock Phase | Configures the clock phase of the transfer. | N |
| PRESCALE | Prescaler Value | Configures a prescaler used to divide the LPSPI functional clock, to generate the timing parameters of the SPI bus transfer. Changing PRESCALE in conjunction with PCS enables the LPSPI module to connect to different slave devices at different frequencies. | N |
| PCS | Peripheral Chip Select | Configures which LPSPI_PCS asserts for the transfer; the polarity of LPSPI_PCS is static and configured by PCSPOL. If PCSCFG is set, then PCS[3:2] should not be selected. | N |
| LSBF | LSB First | Configures if LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first. | Y |
| BYSW | Byte Swap | Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing to devices that organize data as big-endian. | Y |
| CONT | Continuous Transfer | Configures for a continuous transfer that keeps PCS asserted between frames (as configured by FRAMESZ). A new command word is required to cause PCS to negate. Also supports changing the command word at the frame size boundaries. | Y |
| CONTC | Continuing Command | Indicates that this is a new command word for the existing continuous transfer. The CONTC bit when set must only be written to the transmit/ command FIFO on a frame bounday. | Y |
| RXMSK | Receive Data Mask | Masks the receive data and does not store the masked receive data to the receive FIFO or perform receive data matching. Useful for half-duplex transfers or to configure which fields are compared during receive data matching. | Y |
| TXMSK | Transmit Data Mask | Masks the transmit data, so that masked transmit data is not pulled from transmit FIFO, and the output data pin is tristated (unless configured by Output Config CFGR1[OUTCFG]). Useful for half-duplex transfers. | Y |
| WIDTH | Transfer Width | Configures the number of bits shifted on each LPSPI_SCK pulse.<br>• 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats.<br>• 2-bit and 4-bit transfers are useful for interfacing to QuadSPI memory devices, and only support half-duplex data formats, and at least one bit (Transmit Data Mask (TCR[TXMSK] or Receive Data Mask TCR[RXMSK]) must also be set. | Y |
| FRAMESZ | Frame Size | Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits.<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported.<br>• If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32-bit, and the 3rd word is 8-bit. | Y |

**SPI bus transfers:**

- The LPSPI initiates a SPI bus transfer when:
  - data is written to the transmit FIFO
  - and the HREQ pin is asserted (or disabled)
  - and the LPSPI is enabled
- To perform the SPI bus transfer, LPSPI uses the attributes configured in the Transmit Command Register (TCR) and uses the timing parameters in the Clock Configuration Register (CCR).
- The SPI bus transfer ends after the FRAMESZ configuration is reached, or at the end of a word when a new transmit command word is at the top of the transmit/command FIFO.
- The HREQ input is only checked on the next time that the LPSPI goes idle (the LPSPI completes the current transfer and the Transmit Command Register (TCR) is empty).

**Circular FIFO:** The transmit/command FIFO also supports a Circular FIFO feature, which enables the LPSPI master to (periodically) repeat a short data transfer that can fit within the transmit/command FIFO, without requiring additional FIFO accesses. When the circular FIFO is enabled, the current state of the FIFO read pointer is saved and the status flags do not update. After the transmit/command FIFO is considered empty and the LPSPI is idle, the FIFO read pointer is restored with the saved version, so the contents of the transmit/command FIFO are not permanently pulled from the FIFO while circular FIFO mode is enabled.

#### 57.4.2.2 Receive FIFO and Data Match

The receive FIFO is used to store receive data during SPI bus transfers. When the Receive Data Mask TCR[RXMSK] bit is set, the receive data is discarded instead of being stored in the receive FIFO.

- The receive data is written to the receive FIFO at the end of the frame.
- During a multiple word or continuous transfer, the receive data is also written to the receive FIFO at the same time as the new transmit data is read from the transmit FIFO.
- During a continuous transfer, if the transmit FIFO is empty, then the receive data is only written to the receive FIFO after the transmit FIFO is written or after the Transmit Command Register (TCR) is written to end the frame.

Receive data supports a receive data match function that can match received data against one of two words or against a masked data word. The receive data match function can also be configured to compare only the first one or two received data words since the start of the frame.

- Received data that is already discarded due to the Receive Data Mask TCR[RXMSK] bit cannot cause the data match to set, and will delay the receive data match on the first received data word, until all discarded data is received.
- The receiver data match function can also be configured to discard all received data until a data match is detected, using the Receive Data Match Only CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, first clear the Receive Data Match Only CFGR0[RDMO] bit, then clear the Data Match Flag SR[DMF] bit.

### 57.4.2.3 Timing Parameters

The timing parameters that are used for all SPI bus transfers are relative to the LPSPI functional clock divided by the PRESCALE configuration. Although the Clock Configuration Register (CCR) cannot be changed when the LPSPI module is busy, to support interfacing to different slave devices at different frequencies, the Prescaler Value TCR[PRESCALE] configuration can be changed between SPI bus transfers using the Transmit Command Register (TCR).

**Table 57-8. LPSPI Timing Parameters**

| Clock Configuration Register (CCR) | | Description | Min | Max |
|---|---|---|---|---|
| Bit/Field | Name | | | |
| SCKDIV | SCK Divider | Configures the LPSPI_SCK clock period to (SCKDIV +2) cycles. When SCK Divider is configured to an odd number of cycles, the first half of the LPSPI_SCK cycle is one cycle longer than the second half of the LPSPI_SCK cycle. | 0 (2 cycles) | 255 (257 cycles) |
| DBT | Delay Between Transfers | Configures the minimum delay between PCS negation and the next PCS assertion to (DBT + 2) cycles. When the command word is updated between transfers, there is a minimum of (DBT/2)+1 cycles between the command word update and any change on LPSPI_PCS pins. | 0 (2 cycles) | 255 (257 cycles) |
| DBT | Delay Between Transfers | Configures the delay during a continuous transfer between the last SCK edge of a frame and the first SCK edge of the continuing frame to (DBT + 1) cycles. This is useful when the external slave requires a large delay between different words of an SPI bus transfer. | 0 (1 cycle) | 255 (256 cycles) |
| PCSSCK | PCS-to-SCK Delay | Configures the minimum delay between PCS assertion and the first SCK edge to (PCSSCK + 1) cycles. | 0 (1 cycle) | 255 (256 cycles) |
| SCKPCS | SCK-to-PCS Delay | Configures the minimum delay between the last SCK edge and the PCS assertion to (SCKPCS + 1) cycles. | 0 (1 cycle) | 255 (256 cycles) |

#### 57.4.2.4 Pin Configuration

- To swap directions or support half-duplex transfers on the same pin, the LPSPI_SIN and LPSPI_SOUT pins can be configured using the Pin Configuration CFGR1[PINCFG].
- To determine if an output data pin (like LPSPI_SOUT) will tristate when the LPSPI_PCS is negated, or if the output data pin will simply retain the last value, use the Output Config CFGR1[OUTCFG].
- When configuring for half-duplex transfers using the same data pin in single-bit transfer mode, or when configuring any transfer in 2-bit and 4-bit transfer modes, the output data pins must be configured to tristate when LPSPI_PCS is negated; use the Output Config CFGR1[OUTCFG].
- When performing quad-data transfers, the Peripheral Chip Select Configuration CFGR1[PCSCFG] must be enabled. The Peripheral Chip Select Configuration CFGR1[PCSCFG] is also used to disable LPSPI_PCS[3:2] functions.

#### 57.4.2.5 Clock Loopback

The LPSPI master can be configured to use 1 of 2 clocks to sample the input data (for example, LPSPI_SIN):

- either the LPSPI_SCK output clock directly
- or a delayed version of the LPSPI_SCK output clock

The delayed version of the LPSPI_SCK is delayed by the LPSPI_SCK pin output delay, plus the LPSPI_SCK pin input delay, and is configured by setting CFGR1[SAMPLE]. Enabling the loopback version of the LPSPI_SCK can improve the setup time of the input data from the slave.

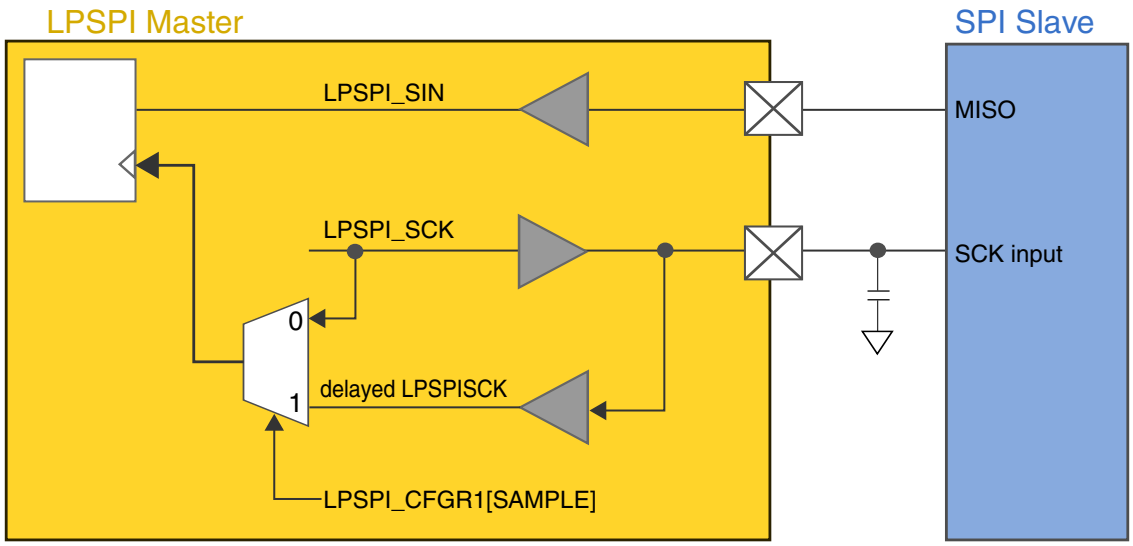



**Figure 57-7. Clock Loopback**

See the device datasheet for the specific input setup time in master loopback mode.

## 57.4.3 Slave Mode

LPSPI slave mode:

- uses the same shift register and logic that master mode uses
- does not use the Clock Configuration Register (CCR)
- during SPI bus transfers, requires that the Transmit Command Register (TCR) remain static (unchanging)

### 57.4.3.1 Transmit and Command FIFO commands

Before enabling the LPSPI in slave mode, the Transmit Command Register (TCR) should be initialized, although the Transmit Command Register register will not update until after the LPSPI is enabled. After being enabled, the Transmit Command Register should only be changed if the LPSPI is idle. In slave mode, the LPSPI command word in the Transmit Command Register (TCR) controls SPI attributes (using bits and fields in registers). Before the LPSPI_PCS input asserts, the transmit FIFO must be filled with transmit data, or the transmit error flag will set.

**Table 57-9. LPSPI Command Word in Slave Mode**

| Transmit Command Register (TCR) | | Description |
|---|---|---|
| Bit/Field | Name | |
| CPOL | Clock Polarity | Configures the polarity of the external LPSPI_SCK input. |
| CPHA | Clock Phase | Configures the clock phase of transfer. |
| PRESCALE | Prescaler Value | Configures the LPSPI functional clock prescaler. |
| PCS | Peripheral Chip Select | Configures which LPSPI_PCS is used, the polarity of LPSPI_PCS is static and configured by PCSPOL. If PCSCFG is set, then PCS[3:2] should not be selected. |
| LSBF | LSB First | Configures if LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first. |
| BYSW | Byte Swap | Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing to devices that organize data as big-endian. |
| CONT | Continuous Transfer | When Continuous Transfer is set, only the first FRAMSZ bits will be transmitted/ received by the LPSPI module. |
| CONTC | Continuing Command | CONTC bit is reserved (in slave mode). |
| RXMSK | Receive Data Mask | Masks the receive data and does not store the masked receive data to the receive FIFO or perform receive data matching. Useful for half-duplex transfers or to configure which fields are compared during receive data matching. |
| TXMSK | Transmit Data Mask | Masks the transmit data, so that the masked transmit data is not pulled from transmit FIFO, and the output data pin is tristated (unless configured by Output Config CFGR1[OUTCFG]). Useful for half-duplex transfers. |
| WIDTH | Transfer Width | Configures the number of bits shifted on each LPSPI_SCK pulse.<br>• 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats.<br>• 2-bit and 4-bit transfers are useful for interfacing to QuadSPI memory devices, and only support half-duplex data formats, and at least one bit (Transmit Data Mask TCR[TXMSK] or Receive Data Mask (TCR[RXMSK]) must also be set. |
| FRAMESZ | Frame Size | Configures the frame size in number of bits equal to (FRAMESZ + 1).<br>• The minimum frame size is 8 bits.<br>• The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported.<br>• If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32-bit, and the 3rd word is 8-bit. |

## 57.4.3.2 Receive FIFO and Data Match

The receive FIFO is used to store received data during SPI bus transfers. When the Receive Data Mask TCR[RXMSK] is set, the received data is discarded (instead of storing the received data in the receive FIFO).

Receive data supports a receive data match function that can match received data against one of two words or against a masked data word. The data match function can also be configured to compare only the first one or two received data words since the start of the frame.

- Received data that is already discarded (because the Receive Data Mask TCR[RXMSK] is set) cannot cause the data match to set, and will delay the match on the first received data word, until all discarded data is received.
- The receiver match function can also be configured to discard all received data until a data match is detected, using the Receive Data Match Only CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, first clear the Receive Data Match Only CFGR0[RDMO] bit, then clear the Data Match Flag SR[DMF] bit.

#### 57.4.3.3 Clocked Interface

The LPSPI module supports interfacing to external masters that provide only clock and data pins (LPSPI_PCS is not required). This interface requires:

- using Clock Phase TCR[CPHA] = 1 (data is changed on the leading edge of SCK and captured on the following edge)
- configuring the LPSPI_PCS input to be always asserted (configure the Peripheral Chip Select Polarity CFGR1[PCSPOL[n]] = 1). For example, to configure PCS[0] to be always asserted, set PCSPOL[0] = 1, and don't configure PCS[0] in the pin muxing.
- setting the Automatic PCS CFGR1[AUTOPCS] bit = 1 (Automatic PCS generation is enabled). When Automatic PCS generation (AUTOPCS) is set, a minimum of 4 LPSPI functional clock cycles (divided by PRESCALE configuration) is required between the last LPSPI_SCK edge of one word and the first LPSPI_SCK edge of the next word.

### 57.4.4 Interrupts and DMA Requests

The next table lists the slave mode sources (status flags) that can generate LPSPI interrupts and LPSPI slave transmit/receive DMA requests.

**Table 57-10. LPSPI Interrupts and DMA Requests**

| Status Register (SR) | | Description | Can generate | | |
|---|---|---|---|---|---|
| Status Flag | Name | | Interrupt? | DMA Request? | Low Power Wakeup? |
| TDF | Transmit Data Flag | Data can be written to transmit FIFO, as configured by the Transmit FIFO Watermark FCR[TXWATER] | Y | TX | Y |
| RDF | Receive Data Flag | Data can be read from the receive FIFO, as configured by the Receive FIFO Watermark FCR[RXWATER] | Y | RX | Y |
| WCF | Word Complete Flag | Word is complete, the last bit of the word has been sampled | Y | N | Y |
| FCF | Frame Complete Flag | Frame is complete, and PCS has negated | Y | N | Y |
| TCF | Transfer Complete Flag | Transfer is complete, PCS has negated, and the transmit/command FIFO is empty | Y | N | Y |
| TEF | Transmit Error Flag | Indicates a transmit/command FIFO underrun. In master mode when the No Stall CFGR1[NOSTALL] bit is clear (0, transfers will stall when transmit FIFO is empty or receive FIFO is full), the Transmit Error Flag bit cannot set. | Y | N | Y |
| REF | Receive Error Flag | Receive error flag, indicates a receive FIFO overflow. In master mode when the No Stall CFGR1[NOSTALL] bit is clear (0, transfers will stall when transmit FIFO is empty or receive FIFO is full), the Receive Error Flag bit cannot set. | Y | N | Y |
| DMF | Data Match Flag | Indicates that the received data has matched the configured data match value | Y | N | Y |
| MBF | Module Busy Flag | LPSPI is busy performing a SPI bus transfer | N | N | N |

## 57.4.5 Peripheral Triggers

The connection of the LPSPI peripheral triggers to other peripherals depend upon the specific device being used.

**Table 57-11. LPSPI Triggers**

| Trigger | Description |
|---|---|
| Frame Output Trigger | The frame output trigger:<br>• asserts at the end of each frame (when PCS negates)<br>• remains asserted until PCS next asserts |
| Word Output Trigger | The word output trigger: |

*Table continues on the next page...*

## Table 57-11. LPSPI Triggers (continued)

| Trigger | Description | |
|---|---|---|
| | • asserts at the end of each received word<br>• remains asserted for 1 LPSPI_SCK period | The LPSPI generates 2 output triggers that can be connected to other peripherals on the device. |
| Input Trigger | To control the start of a LPSPI bus transfer, the LPSPI input trigger can be selected instead of the LPSPI_HREQ input.<br>• The LPSPI input trigger is synchronized, and must assert for at least 2 cycles of the LPSPI functional clock divided by the PRESCALE configuration, so that the input trigger can be detected.<br>• When the LPSPI module is busy, the LPSPI_HREQ input (and therefore the LPSPI input trigger) is ignored . | |

# Chapter 58
# Flexible I/O (FlexIO)

## 58.1 Chip-specific FlexIO information

### 58.1.1 FlexIO overview

The FlexIO is a highly configurable module providing a wide range of functionality including:

- Flexible shift registers for emulation of a variety of serial/parallel communication protocols
- Programmable logic arrays allowing the integration of external digital logic functions on-chip
- Up to 16-bit timer or pulse counters with trigger/interrupt generation on compare match
- DMA support

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI
- I2S
- Camera IF
- LCD RGB
- PWM / Waveform generation

The module can remain functional in VLPS mode provided the clock it is using remains enabled.

### 58.1.2 FlexIO Instantiation

This section summarize the features and module configurations of FlexIO

Amongst other features, the FlexIO module in this device supports:
1. Emulation in serial/parallel communication protocols
2. Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules

## 58.2 Introduction

### 58.2.1 Overview

The FlexIO is a highly configurable module providing a wide range of functionality including:
- Emulation of a variety of serial/parallel communication protocols
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions
- Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules
- Programmable state machine for offloading basic system control functions from CPU

### 58.2.2 Features

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI
- I2S
- Camera IF
- Motorola 68K/Intel 8080 bus
- PWM/Waveform generation

The following key features are provided:

- Array of 32-bit shift registers with transmit, receive and data match modes

- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation
- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support
- Interrupt, DMA or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions
- Programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs
- Programmable state machine for offloading basic system control functions from CPU with support for up to 8 states, 8 outputs and 3 selectable inputs per state

## 58.2.3 Block Diagram

The following diagram gives a high-level overview of the configuration of FlexIO timers and shifters.

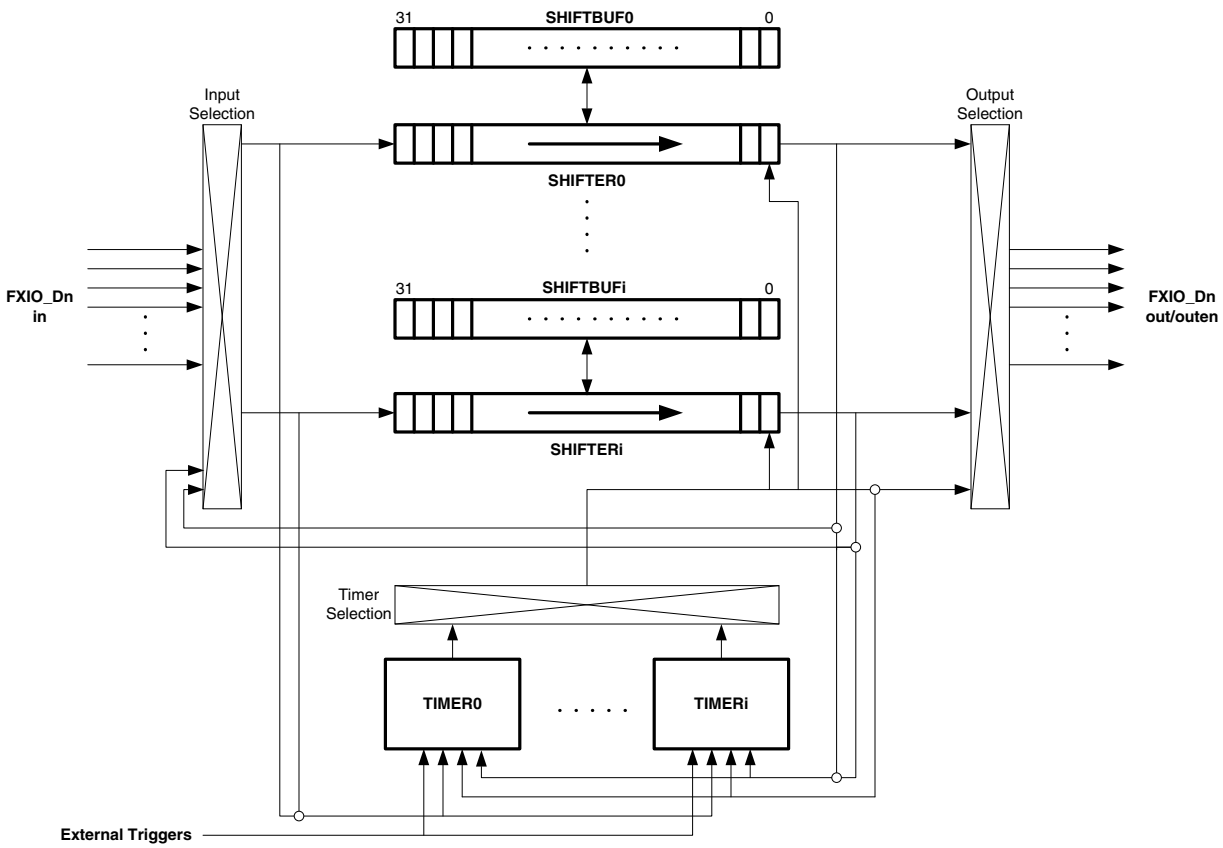



**Figure 58-1. FlexIO block diagram**

### 58.2.4 Modes of operation

The FlexIO module supports the chip modes described in the following table.

**Table 58-1. Chip modes supported by the FlexIO module**

| Chip mode | FlexIO Operation |
|---|---|
| Run | Normal operation |
| Stop/Wait | Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is clear and the FlexIO is using an external or internal clock source which remains operating during stop/wait modes. |
| LLS | The Doze Enable (CTRL[DOZEN]) bit is ignored and the FlexIO will wait for all Timers to complete any pending operation before acknowledging LLS mode entry. |
| Debug | Can continue operating provided the Debug Enable bit (CTRL[DBGE]) is set. |

### 58.2.5 FlexIO Signal Descriptions

| Signal | Description | I/O |
|---|---|---|
| FXIO_Dn (n=0...31) | Bidirectional FlexIO Shifter and Timer pin inputs/outputs | I/O |

## 58.3 Memory Map and Registers

### 58.3.1 FLEXIO register descriptions

#### 58.3.1.1 FLEXIO Memory map

FLEXIO0 base address: 4003_9000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0101_0001h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0420_0808h |
| 8h | FlexIO Control Register (CTRL) | 32 | RW | 0000_0000h |
| Ch | Pin State Register (PIN) | 32 | RO | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10h | Shifter Status Register (SHIFTSTAT) | 32 | W1C | 0000_0000h |
| 14h | Shifter Error Register (SHIFTERR) | 32 | W1C | 0000_0000h |
| 18h | Timer Status Register (TIMSTAT) | 32 | W1C | 0000_0000h |
| 20h | Shifter Status Interrupt Enable (SHIFTSIEN) | 32 | RW | 0000_0000h |
| 24h | Shifter Error Interrupt Enable (SHIFTEIEN) | 32 | RW | 0000_0000h |
| 28h | Timer Interrupt Enable Register (TIMIEN) | 32 | RW | 0000_0000h |
| 30h | Shifter Status DMA Enable (SHIFTSDEN) | 32 | RW | 0000_0000h |
| 40h | Shifter State Register (SHIFTSTATE) | 32 | RW | 0000_0000h |
| 80h - 9Ch | Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7) | 32 | RW | 0000_0000h |
| 100h - 11Ch | Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7) | 32 | RW | 0000_0000h |
| 200h - 21Ch | Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7) | 32 | RW | 0000_0000h |
| 280h - 29Ch | Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7) | 32 | RW | 0000_0000h |
| 300h - 31Ch | Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7) | 32 | RW | 0000_0000h |
| 380h - 39Ch | Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7) | 32 | RW | 0000_0000h |
| 400h - 41Ch | Timer Control N Register (TIMCTL0 - TIMCTL7) | 32 | RW | 0000_0000h |
| 480h - 49Ch | Timer Configuration N Register (TIMCFG0 - TIMCFG7) | 32 | RW | 0000_0000h |
| 500h - 51Ch | Timer Compare N Register (TIMCMP0 - TIMCMP7) | 32 | RW | 0000_0000h |
| 680h - 69Ch | Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7) | 32 | RW | 0000_0000h |
| 700h - 71Ch | Shifter Buffer N Half Word Swapped Register (SHIFTBUFHWS0 - SHIFTBUFHWS7) | 32 | RW | 0000_0000h |
| 780h - 79Ch | Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7) | 32 | RW | 0000_0000h |

## 58.3.1.2 Version ID Register (VERID)

### 58.3.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### 58.3.1.2.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### 58.3.1.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000000b - Standard features implemented.<br>0000000000000001b - Supports state, logic and parallel modes. |

### 58.3.1.3 Parameter Register (PARAM)

#### 58.3.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### 58.3.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRIGGER | | | | | | | | PIN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMER | | | | | | | | SHIFTER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

### 58.3.1.3.3 Fields

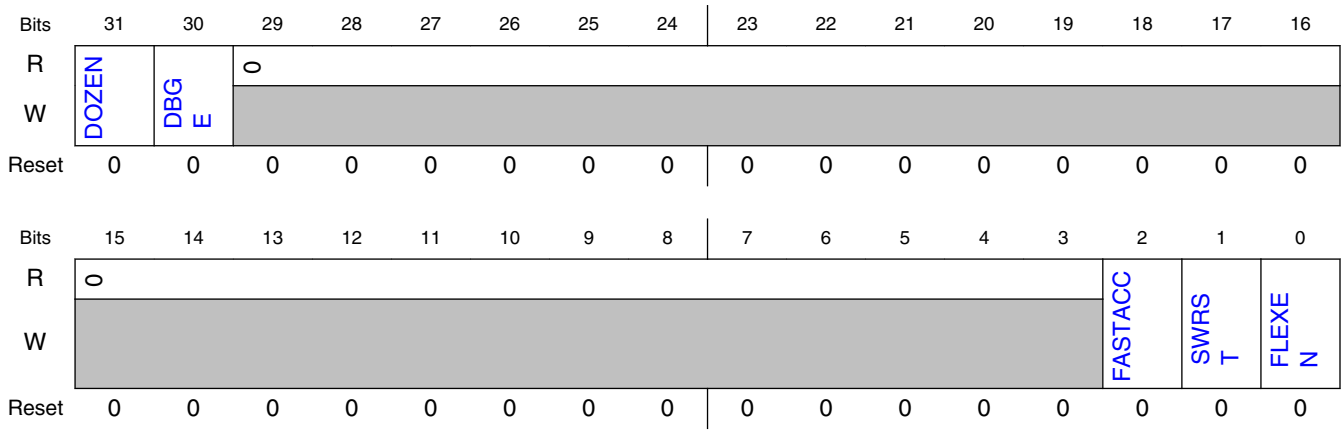
| Field | Function |
|---|---|
| 31-24<br>TRIGGER | Trigger Number<br>Number of external triggers implemented. |
| 23-16<br>PIN | Pin Number<br>Number of Pins implemented. |
| 15-8<br>TIMER | Timer Number<br>Number of Timers implemented. |
| 7-0<br>SHIFTER | Shifter Number<br>Number of Shifters implemented. |

## 58.3.1.4 FlexIO Control Register (CTRL)

### 58.3.1.4.1 Offset

| Register | Offset |
|---|---|
| CTRL | 8h |

## 58.3.1.4.2 Diagram




## 58.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31<br>DOZEN | Doze Enable<br>Disables FlexIO operation in Doze modes.<br>0b - FlexIO enabled in Doze modes.<br>1b - FlexIO disabled in Doze modes. |
| 30<br>DBGE | Debug Enable<br>Enables FlexIO operation in Debug mode.<br>0b - FlexIO is disabled in debug modes.<br>1b - FlexIO is enabled in debug modes |
| 29-3<br>— | Reserved. |
| 2<br>FASTACC | Fast Access<br>Enables fast register accesses to FlexIO registers, but requires the FlexIO functional clock to be at least twice the frequency of the bus clock.<br>0b - Configures for normal register accesses to FlexIO<br>1b - Configures for fast register accesses to FlexIO |
| 1<br>SWRST | Software Reset<br>The FlexIO Control Register is not affected by the software reset, all other logic in the FlexIO is affected by the software reset and register accesses are ignored until this bit is cleared. This register bit will remain set until cleared by software, and the reset has cleared in the FlexIO clock domain.<br>0b - Software reset is disabled<br>1b - Software reset is enabled, all FlexIO registers except the Control Register are reset. |
| 0<br>FLEXEN | FlexIO Enable<br>0b - FlexIO module is disabled.<br>1b - FlexIO module is enabled. |

## 58.3.1.5 Pin State Register (PIN)

### 58.3.1.5.1 Offset

| Register | Offset |
|---|---|
| PIN | Ch |

### 58.3.1.5.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.5.3 Fields

| Field | Function |
|---|---|
| 31-0<br>PDI | Pin Data Input<br>Returns the input data on each of the FlexIO pins. |

## 58.3.1.6 Shifter Status Register (SHIFTSTAT)

### 58.3.1.6.1 Offset

| Register | Offset |
|---|---|
| SHIFTSTAT | 10h |

### 58.3.1.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SSF | | | | | | | |
| W | | | | | | | | | W1C | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SSF | Shifter Status Flag<br>The shifter status flag is updated when one of the following events occurs:<br>For SMOD=Receive, the status flag is set when SHIFTBUF has been loaded with data from Shifter (SHIFTBUF is full), and the status flag is cleared when SHIFTBUF register is read.<br>For SMOD=Transmit, the status flag is set when SHIFTBUF data has been transferred to the Shifter (SHIFTBUF is empty) or when initially configured for SMOD=Transmit, and the status flag is cleared when the SHIFTBUF register is written.<br>For SMOD=Match Store, the status flag is set when a match has occured between SHIFTBUF and Shifter, and the status flag is cleared when the SHIFTBUF register is read.<br>For SMOD=Match Continuous, returns the current match result between the SHIFTBUF and Shifter.<br>For SMOD=State, the status flag for a shifter will set when it is selected by the current state pointer.<br>For SMOD=Logic, returns the current value of the programmable logic block output.<br>The status flag can also be cleared by writing a logic one to the flag for all modes except Match Continuous/State/Logic.<br>0b - Status flag is clear.<br>1b - Status flag is set. |

## 58.3.1.7 Shifter Error Register (SHIFTERR)

### 58.3.1.7.1 Offset

| Register | Offset |
|---|---|
| SHIFTERR | 14h |

### 58.3.1.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SEF | | | | | | | |
| W | | | | | | | | | W1C | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.7.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SEF | Shifter Error Flags<br>The shifter error flag is set when one of the following events occurs:<br>For SMOD=Receive, indicates Shifter was ready to store new data into SHIFTBUF before the previous data was read from SHIFTBUF (SHIFTBUF Overrun), or indicates that the received start or stop bit does not match the expected value.<br>For SMOD=Transmit, indicates Shifter was ready to load new data from SHIFTBUF before new data had been written into SHIFTBUF (SHIFTBUF Underrun).<br>For SMOD=Match Store, indicates a match event occured before the previous match data was read from SHIFTBUF (SHIFTBUF Overrun).<br>For SMOD=Match Continuous, the error flag is set when a match has occured between SHIFTBUF and Shifter.<br>For SMOD=Logic, the error flag is set when the output of the programmable logic block has asserted.<br>Can be cleared by writing logic one to the flag. For SMOD=Match Continuous, can also be cleared when the SHIFTBUF register is read.<br>0b - Shifter Error Flag is clear.<br>1b - Shifter Error Flag is set. |

## 58.3.1.8 Timer Status Register (TIMSTAT)

### 58.3.1.8.1 Offset

| Register | Offset |
|---|---|
| TIMSTAT | 18h |

### 58.3.1.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TSF | | | | | | | |
| W | | | | | | | | | W1C | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.8.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>TSF | Timer Status Flags<br>The timer status flag sets depending on the timer mode, and can be cleared by writing logic one to the flag.<br>In 8-bit baud counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements.<br>In 8-bit high PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements.<br>In 16-bit counter mode, the timer status flag is set when the 16-bit counter equals zero and decrements.<br>0b - Timer Status Flag is clear.<br>1b - Timer Status Flag is set. |

## 58.3.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

#### 58.3.1.9.1 Offset

| Register | Offset |
|---|---|
| SHIFTSIEN | 20h |

#### 58.3.1.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SSIE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 58.3.1.9.3 Fields

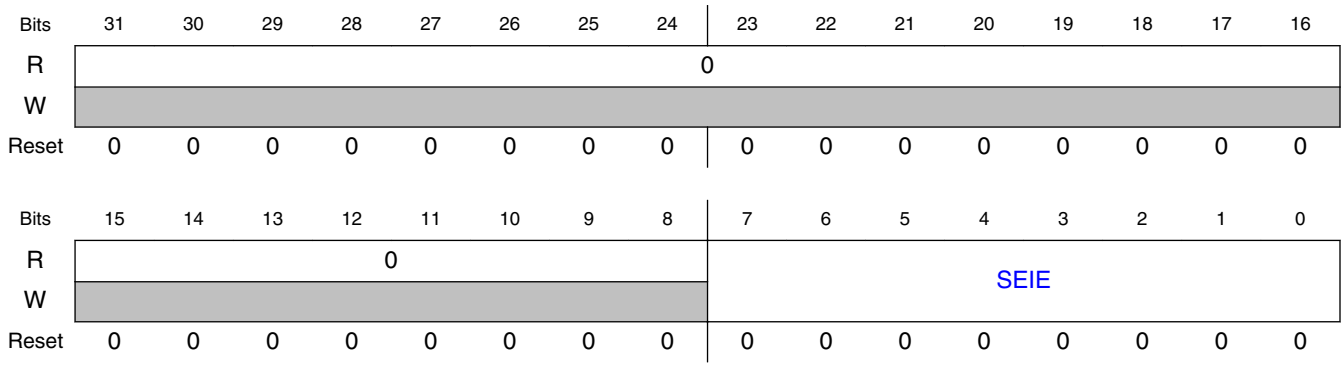
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SSIE | Shifter Status Interrupt Enable<br>Enables interrupt generation when corresponding SSF is set.<br>0b - Shifter Status Flag interrupt disabled.<br>1b - Shifter Status Flag interrupt enabled. |

### 58.3.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

#### 58.3.1.10.1 Offset

| Register | Offset |
|---|---|
| SHIFTEIEN | 24h |

### 58.3.1.10.2 Diagram




### 58.3.1.10.3 Fields

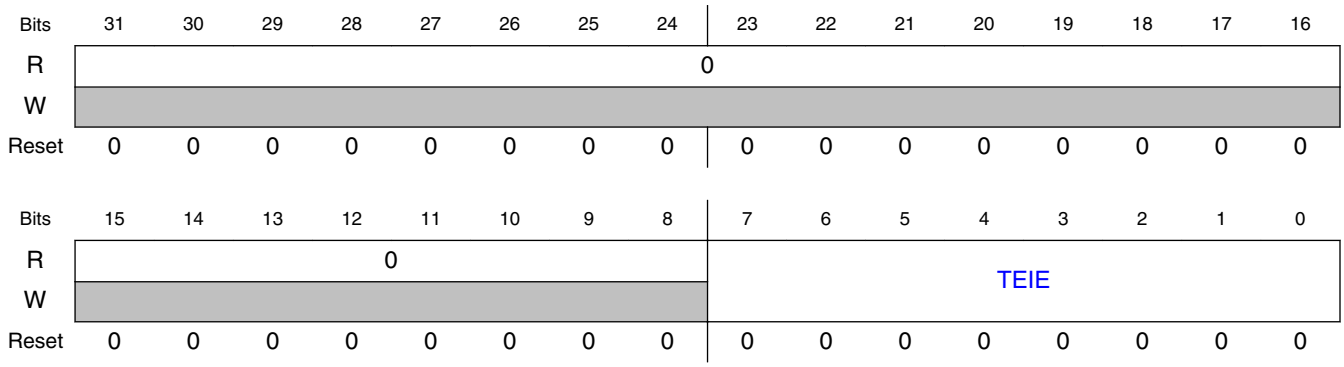
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SEIE | Shifter Error Interrupt Enable<br>Enables interrupt generation when corresponding SEF is set.<br>0b - Shifter Error Flag interrupt disabled.<br>1b - Shifter Error Flag interrupt enabled. |

## 58.3.1.11 Timer Interrupt Enable Register (TIMIEN)

### 58.3.1.11.1 Offset

| Register | Offset |
|---|---|
| TIMIEN | 28h |

### 58.3.1.11.2 Diagram




### 58.3.1.11.3 Fields

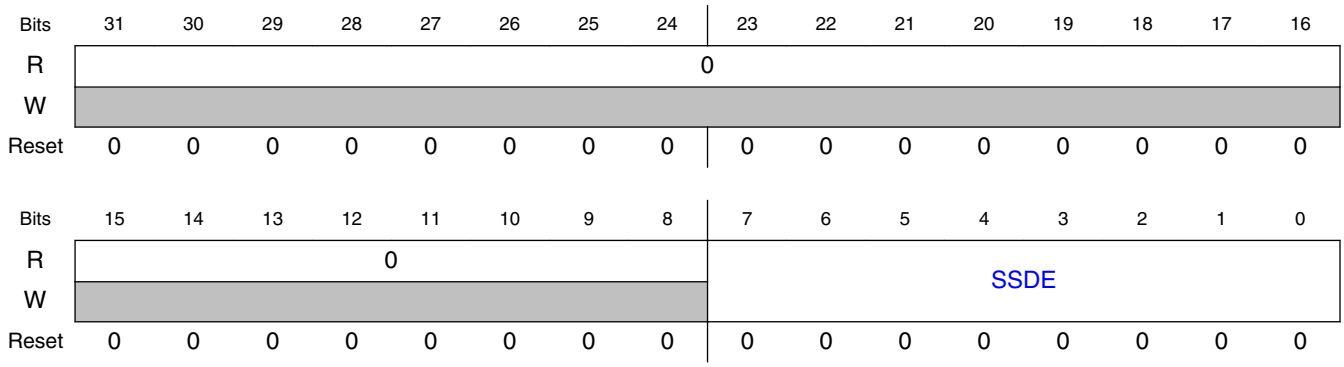
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>TEIE | Timer Status Interrupt Enable<br>Enables interrupt generation when corresponding TSF is set.<br>0b - Timer Status Flag interrupt is disabled.<br>1b - Timer Status Flag interrupt is enabled. |

## 58.3.1.12 Shifter Status DMA Enable (SHIFTSDEN)

### 58.3.1.12.1 Offset

| Register | Offset |
|---|---|
| SHIFTSDEN | 30h |

### 58.3.1.12.2 Diagram




### 58.3.1.12.3 Fields

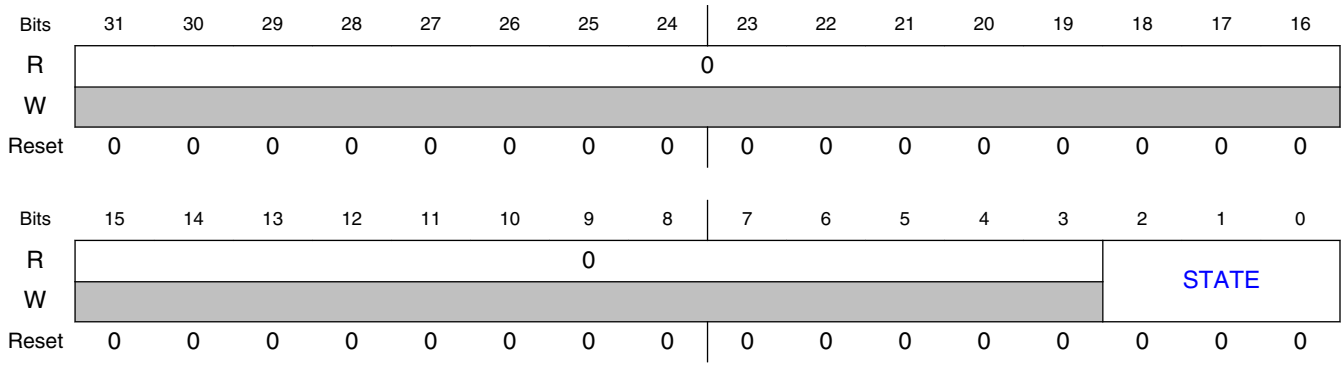
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SSDE | Shifter Status DMA Enable<br>Enables DMA request generation when corresponding SSF is set.<br>0b - Shifter Status Flag DMA request is disabled.<br>1b - Shifter Status Flag DMA request is enabled. |

## 58.3.1.13 Shifter State Register (SHIFTSTATE)

### 58.3.1.13.1 Offset

| Register | Offset |
|---|---|
| SHIFTSTATE | 40h |

### 58.3.1.13.2 Diagram




### 58.3.1.13.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2-0<br>STATE | Current State Pointer<br>The current state field maintains a pointer to keep track of the current Shifter (configured for State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the incorrect state returned. |

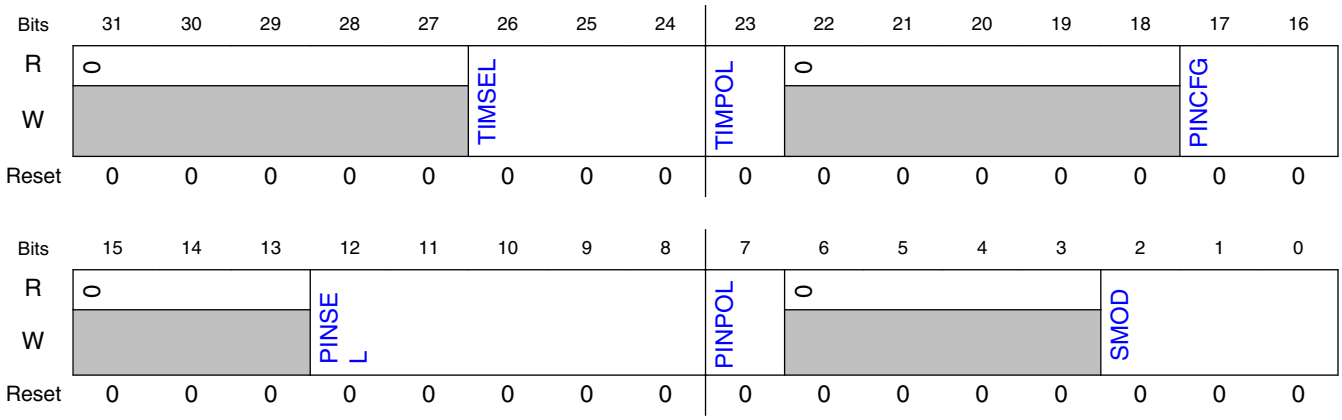
## 58.3.1.14 Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)

### 58.3.1.14.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTCTLa | 80h + (a × 4h) |

## 58.3.1.14.2 Diagram




## 58.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-24<br>TIMSEL | Timer Select<br>Selects which Timer is used for controlling the logic/shift register and generating the Shift clock. TIMSEL=i will select TIMERi. |
| 23<br>TIMPOL | Timer Polarity<br>0b - Shift on posedge of Shift clock<br>1b - Shift on negedge of Shift clock |
| 22-18<br>— | Reserved. |
| 17-16<br>PINCFG | Shifter Pin Configuration<br>For pins configured as an output (PINCFG=11), this field will take effect when the register is written.<br>00b - Shifter pin output disabled<br>01b - Shifter pin open drain or bidirectional output enable<br>10b - Shifter pin bidirectional output data<br>11b - Shifter pin output |
| 15-13<br>— | Reserved. |
| 12-8<br>PINSEL | Shifter Pin Select<br>Selects which pin is used by the Shifter input or output. PINSEL=i will select the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field will take effect when the register is written. |
| 7<br>PINPOL | Shifter Pin Polarity<br>For pins configured as an output (PINCFG=11), this field will take effect when the register is written.<br>0b - Pin is active high<br>1b - Pin is active low |
| 6-3<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2-0<br>SMOD | Shifter Mode<br>Configures the mode of the Shifter.<br>000b - Disabled.<br>001b - Receive mode. Captures the current Shifter content into the SHIFTBUF on expiration of the Timer.<br>010b - Transmit mode. Load SHIFTBUF contents into the Shifter on expiration of the Timer.<br>011b - Reserved.<br>100b - Match Store mode. Shifter data is compared to SHIFTBUF content on expiration of the Timer.<br>101b - Match Continuous mode. Shifter data is continuously compared to SHIFTBUF contents.<br>110b - State mode. SHIFTBUF contents are used for storing programmable state attributes.<br>111b - Logic mode. SHIFTBUF contents are used for implementing programmable logic look up table. |

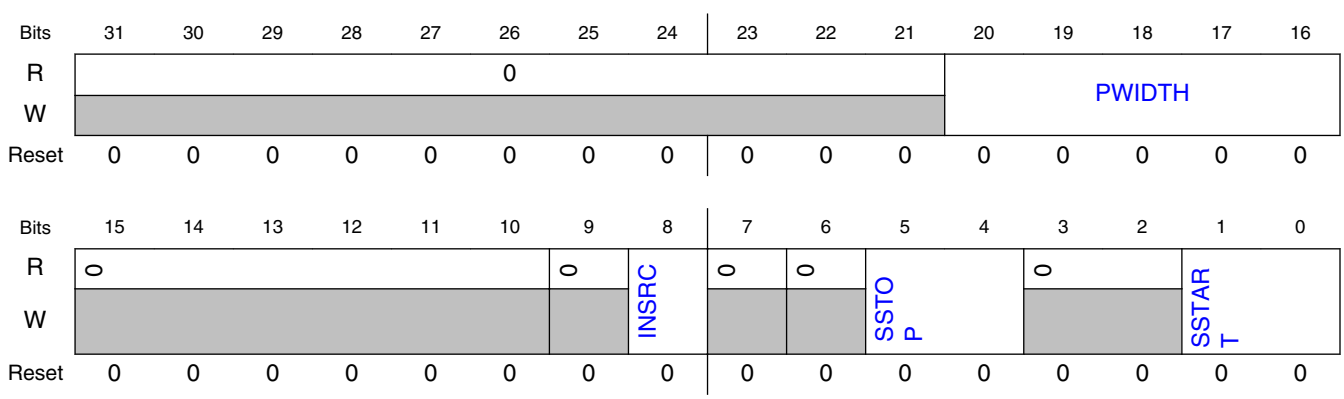
## 58.3.1.15 Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)

### 58.3.1.15.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTCFGa | 100h + (a × 4h) |

### 58.3.1.15.2 Diagram




### 58.3.1.15.3 Fields

| Field | Function |
|---|---|
| 31-21 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 20-16<br>PWIDTH | Parallel Width<br>For all Shifters, this register field configures the number of bits to be shifted on each Shift clock as follows:<br>1-bit shift for PWIDTH=0<br>4-bit shift for PWIDTH=1...3<br>8-bit shift for PWIDTH=4...7<br>16-bit shift for PWIDTH=8...15<br>32-bit shift for PWIDTH=16...31<br>For Shifters which support parallel transmit (SHIFTER0, SHIFTER4) or parallel receive (SHIFTER3, SHIFTER7), this register field, together with SHIFTCTL[PINSEL], also selects the pins to be driven or sampled on each Shift clock as follows:<br>FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]<br>For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail. |
| 15-10<br>— | Reserved. |
| 9<br>— | Reserved. |
| 8<br>INSRC | Input Source<br>Selects the input source for the shifter.<br>0b - Pin<br>1b - Shifter N+1 Output |
| 7<br>— | Reserved. |
| 6<br>— | Reserved. |
| 5-4<br>SSTOP | Shifter Stop bit<br>For SMOD=Transmit, this field allows automatic stop bit insertion if the selected timer has also enabled a stop bit.<br>For SMOD=Receive or Match Store, this field allows automatic stop bit checking if the selected timer has also enabled a stop bit.<br>For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail.<br>For SMOD=Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail.<br>00b - Stop bit disabled for transmitter/receiver/match store<br>01b - Reserved for transmitter/receiver/match store<br>10b - Transmitter outputs stop bit value 0 on store, receiver/match store sets error flag if stop bit is not 0<br>11b - Transmitter outputs stop bit value 1 on store, receiver/match store sets error flag if stop bit is not 1 |
| 3-2<br>— | Reserved. |
| 1-0<br>SSTART | Shifter Start bit |

| Field | Function |
|---|---|
| | For SMOD=Transmit, this field allows automatic start bit insertion if the selected timer has also enabled a start bit.<br>For SMOD=Receive or Match Store, this field allows automatic start bit checking if the selected timer has also enabled a start bit.<br>For SMOD=State, this field is used to disable state outputs. See 'State Mode' section for more detail.<br>For SMOD=Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail.<br>00b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on enable<br>01b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on first shift<br>10b - Transmitter outputs start bit value 0 before loading data on first shift, receiver/match store sets error flag if start bit is not 0<br>11b - Transmitter outputs start bit value 1 before loading data on first shift, receiver/match store sets error flag if start bit is not 1 |

## 58.3.1.16 Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)

### 58.3.1.16.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFa | 200h + (a × 4h) |

### 58.3.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUF | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUF | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.16.3 Fields

| Field | Function |
|---|---|
| 31-0 | Shift Buffer |

| Field | Function |
|---|---|
| SHIFTBUF | Shift buffer data is used for a variety of functions depending on the SMOD setting:<br>For SMOD=Receive, Shifter data is transferred into SHIFTBUF at the expiration of Timer. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available.<br>For SMOD=Transmit, SHIFTBUF data is transferred into the Shifter before the Timer begins.<br>For SMOD=Match Store, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). The Match is checked when the Timer expires and Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available.<br>For SMOD=Match Continuous, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask).<br>For SMOD=Logic, SHIFTBUF[31:0] is used to implement a 5-input, 32-bit programmable logic look-up table. See 'Logic Mode' section for more detail.<br>For SMOD=State, SHIFTBUF[31:24] is used to drive the output value when this shifter is selected by the current state pointer and SHIFTBUF[23:0] is used to configure the value of the next state transition. See 'State Mode' section for more detail. |

## 58.3.1.17 Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)

### 58.3.1.17.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBISa | 280h + (a × 4h) |

### 58.3.1.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 58.3.1.17.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBIS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are bit swapped. Reads return SHIFTBUF[0:31]. |

### 58.3.1.18 Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)

#### 58.3.1.18.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBYSa | 300h + (a × 4h) |

#### 58.3.1.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBYS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBYS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 58.3.1.18.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBYS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are byte swapped. Reads return { SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24] }. |

## 58.3.1.19 Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)

### 58.3.1.19.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBBSa | 380h + (a × 4h) |

### 58.3.1.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.19.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBBS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are bit swapped within each byte. Reads return { SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7] }. |

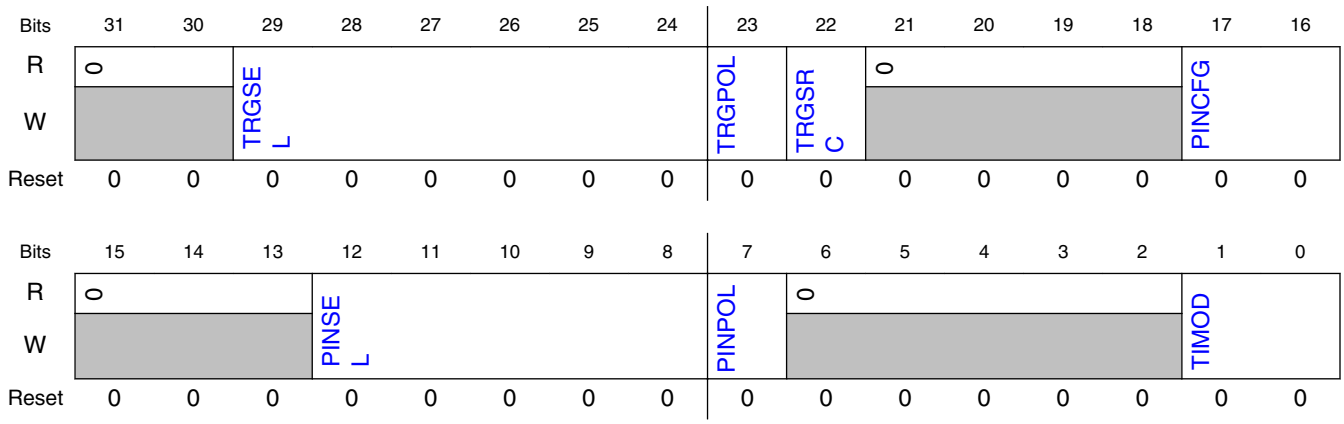
## 58.3.1.20 Timer Control N Register (TIMCTL0 - TIMCTL7)

### 58.3.1.20.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| TIMCTLa | 400h + (a × 4h) |

## 58.3.1.20.2 Diagram




## 58.3.1.20.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-24<br>TRGSEL | Trigger Select<br>The valid values for TRGSEL will depend on the FLEXIO_PARAM register.<br>• When TRGSRC = 1, the valid values for N will depend on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register.<br>• When TRGSRC = 0, the valid values for N will depend on TRIGGER field in FLEXIO_PARAM register.<br><br>Refer to the chip-specific FlexIO information for external trigger selection.<br>**NOTE:** For a pin, N=0 to 31. For a Shifter, N=0 to 7. For a Timer, N=0 to 7.<br>When TRGSRC = 0 the trigger selection is configured as follows:<br>• N - External trigger N input<br><br>When TRGSRC = 1 the internal trigger can be configured to select an input pin as follows:<br>• 2*N - Pin N input<br><br>When TRGSRC = 1 the internal trigger can be configured to select a shifter/timer signal as follows:<br>• 4*N+1 - Shifter N status flag<br>• 4*N+3 - Timer N trigger output<br><br>In other words, the expanded internal trigger selection (TRGSRC = 1) is as follows:<br>• 0000 = Pin 0<br>• 0001 = Shifter 0 Flag<br>• 0010 = Pin 1<br>• 0011 = Timer 0 Trigger |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • 0100 = Pin 2<br>• 0101 = Shifter 1 Flag<br>• 0110 = Pin 3<br>• 0111 = Timer 1 Trigger<br>• ...<br>• This continues up to the Pin 31, Shifter 7 and Timer 7. |
| 23<br>TRGPOL | Trigger Polarity<br>0b - Trigger active high<br>1b - Trigger active low |
| 22<br>TRGSRC | Trigger Source<br>0b - External trigger selected<br>1b - Internal trigger selected |
| 21-18<br>— | Reserved. |
| 17-16<br>PINCFG | Timer Pin Configuration<br>Configures the direction of the Timer pin. For pins configured as an output (PINCFG=11), this field will take effect when the register is written.<br>00b - Timer pin output disabled<br>01b - Timer pin open drain or bidirectional output enable<br>10b - Timer pin bidirectional output data<br>11b - Timer pin output |
| 15-13<br>— | Reserved. |
| 12-8<br>PINSEL | Timer Pin Select<br>Selects which pin is used by the Timer input or output. PINSEL=i will select the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field will take effect when the register is written. |
| 7<br>PINPOL | Timer Pin Polarity<br>For pins configured as an output (PINCFG=11), this field will take effect when the register is written.<br>0b - Pin is active high<br>1b - Pin is active low |
| 6-2<br>— | Reserved. |
| 1-0<br>TIMOD | Timer Mode<br>In 8-bit baud counter mode, the lower 8-bits of the counter and compare register are used to configure the baud rate of the timer shift clock, and the upper 8-bits are used to configure the shifter bit count.<br>In 8-bit PWM high mode, the lower 8-bits of the counter and compare register are used to configure the high period of the timer shift clock, and the upper 8-bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal.<br>In 16-bit counter mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count.<br>00b - Timer Disabled.<br>01b - Dual 8-bit counters baud mode.<br>10b - Dual 8-bit counters PWM high mode.<br>11b - Single 16-bit counter mode. |

## 58.3.1.21 Timer Configuration N Register (TIMCFG0 - TIMCFG7)

### 58.3.1.21.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| TIMCFGa | 480h + (a × 4h) |

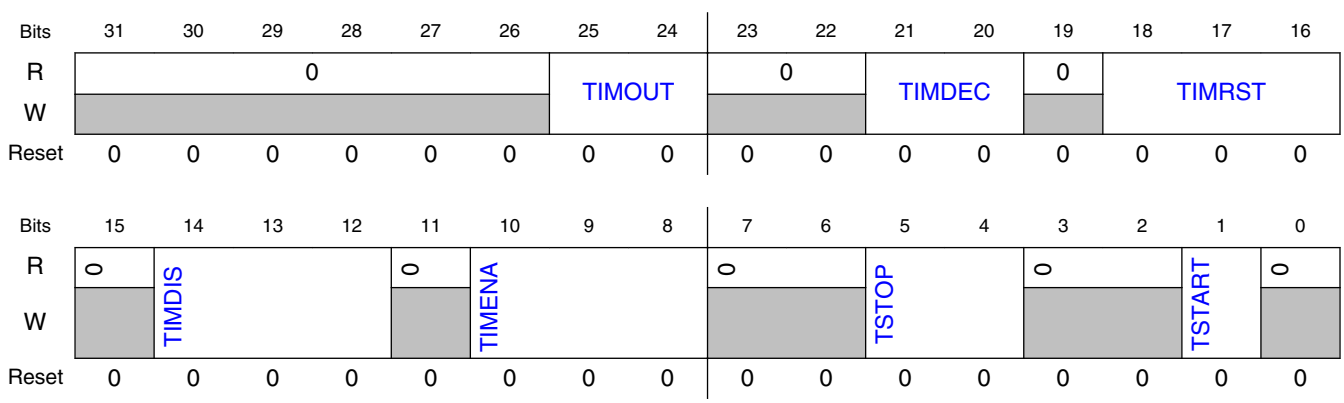
### 58.3.1.21.2 Function

The options to enable or disable the timer using the Timer N-1 enable or disable are reserved when N is evenly divisible by 4 (eg: Timer 0).

**NOTE**

The pin and trigger level/edges specified below refer to the signal state after being modified by the PINPOL or TRGPOL setting in the TIMCTL register. For example, "Trigger low" means a trigger is actually at logic level 1 if the TRGPOL is set to 1 (active low).

### 58.3.1.21.3 Diagram




### 58.3.1.21.4 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-24 | Timer Output |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIMOUT | Configures the initial state of the Timer Output and whether it is affected by the Timer reset.<br>00b - Timer output is logic one when enabled and is not affected by timer reset<br>01b - Timer output is logic zero when enabled and is not affected by timer reset<br>10b - Timer output is logic one when enabled and on timer reset<br>11b - Timer output is logic zero when enabled and on timer reset |
| 23-22<br>— | Reserved. |
| 21-20<br>TIMDEC | Timer Decrement<br>Configures the source of the Timer decrement and the source of the Shift clock.<br>00b - Decrement counter on FlexIO clock, Shift clock equals Timer output.<br>01b - Decrement counter on Trigger input (both edges), Shift clock equals Timer output.<br>10b - Decrement counter on Pin input (both edges), Shift clock equals Pin input.<br>11b - Decrement counter on Trigger input (both edges), Shift clock equals Trigger input. |
| 19<br>— | Reserved. |
| 18-16<br>TIMRST | Timer Reset<br>Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset will only reset the lower 8-bits that configure the baud rate. In all other modes, the timer reset will reset the full 16-bits of the counter.<br>000b - Timer never reset<br>001b - Reserved<br>010b - Timer reset on Timer Pin equal to Timer Output<br>011b - Timer reset on Timer Trigger equal to Timer Output<br>100b - Timer reset on Timer Pin rising edge<br>101b - Reserved<br>110b - Timer reset on Trigger rising edge<br>111b - Timer reset on Trigger rising or falling edge |
| 15<br>— | Reserved. |
| 14-12<br>TIMDIS | Timer Disable<br>Configures the condition that causes the Timer to be disabled and stop decrementing.<br>000b - Timer never disabled<br>001b - Timer disabled on Timer N-1 disable<br>010b - Timer disabled on Timer compare (upper 8-bits match and decrement)<br>011b - Timer disabled on Timer compare (upper 8-bits match and decrement) and Trigger Low<br>100b - Timer disabled on Pin rising or falling edge<br>101b - Timer disabled on Pin rising or falling edge provided Trigger is high<br>110b - Timer disabled on Trigger falling edge<br>111b - Reserved |
| 11<br>— | Reserved. |
| 10-8<br>TIMENA | Timer Enable<br>Configures the condition that causes the Timer to be enabled and start decrementing.<br>000b - Timer always enabled<br>001b - Timer enabled on Timer N-1 enable<br>010b - Timer enabled on Trigger high<br>011b - Timer enabled on Trigger high and Pin high<br>100b - Timer enabled on Pin rising edge<br>101b - Timer enabled on Pin rising edge and Trigger high<br>110b - Timer enabled on Trigger rising edge |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 111b - Timer enabled on Trigger rising or falling edge |
| 7-6<br>— | Reserved. |
| 5-4<br>TSTOP | Timer Stop Bit<br>The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters will output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable.<br>00b - Stop bit disabled<br>01b - Stop bit is enabled on timer compare<br>10b - Stop bit is enabled on timer disable<br>11b - Stop bit is enabled on timer compare and timer disable |
| 3-2<br>— | Reserved. |
| 1<br>TSTART | Timer Start Bit<br>When start bit is enabled, configured shifters will output the contents of the start bit when the timer is enabled and the timer counter will reload from the compare register on the first rising edge of the shift clock.<br>0b - Start bit disabled<br>1b - Start bit enabled |
| 0<br>— | Reserved. |

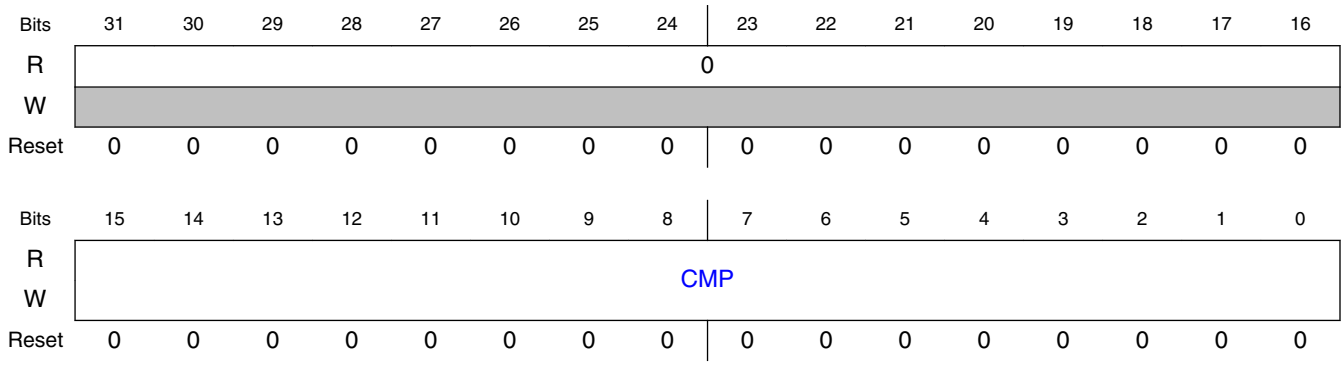
## 58.3.1.22 Timer Compare N Register (TIMCMP0 - TIMCMP7)

### 58.3.1.22.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| TIMCMPa | 500h + (a × 4h) |

### 58.3.1.22.2 Diagram




### 58.3.1.22.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-0<br>CMP | Timer Compare Value<br>The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset and when the timer decrements down to zero.<br>In 8-bit baud counter mode, the lower 8-bits configure the baud rate divider equal to (CMP[7:0] + 1) * 2. The upper 8-bits configure the number of bits in each word equal to (CMP[15:8] + 1) / 2.<br>In 8-bit PWM high mode, the lower 8-bits configure the high period of the output to (CMP[7:0] + 1) and the upper 8-bits configure the low period of the output to (CMP[15:8] + 1).<br>In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal (CMP[15:0] + 1) * 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word equal to (CMP[15:0] + 1) / 2. |

## 58.3.1.23 Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)

### 58.3.1.23.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFNBSa | 680h + (a × 4h) |

### 58.3.1.23.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.23.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFNBS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped within each byte. Reads return { SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4] }. |

## 58.3.1.24 Shifter Buffer N Half Word Swapped Register (SHIFTBUF HWS0 - SHIFTBUFHWS7)

### 58.3.1.24.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFHWSa | 700h + (a × 4h) |

### 58.3.1.24.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHWS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHWS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.24.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFHWS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are half word swapped. Reads return { SHIFTBUF[15:0], SHIFTBUF[31:16] }. |

## 58.3.1.25 Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)

### 58.3.1.25.1 Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFNISa | 780h + (a × 4h) |

### 58.3.1.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 58.3.1.25.3 Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFNIS | Shift Buffer<br>Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped. Reads return { SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28] }. |

# 58.4 Functional description

## 58.4.1 Clocking and Resets

### 58.4.1.1 Functional clock

The FlexIO functional clock is asynchronous to the bus clock and can remain enabled in low power modes. The FlexIO functional clock must be enabled before accessing any FlexIO registers. Provided the FlexIO functional clock is at least two times faster than the bus clock, the CTRL[FASTACC] bit can be set to support fast register accesses.

### 58.4.1.2 Bus clock

The bus clock is only used for bus accesses to the control and configuration registers.

### 58.4.1.3 Chip reset

The logic and registers for the FlexIO are reset to their default state on a chip reset.

### 58.4.1.4 Software reset

The FlexIO implements a software reset bit in its Control Register. The CTRL[RST] will reset all logic and registers to their default state, except for the CTRL itself.

## 58.4.2 Shifter operation

Shifters are responsible for buffering and shifting data into or out of the FlexIO. The timing of shift, load and store events are controlled by the Timer assigned to the Shifter via the SHIFTCTL[TIMSEL] register. The Shifters are designed to support either DMA, interrupt or polled operation. The following block diagram provides a detailed view of the Shifter microarchitecture.

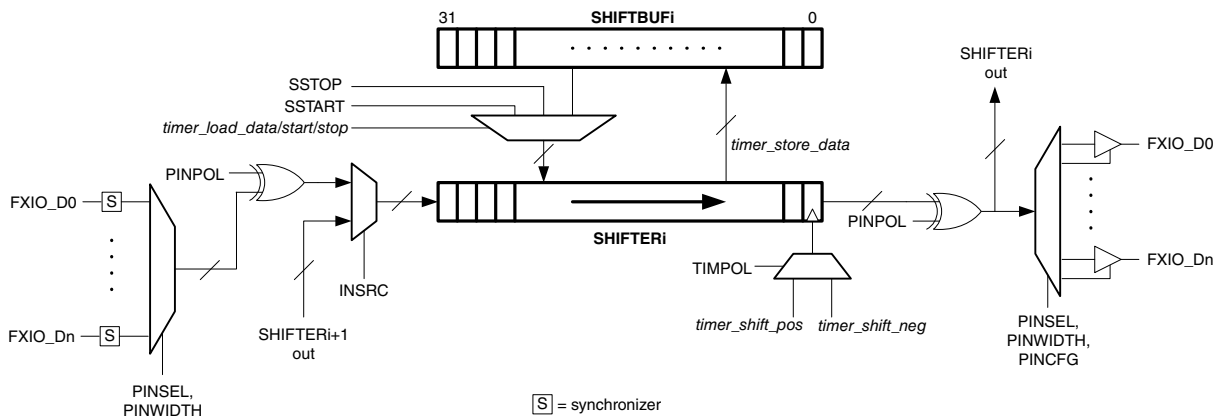



**Figure 58-2. Shifter Microarchitecture**

### 58.4.2.1 Transmit Mode

When configured for Transmit mode (SHIFTCTL[SMOD]=Transmit), the shifter will load data from the SHIFTBUF register and shift data out when a load event is signalled by the assigned Timer. An optional start/stop bit can also be automatically loaded before/after SHIFTBUF data by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Note that the shifter will immediately load a stop bit when the Shifter is initially configured for Transmit mode if a stop bit is enabled.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been loaded from the SHIFTBUF register into the Shifter or when the Shifter is initially configured into Transmit mode. The flag will clear when new data has been written into the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). The flag can be cleared by writing it with logic 1.

### 58.4.2.2 Receive Mode

When configured for Receive mode (SHIFTCTL[SMOD]=Receive), the shifter will shift data in and store data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to store data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

### 58.4.2.3 Match Store Mode

When configured for Match Store mode (SHIFTCTL[SMOD]=Match Store), the shifter will shift data in, check for a match result and store matched data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs and matched data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the matched data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

### 58.4.2.4 Match Continuous Mode

When configured for Match Continuous mode (SHIFTCTL[SMOD]=Match Continuous), the shifter will shift data in and continuously check for a match result whenever a shift event is signalled by the assigned Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs. The flag will clear automatically as soon as there is no longer a match between Shifter data and SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when a match occurs. The flag will clear when there is a read from the SHIFTBUF register or it written with logic 1.

### 58.4.2.5 State Mode

Using State mode enables the user to implement any state machine with up to 8 states, 8 outputs and 3 selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU, which could potentially remain in a STOP/VLPS low power mode.

When configured for State mode (SHIFTCTL[SMOD]=State), the SHIFTBUF register is used to drive the output and compute next state values when the Shifter has been selected by the current state pointer (SHIFTSTATE[STATE]). The following diagram provides a detailed view of Shifter microarchitecture when configured for State mode.

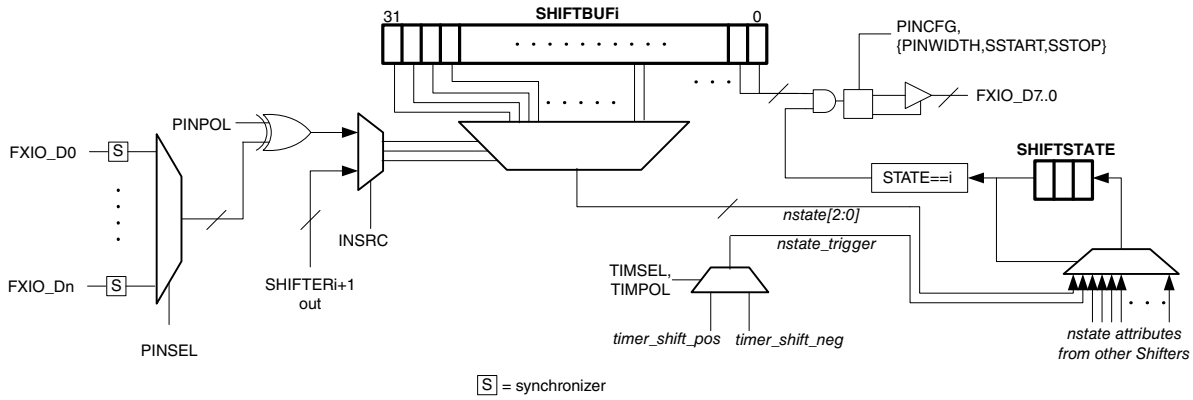



**Figure 58-3. State Microarchitecture**

When Shifter i has been selected by the current state pointer, output pins FXIO_D[7:0] will be driven by SHIFTBUFi[31:24] using the configuration set by SHIFTCTLi[PINCFG]. When set, SHIFTCFGi{PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]} are respectively used to disable the output drive on pins FXIO_D[7:0] for state machine applications which require less than 8 output pins.

The next state value is computed using the 3 input pins selected by SHIFTCTLi[PINSEL] together with SHIFTBUFi[23:0]. Note that each state could potentially use a different set of 3 input pins. The following table details how the next state value is computed when the current state pointer is pointing to Shifter i.

**Table 58-2. Next State computation for SHIFTSTATE[STATE]=i**

| FXIO_D[PINSEL+2] | FXIO_D[PINSEL+1] | FXIO_D[PINSEL] | Next State Value |
|---|---|---|---|
| 0 | 0 | 0 | SHIFTBUFi[2:0] |
| 0 | 0 | 1 | SHIFTBUFi[5:3] |
| 0 | 1 | 0 | SHIFTBUFi[8:6] |
| 0 | 1 | 1 | SHIFTBUFi[11:9] |
| ... | ... | ... | ... |
| 1 | 1 | 1 | SHIFTBUFi[23:21] |

Note that other shifters/timers could potentially be configured to drive the input pins of a given state, allowing the user to create complex combinations of shifters/timers as desired e.g. the output of a Shifter configured for logic mode could potentially be used to drive a state machine input.

The next state transition is triggered using the Timer output selected by SHIFTCTLi[TIMSEL] with polarity controlled by SHIFTCTLi[TIMPOL]. Note that each state could potentially use a different Timer to trigger each next state transition, allowing a variety of internal/external trigger sources and clocking configurations to be used (see Timer section for more detail).

The current state pointer defaults to Shifter 0 at reset, however it can be written by the user to select a different Shifter for the initial state. If the current state pointer selects a Shifter which is not configured for State mode, then outputs will not be driven and a next state transition is never triggered.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set whenever the Shifter has been selected by the current state pointer. The flag will clear when the current state pointer is updated to a different Shifter.

### 58.4.2.6 Logic Mode

Using logic mode enables the user to implement a small amount of programmable digital logic within a FlexIO Shifter.

When configured for Logic mode (SHIFTCTL[SMOD]=Logic), the SHIFTBUF register is used to implement a 5-input, 32-bit programmable logic look-up table. The following diagram provides a detailed view of Shifter microarchitecture when configured for Logic mode.

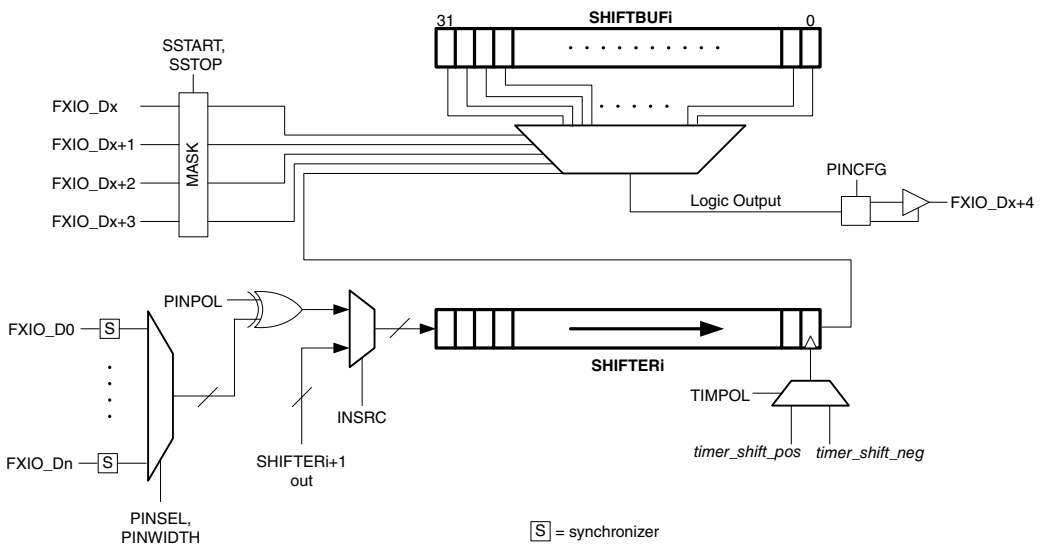



**Figure 58-4. Logic Microarchitecture**

The look-up table is driven using 4 pin inputs (maskable using SHIFTCFG[SSTOP] and SHFITCFG[SSTART]) plus 1 input from the internal shifter and can be configured to drive an output pin using the SHIFTCTL[PINCFG] field. Pin inputs and outputs are fixed for each logic look-up table and are not selectable. The following table lists the logic output value selected by the look-up table for Shifter 'i'.

**Table 58-3. Logic Look-up table for Shifter 'i'**

| SHIFTERi[0] | FXIO_D[x+3][1] | FXIO_D[x+2] | FXIO_D[x+1] | FXIO_D[x] | Logic Output to FXIO_D[x+4] |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | SHIFTBUFi[0] |
| 0 | 0 | 0 | 0 | 1 | SHIFTBUFi[1] |
| 0 | 0 | 0 | 1 | 0 | SHIFTBUFi[2] |
| 0 | 0 | 0 | 1 | 1 | SHIFTBUFi[3] |
| ... | ... | ... | ... | ... | ... |
| 1 | 1 | 1 | 1 | 1 | SHIFTBUFi[31] |

1. for Shifter i=0...3, x=i

   for Shifter i=4...7, x=i+4

To minimize output glitches, SHIFTCFG[SSTOP] and SHIFTCFG[SSTART] can be used to mask unused input pins. When set, {SSTOP[1:0], SSTART[1:0]} will mask FXIO_D[x+3]...FXIO_D[x] inputs respectively, so that any transitions on these pins will not cause the logic output to glitch.

Note that other shifters/timers could potentially be configured to drive the input pins of a given look-up table, allowing the user to concatenate look-up tables or create complex combinations of shifters/timers as desired.

SHIFTCFG[PWIDTH] will control the number of delay stages introduced by the internal shifter input (SHIFTERi[0]). For example, when configured for 1-bit shift (PWIDTH=0), the internal shifter will introduce a 32 Shift clock delay before passing its input (selected by SHIFTCTL[PINSEL]) to the look-up table. When configured for 32-bit shift (PWIDTH=16...31), the internal shifter will introduce a 1 Shift clock delay to its input.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set whenever the output pin allocated to the logic look-up table has a value of 1 (after being synchronized to the FlexIO clock). The flag will clear when the output pin has a value of 0. This also allows the SSF flag to be used as a trigger to a Timer if desired.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when the output pin allocated to the logic look-up table has a value of 1. The flag can be cleared by writing it with logic 1.

## 58.4.3 Timer Operation

The FlexIO 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock. They can perform generic timer functions such as generating a clock or select output or a PWM waveform. Timers can be configured to enable in response to a

trigger, pin or shifter condition; decrement always or only on a trigger or pin edge; reset in response to a trigger or pin condition; and disable on a trigger or pin condition or on a timer compare. Timers can optionally include a start condition and/or stop condition.

Each timer operates independently, although a timer can be configured to enable or disable at the same time as the previous timer (eg: timer1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently and can be configured to be a timer output, shifter status flag, pin input or an external trigger input (refer to the chip-specific FlexIO information for details on the external trigger connections). The trigger configuration is separate from the pin configuration, which can be configured for input, output data or output enable.

The Timer Configuration Register (TIMCFGn) should be configured before setting the Timer Mode (TIMOD).

#### 58.4.3.1 Timer 8-bit Baud Counter Mode

In 8-bit Baud Counter Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the baud rate of the shift clock and the upper 8-bits are used to configure the number of shift clock edges in the transfer. When the lower 8-bits decrement to zero, the timer output is toggled and the lower 8-bits reload from the compare register. The upper 8-bits only decrement when the lower 8-bits equal zero and decrement.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

#### 58.4.3.2 Timer 8-bit High PWM Mode

In 8-bit High PWM Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the timer output high period and the upper 8-bits are used to configure the timer output low period. The lower 8-bits decrement when the output is high. When the lower 8-bits equal zero and decrement, the timer output is cleared and the lower 8-bits are reloaded from the compare register. The upper 8-bits decrement when the output is low. When the upper 8-bits equal zero and decrement, the timer output is set and the upper 8-bits are reloaded from the compare register.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

### 58.4.3.3 Timer 16-bit Counter Mode

In 16-bit Counter Mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (eg: TIMDEC[1:0] != 10 or 11) or the number of shift clock edges in the transfer (eg: TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer compare event.

### 58.4.3.4 Timer Enable and Start Bit

When the TIMOD is configured for the desired mode, and the condition configured by timer enable (TIMENA) is detected then the following events occur.

- Timer counter will load the current value of the compare Register and start decrementing as configured by TIMDEC.
- Timer output may update to its initial state depending on the TIMOUT configuration. Shifters that are controlled by this timer do not see this as a rising edge on the timer shift clock.
- Transmit shifters controlled by this timer will either output their start bit value, or load the shift register from the shift buffer and output the first bit, as configured by SSTART.

If the Timer start bit is enabled, the timer counter will reload with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMOUT=1), a shifter that is configured to shift on falling edge and load on the first shift will not load correctly.

### 58.4.3.5 Timer Decrement and Reset

The Timer will then generate the timer output and timer shift clock depending on the TIMOD and TIMDEC fields. The shifter clock is either equal to the timer output (when TIMDEC != 10 or 11) or equal to the decrement clock (when TIMDEC = 10 or 11). When TIMDEC is configured to decrement from a pin or trigger, the timer will decrement on both rising and falling edges.

When the Timer is configured to reset as configured in the TIMRST field then the Timer counter will load the current value of the compare register again, the timer output may also be affected by the reset as configured in TIMOUT.

In general, when the timer counter decrements to zero a timer compare event is triggered. The timer compare event will cause the timer counter to load the contents of the timer compare register, the timer output to toggle, any configured transmit shift registers to load, and any configured receive shift registers to store. Depending on the timer mode, the timer status flag may also be set.

### 58.4.3.6 Timer Disable and Stop Bit

When the is Timer is configured to add a stop bit on each compare, the following additional events will occur.

- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.
- On the first rising edge of the shifter clock after the compare, the timer counter will reload the current value of the Compare Register.

Transmit shifters must be configured to load on the first shift when the timer is configured to insert a stop bit on each compare.

When the condition configured by timer disable (TIMDIS) is detected, the following events occur.

- Timer counter will reload the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output will clear. Shifters that are controlled by this timer do not see this as a falling edge on the timer shift clock, but can generate a shift event if the timer shift clock would otherwise generate one.
- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.

If the timer stop bit is enabled, the timer counter will continue decrementing until the next rising edge of the shift clock is detected, at which point it will finish. Although the timer output is forced low during the stop bit, the timer shift clock can toggle during the stop bit, but does not generate shift events.

A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). Receive shift registers with stop bit enabled will store the contents of the shift register into the shift buffer and verify the state of the input data

on the configured shift edge while the timer is in the stop state condition. If there is no configured edge between the timer disable and the next rising edge of the shift clock then the final store and verify do not occur.

## 58.4.4 Pin operation

The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity. Each timer and shifter can be configured as an input, output data, output enable or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity, since the output enable assertion would cause logic zero to be output on the pin) or to control the enable on the bidirectional output. Any timer or shifter could be configured to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

### 58.4.4.1 Parallel Interface

Shifters can be configured to use multiple FlexIO pins in parallel using the SHIFTCFG[PWIDTH] field. PWIDTH is used to configure the following settings of a shifter:

1. Number of bits shifted per Shift clock.
2. Number of pins driven by the shifter per Shift clock (only on shifters supporting parallel transmit i.e. SHIFTER0, SHIFTER4).
3. Number of pins sampled by the shifter per Shift clock (only on Shifter supporting parallel receive i.e. SHIFTER3, SHIFTER7).

When configured for parallel shift, either 4, 8, 16 or 32-bits can be shifted on every Shift clock. If an adjacent shifter is selected as the input source (SHIFTCFG[INSRC]=1), the least significant 4, 8, 16 or 32-bits from the adjacent shifter will be sampled on each Shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), the shifter can be configured to sample multiple pins (SHIFTCFG[INSRC]=0), with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]. Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant bits will be masked with 0 e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[31:24] will sample {0,0,FXIO_D[12:7]} on each Shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), the shifter can be configured to drive multiple pins using SHIFTCTL[PINCFG], with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant pins will not be driven e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[5:0] will drive only FXIO_D[12:7] on each Shift clock.

#### 58.4.4.2 Pin Synchronization

When configuring a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized to the FlexIO clock before the signal is used by a timer or shifter. This introduces a small latency of between 0.5 to 1.5 FlexIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FlexIO clock cycles.

If an input is used by more than one timer or shifter then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

Note that FlexIO pins are also connected internally, configuring a FlexIO shifter or timer to output data on an unused pin will make an internal connection that allows other shifters and timer to use this pin as an input. This allows a shifter output to be used to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized to the FlexIO clock and therefore incurs a 1 cycle latency.

So when using a Pin input as a Timer Trigger, Timer Clock or Shifter Data Input, the following synchronization delays occur:

1. 0.5 – 1.5 FlexIO clock cycles for external pin
2. 1 FlexIO clock cycle for an internally driven pin

For timing considerations such as output valid time and input setup time for specific applications (SPI Master, SPI Slave, I2C Master, I2S Master, I2S Slave) please refer to the FlexIO Application Information Section.

### 58.4.5 Interrupts and DMA Requests

The following table illustrates the status flags that can generate the FlexIO interrupt and DMA requests.

**Table 58-4. FlexIO Interrupts and DMA Requests**

| Flag | Description | Interrupt | DMA Request | Low Power Wakeup |
|---|---|---|---|---|
| SSF | Shifter Status Flag. | Y | Y | Y |
| SEF | Shifter Error Flag. | Y | N | Y |
| TSF | Timer Status Flag. | Y | N | Y |

### 58.4.6 Peripheral Triggers

The connection of the FlexIO peripheral triggers with other peripherals are device specific.

#### 58.4.6.1 Output Triggers

Each FlexIO Timer generates an output trigger equal to the timer output. The output trigger is not affected by the timer pin polarity configuration.

#### 58.4.6.2 Input Trigger

FlexIO supports multiple external trigger inputs that can be used to trigger one or more FlexIO timers. The external triggers are synchronized to the FlexIO functional clock and must assert for at least two cycles of the FlexIO functional clock to be sampled correctly.

## 58.5 Application Information

This section provides examples for a variety of FlexIO module applications.

### 58.5.1 UART Transmit

UART transmit can be supported using one Timer, one Shifter and one Pin (two Pins if supporting CTS). The start and stop bit insertion is handled automatically and multiple transfers can be supported using DMA controller. The timer status flag can be used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention, before transmitting a break or idle character the SSTART and SSTOP fields should be altered to transmit the required state and the data to transmit must equal 0xFF or 0x00. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). Note that when performing byte writes to SHIFTBUFn (or SHIFTBUFBIS for transmitting MSB first), the rest of the register remains unaltered allowing an address mark bit or additional stop bit to remain undisturbed.

FlexIO does not support automatic insertion of parity bits.

**Table 58-5. UART Transmit Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0032 | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTLn | 0x0003_0002 | Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL, or can support open drain by setting PINPOL=0x1 and PINCFG=0x1. |
| TIMCMPn | 0x0000_0F01 | Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFGn | 0x0000_2222 | Configure start bit, stop bit, enable on trigger asserted and disable on compare. Can support CTS by configuring TIMEN=0x3. |
| TIMCTLn | 0x01C0_0001 | Configure dual 8-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL=0x1 (for Pin 1) and PINPOL=0x1. |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS[7:0] register instead. |

## 58.5.2 UART Receive

UART receive can be supported using one Timer, one Shifter and one Pin (two Timers and two Pins if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers can be supported using the DMA controller. The timer status flag can be used to indicate when the stop bit of each word is received.

Triple voting of the received data is not supported by FlexIO, data is sampled only once in the middle of each bit. Another timer can be used to implement a glitch filter on the incoming data, another Timer can also be used to detect an idle line of programmable length. Break characters will cause the error flag to set and the shifter buffer register will return 0x00.

FlexIO does not support automatic verification of parity bits.

**Table 58-6. UART Receiver Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0032 | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTLn | 0x0080_0001 | Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL. |
| TIMCMPn | 0x0000_0F01 | Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFGn | 0x0204_2422 | Configure start bit, stop bit, enable on pin posedge and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4. |
| TIMCTLn | 0x0000_0081 | Configure dual 8-bit counter using inverted Pin 0 input. |
| SHIFTBUFn | Data to receive | Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead. |

The UART Receiver with RTS configuration uses a 2nd Timer to generate the RTS output. The RTS will assert when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit will be detected while the RTS is asserted, the received data is simply ignored.

**Table 58-7. UART Receiver with RTS Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0032 | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTLn | 0x0080_0001 | Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL. |
| TIMCMPn | 0x0000_0F01 | Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFGn | 0x0204_2522 | Configure start bit, stop bit, enable on pin posedge with trigger asserted and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4. |

*Table continues on the next page...*

**Table 58-7. UART Receiver with RTS Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| TIMCTLn | 0x02C0_0081 | Configure dual 8-bit counter using inverted Pin 0 input. Trigger is internal using inverted Pin 1 input. |
| TIMCMP(n+1) | 0x0000_FFFF | Never compare. |
| TIMCFG(n+1) | 0x0030_6100 | Enable on Timer N enable and disable on trigger falling edge. Decrement on trigger to ensure no compare. |
| TIMCTL(n+1) | 0x0143_0003 | Configure 16-bit counter and output on Pin 1. Trigger is internal using Shifter 0 flag. |
| SHIFTBUFn | Data to receive | Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead. |

## 58.5.3 SPI Master

SPI master mode can be supported using two Timers, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of 1 clock cycle between the slave select negating and before the next transfer. Writing to the transmit buffer by either core or DMA is used to initiate each transfer.

Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

**Table 58-8. SPI Master (CPHA=0) Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTLn | 0x0083_0002 | Configure transmit using Timer 0 on negedge of clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0000_0101 | Configure receive using Timer 0 on posedge of clock with input data on Pin 1. |

*Table continues on the next page...*

## Table 58-8. SPI Master (CPHA=0) Configuration (continued)

| Register | Value | Comments |
|---|---|---|
| TIMCMPn | 0x0000_3F01 | Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFGn | 0x0100_2222 | Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. |
| TIMCTLn | 0x01C3_0201 | Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock. |
| TIMCMP(n+1) | 0x0000_FFFF | Never compare. |
| TIMCFG(n+1) | 0x0000_1100 | Enable when Timer 0 is enabled and disable when Timer 0 is disabled. |
| TIMCTL(n+1) | 0x0003_0383 | Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select). |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead. |

## Table 58-9. SPI Master (CPHA=1) Configuration

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0021 | Start bit loads data on first shift. |
| SHIFTCTLn | 0x0003_0002 | Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0080_0101 | Configure receive using Timer 0 on negedge of clock with input data on Pin 1. |
| TIMCMPn | 0x0000_3F01 | Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |

*Table continues on the next page...*

**Table 58-9. SPI Master (CPHA=1) Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| TIMCFGn | 0x0100_2222 | Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. |
| TIMCTLn | 0x01C3_0201 | Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock. Set TIMDIS=3 to keep slave select asserted for as long as there is data in the transmit buffer. |
| TIMCMP(n+1) | 0x0000_FFFF | Never compare. |
| TIMCFG(n+1) | 0x0000_1100 | Enable when Timer 0 is enabled and disable when Timer 0 is disabled. |
| TIMCTL(n+1) | 0x0003_0383 | Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select). |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead. |

## 58.5.4 SPI Slave

SPI slave mode can be supported using one Timer, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The transmit data must be written to the transmit buffer register before the external slave select asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

## Table 58-10. SPI Slave (CPHA=0) Configuration

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTLn | 0x0083_0002 | Configure transmit using Timer 0 on falling edge of shift clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0000_0101 | Configure receive using Timer 0 on rising edge of shift clock with input data on Pin 1. |
| TIMCMPn | 0x0000_003F | Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1. |
| TIMCFGn | 0x0120_6000 | Configure enable on trigger rising edge, initial clock state is logic 0 and decrement on pin input. |
| TIMCTLn | 0x06C0_0203 | Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger. |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead. |

## Table 58-11. SPI Slave (CPHA=1) Configuration

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0001 | Shifter configured to load on first shift and stop bit disabled. |
| SHIFTCTLn | 0x0003_0002 | Configure transmit using Timer 0 on rising edge of shift clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0080_0101 | Configure receive using Timer 0 on falling edge of shift clock with input data on Pin 1. |
| TIMCMPn | 0x0000_003F | Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1. |
| TIMCFGn | 0x0120_6602 | Configure start bit, enable on trigger rising edge, disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input. |

*Table continues on the next page...*

**Table 58-11. SPI Slave (CPHA=1) Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| TIMCTLn | 0x06C0_0203 | Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger. |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead. |

## 58.5.5 I2C Master

I2C master mode can be supported using two Timers, two Shifters and two Pins. One timer is used to generate the SCL output and one timer is used to control the shifters. The two shifters are used to transmit and receive for every word, when receiving the transmitter must transmit 0xFF to tristate the output. FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK. FlexIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (START to Repeated START/ STOP), so the compare register needs to be programmed with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin equal to output (although this increases both the clock high and clock low periods by at least 1 FlexIO clock cycle each). The second timer uses the SCL input pin to control the transmit/receive shift registers, this enforces an SDA data hold time by an extra 2 FlexIO clock cycles.

Both the transmit and receive shifters need to be serviced for each word in the transfer, the transmit shifter must transmit 0xFF when receiving and the receive shifter returns the data actually present on the SDA pin. The transmit shifter will load 1 additional word on the last falling edge of SCL pin, this word should be 0x00 if generating a STOP condition or 0xFF if generating a repeated START condition. During the last word of a master-receiver transfer, the transmit SSTOP bit should be set by software to generate a NACK.

The receive shift register will assert an error interrupt if a NACK is detected, but software is responsible for generating the STOP or repeated START condition. If a NACK is detected during master-transmit, the interrupt routine should immediately write the transmit shifter register with 0x00 (if generating STOP) or 0xFF (if generating repeated START). Software should then wait for the next rising edge on SCL and then disable both timers. The transmit shifter should then be disabled after waiting the setup delay for a repeated START or STOP condition.

Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The I2C master data valid is delayed 2 cycles because the clock output is passed through a synchronizer before clocking the transmit/receive shifter (to guarantee some SDA hold time). Since the SCL output is synchronous with FlexIO clock, the synchronization delay is 1 cycle and then 1 cycle to generate the output.

**Table 58-12. I2C Master Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0032 | Start bit enabled (logic 0) and stop bit enabled (logic 1). |
| SHIFTCTLn | 0x0101_0082 | Configure transmit using Timer 1 on rising edge of clock with inverted output enable (open drain output) on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0020 | Start bit disabled and stop bit enabled (logic 0) for ACK/NACK detection. |
| SHIFTCTL(n+1) | 0x0180_0001 | Configure receive using Timer 1 on falling edge of clock with input data on Pin 0. |
| TIMCMPn | 0x0000_2501 | Configure 2 word transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of words x 18) + 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFGn | 0x0102_2222 | Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset. |
| TIMCTLn | 0x01C1_0101 | Configure dual 8-bit counter using Pin 1 output enable (SCL open drain), with Shifter 0 flag as the inverted trigger. |
| TIMCMP(n+1) | 0x0000_000F | Configure 8-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1. |
| TIMCFG(n+1) | 0x0020_1112 | Enable when Timer 0 is enabled, disable when Timer 0 is disabled, enable start bit and stop bit at end of each word, decrement on pin input. |
| TIMCTL(n+1) | 0x01C0_0183 | Configure 16-bit counter using inverted Pin 1 input (SCL). |

*Table continues on the next page...*

**Table 58-12. I2C Master Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUFBBS[7:0], use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBIS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. |

## 58.5.6 I2S Master

I2S master mode can be supported using two Timers, two Shifters and four Pins. One timer is used to generate the bit clock and control the shifters and one timer is used to generate the frame sync. FlexIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FlexIO clock frequency, and the initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure the frame sync is generated one clock cycle before the first output data.

Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

**Table 58-13. I2S Master Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0001 | Load transmit data on first shift and stop bit disabled. |
| SHIFTCTLn | 0x0003_0002 | Configure transmit using Timer 0 on rising edge of clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0080_0101 | Configure receive using Timer 0 on falling edge of clock with input data on Pin 1. |
| TIMCMPn | 0x0000_3F01 | Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |

*Table continues on the next page...*

**Table 58-13. I2S Master Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| TIMCFGn | 0x0000_0202 | Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1. |
| TIMCTLn | 0x01C3_0201 | Configure dual 8-bit counter using Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock. |
| TIMCMP(n+1) | 0x0000_007F | Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (number of bits x baud rate divider) - 1. |
| TIMCFG(n+1) | 0x0000_0100 | Enable when Timer 0 is enabled and never disable. |
| TIMCTL(n+1) | 0x0003_0383 | Configure 16-bit counter using inverted Pin 3 output (as frame sync). |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead. |

### 58.5.7 I2S Slave

I2S slave mode can be supported using three Timers, two Shifters and four Pins (for single transmit and single receive, other combinations of transmit and receive are possible).

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The output valid time of I2S slave is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data

Timer 2 detects the falling edge of frame sync (start of new frame) and asserts output until falling edge of bit clock (when frame sync is normally sampled). Timer 0 detects falling edge of bit clock with Timer 2 output asserted and asserts output for length of frame. Timer 1 detects rising edge of bit clock with Timer 0 output asserted and controls shift registers for 32-bit transfers.

**Table 58-14. I2S Slave Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTLn | 0x0103_0002 | Configure transmit using Timer 1 on rising edge of shift clock with output data on Pin 0. |
| SHIFTCFG(n+1) | 0x0000_0000 | Start and stop bit disabled. |
| SHIFTCTL(n+1) | 0x0180_0101 | Configure receive using Timer 1 on falling edge of shift clock with input data on Pin 1. |
| TIMCMPn | 0x0000_007F | Configure two 32-bit transfers per frame. Set TIMCMP[15:0] = (number of bits x 4) - 1. |
| TIMCFGn | 0x0020_2500 | Configure enable on pin rising edge (inverted bit clock) with trigger high (Timer 2) and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock). |
| TIMCTLn | 0x0B40_0283 | Configure 16-bit counter using inverted Pin 2 input (bit clock), with Timer 2 output as the trigger. |
| TIMCMP(n+1) | 0x0000_003F | Configure 32-bit transfers. Set TIMCMP[15:0] = (number of bits x 2) - 1. |
| TIMCFG(n+1) | 0x0020_2500 | Configure enable on pin (bit clock) rising edge with trigger (Timer 0) high and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock). |
| TIMCTL(n+1) | 0x0340_0283 | Configure 16-bit counter using inverted Pin 2 input (bit clock), with Timer 0 output as the trigger. |
| TIMCMP(n+2) | 0x0000_0000 | Compare on zero (first edge). |
| TIMCFG(n+2) | 0x0020_6400 | Configure enable on inverted pin (frame sync) rising edge and disable on trigger falling edge (bit clock), initial clock state is logic 1 and decrement on inverted pin input (frame sync). |
| TIMCTL(n+2) | 0x0440_0383 | Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 input (bit clock) as the trigger. |
| SHIFTBUFn | Data to transmit | Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. |

*Table continues on the next page...*

**Table 58-14. I2S Slave Configuration (continued)**

| Register | Value | Comments |
|---|---|---|
| | | Can support LSB first transfer by writing to SHIFTBUF register instead. |
| SHIFTBUF(n+1) | Data to receive | Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead. |

## 58.5.8 Camera Interface

Camera Interface can be supported using one Timer, one or more Shifters and multiple Pins. Multiple transfers can be supported using DMA controller.

The example below describes FlexIO configuration for interfacing to an 8-bit CMOS sensor with PCLK, VSYNC, HREF and D[7:0] outputs. The example uses a 128-bit buffer to capture 16-pixels of image data before interrupt or DMA transfer, however a bigger or smaller buffer may be used depending on system DMA performance and FlexIO resource usage by other applications. Note that additional timers may be used to track number of pixels per row and number of rows per frame or HREF/VSYNC may be assigned as GPIO interrupts for software tracking.

**Table 58-15. Camera Interface Configuration for 8-bit CMOS sensor**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFGn...n+2[1] | 0x0007_0100 | Configure 8-bit parallel shift in from adjacent shifter. |
| SHIFTCFGn+3 | 0x0007_0000 | Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]). |
| SHIFTCTLn...n+3 | 0x0080_0001 | Configure receive using Timer 0 on negedge of clock. |
| TIMCMPn | 0x0000_001F | Configure 16-pixel (8 bits/pixel x 16 pixels = 128-bits) transfer. Set TIMCMP[15:0] = (number of pixels x 2) - 1. |
| TIMCFGn | 0x0120_6600 | Configure enable on trigger (HREF) rising edge and disable on trigger falling edge, initial Shift clock state is logic 0 and decrement on PCLK input. |
| TIMCTLn | 0x12C0_0803 | Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger. |
| SHIFTBUFn...n+3 | Data to receive | Received data can be read from SHIFTBUFn...n+3, use the Shifter Status |

**Table 58-15. Camera Interface Configuration for 8-bit CMOS sensor**

| Register | Value | Comments |
|---|---|---|
| | | Flag to indicate when data can be read using interrupt or DMA request. |

1. n=0 or 4

## 58.5.9 Motorola 68K/Intel 8080 Bus Interface

The Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by Smart/Asynchronous LCD controllers. In conjunction with GPIO, FlexIO is able to drive these interfaces using one Timer and one Shifter, although additional Shifters could be used to support large transfers via the DMA controller.

The configuration below provides an example of how to drive a 16-bit 68K or 8080 bus. For a 8080 bus, two GPIO are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

**Table 58-16. Motorola 68K/Intel 8080 Write Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFG0...7 | 0x000F_0100 | Configure 16-bit parallel shift in from adjacent shifter. |
| SHIFTCTL0 | 0x0003_0002 | Configure transmit using Timer 0 on posedge of clock with data output to FXIO_D[15:0]. |
| SHIFTCTL1...7 | 0x0000_0002 | Configure transmit using Timer 0 on posedge of clock. |
| TIMCMP0 | 0x0000_0101 (1-beat)<br>0x0000_1F01 (16-beats) | Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFG0 | 0x0000_2200 | Configure enable on trigger high and disable on compare. Timer output high on enable. |
| TIMCTL0 | 0x01C3_1001 (Motorola 68K, 1-beat)<br>0x1DC3_1001 (Motorola 68K, 16-beats)<br>0x01C3_1081 (Intel 8080, 1-beat)<br>0x1DC3_1081 (Intel 8080, 16-beats) | Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, nWR pin for 8080), with Shifter 0 (1-beat) or Shifter 7 (16-beats) flag as the inverted trigger. |
| SHIFTBUF0...7 | Data to transmit | Transmit data can be written to SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats) to initiate a transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. |

**Table 58-17. Motorola 68K/Intel 8080 Read Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFG0...6 | 0x000F_0100 | Configure 16-bit parallel shift in from adjacent shifter. |
| SHIFTCFG7 | 0x000F_0000 | Configure 16-bit parallel shift in from pin. |
| SHIFTCTL0...7 | 0x0080_0001 | Configure receive using Timer 0 on negedge of clock with data input from FXIO_D[15:0]. |
| TIMCMP0 | 0x0000_0101 (1-beat)<br>0x0000_1F01 (16-beats) | Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFG0 | 0x0000_2220 | Configure stop_bit, enable on trigger high and disable on compare. Timer output high on enable. |
| TIMCTL0 | 0x1DC3_1001 (Motorola 68K, 1 beat)<br>0x01C3_1001 (Motorola 68K, 16 beats)<br>0x1DC3_1181 (Intel 8080, 1 beat)<br>0x01C3_1181 (Intel 8080, 16 beats) | Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (nRD pin for 8080), with Shifter 7 flag (1-beat) or Shifter 0 flag (16-beats) as the inverted trigger. |
| SHIFTBUF0...7 | Data received | Received data can be read from SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats), use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. |

In general, any operation to a 68K/8080 bus slave will begin with a register write cycle followed by one or more data read or write cycles. To accomplish this, the following program flow should be used:

1. Configure FlexIO with 1-beat write configuration
2. Configure GPIO to assert nCS, RS pins (and deassert RDWR pin for 68K)
3. Write register index data to SHIFTBUF0[15:0]
4. Configure GPIO to deassert RS pin (and assert RDWR pin for 68K data read)
5. Configure FlexIO with desired read or write configuration (e.g. 1 or 16-beats)
6. Use the Shifter Status Flag to trigger interrupt or DMA driven data transfers to/from SHIFTBUF registers
7. Configure GPIO to deassert nCS pin

## 58.5.10 Low Power State Machine

The configuration below details a hypothetical state machine example to illustrate the flexibility allowed when using Shifter state mode.

In this example, FlexIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted (This assumes comparator is connected to external trigger 15. See the chip-specific FlexIO information for actual FlexIO trigger mappings). Throughout this operation, the CPU can be kept in a STOP/ VLPS mode, by clearing the CTRL[DOZEN] bit and ensuring the FLEXIO_CLK is enabled. The state diagram below shows the states and transitions implemented by this example.

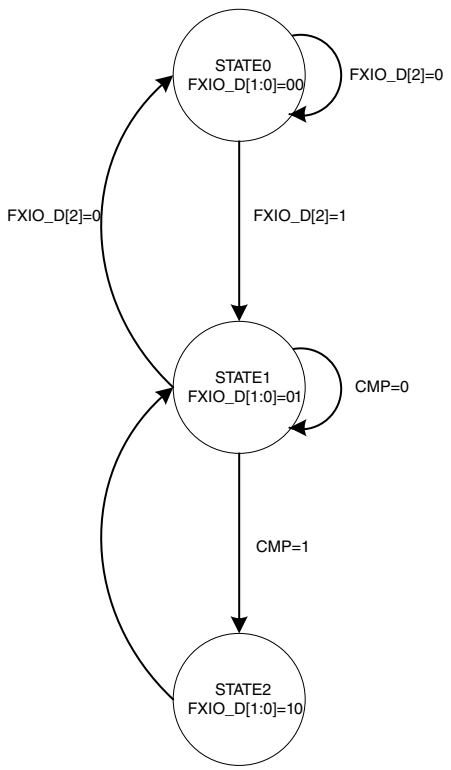



**Figure 58-5. State Diagram**

**Table 58-18. State Machine Configuration**

| Register | Value | Comments |
|---|---|---|
| SHIFTCFG0...2 | 0x0000_0003 | Enable FXIO_D[1:0] as outputs. |
| SHIFTCTL0 | 0x0080_0206 | Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state. |
| SHIFTBUF0 | 0x0020_8208 | State0: Drive FXIO_D[1:0]=00, transition to State0 if FXIO_D[2]=0, State1 if FXIO_D[2]=1. |

*Table continues on the next page...*

## Table 58-18. State Machine Configuration (continued)

| Register | Value | Comments |
|---|---|---|
| SHIFTCTL1 | 0x0000_0206 | Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output high to trigger next state. |
| SHIFTBUF1 | 0x0140_8408 | State1: Drive FXIO_D[1:0]=01, transition to State0 if FXIO_D[2]=0, State1 if CMP=0, State2 if CMP=1 (FXIO_D[3]=1) |
| SHIFTCTL2 | 0x0080_0206 | Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state. |
| SHIFTBUF2 | 0x0224_9249 | State2: Drive FXIO_D[1:0]=10, transition to State1 when Timer0 output low |
| TIMCMP0 | 0x0000_FFFF | Configure baud rate of divide by 131072 of the FlexIO clock. Set TIMCMP[7:0] = (baud rate divider / 2) - 1. |
| TIMCFG0 | 0x0000_0000 | Configure timer always enabled. |
| TIMCTL0 | 0x0000_0003 | Configure single 16-bit counter. |
| TIMCFG1 | 0x0010_7600 | Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge. |
| TIMCTL1 | 0x0F03_0303 | Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected. |

# Chapter 59
# Inter-IC Sound (I2S)

## 59.1 Chip-specific I2S information

### 59.1.1 Instantiation information

This device contains 1 $I^2S$ module.

As configured on the device, module features include:

- TX data lines: 2
- RX data lines: 2
- FIFO size (words): 8 word
- Maximum words per frame: 32
- Maximum bit clock divider: 512

### 59.1.2 $I^2S$/SAI operation in low power modes

#### 59.1.2.1 Stop and very low power modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit is set (TCSR[STOPE] and/or RCSR[STOPE], respectively), and provided the transmitter and/or receiver is/are using an externally generated bit clock or an Audio Master Clock that remains operating in Stop mode. The SAI transmitter and/or receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

In VLPS mode, the module behaves as it does in stop mode if VLPS mode is entered from run mode. However, if VLPS mode is entered from VLPR mode, the FIFO might underflow or overflow before wakeup from stop mode due to the limits in bus bandwidth. In VLPW and VLPR modes, the module is limited by the maximum bus clock frequencies.

When operating from an internally generated bit clock or Audio Master Clock that is disabled in stop modes:

In Stop mode, if the Transmitter Stop Enable (TCSR[STOPE]) bit is clear, the transmitter is disabled after completing the current transmit frame, and, if the Receiver Stop Enable (RCSR[STOPE]) bit is clear, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

#### 59.1.2.2 Low-leakage modes

When entering low-leakage modes, the Stop Enable (TCSR[STOPE] and RCSR[STOPE]) bits are ignored, and the SAI is disabled after completing the current transmit and receive Frames. Entry into stop mode is prevented (not acknowledged) while waiting for the transmitter and receiver to be disabled at the end of the current frame.

## 59.2 Introduction

The $I^2S$ (or I2S) module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as $I^2S$, AC97, TDM, and codec/DSP interfaces.

### 59.2.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

- Transmitter with independent bit clock and frame sync supporting 2 data lines
- Receiver with independent bit clock and frame sync supporting 2 data lines
- Each data line can support a maximum Frame size of 32 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame

- Asynchronous 8 × 32-bit FIFO for each transmit and receive data line
- Supports graceful restart after FIFO error
- Supports automatic restart after FIFO error without software intervention
- Supports packing of 8-bit and 16-bit data into each 32-bit FIFO word
- Supports combining multiple data line FIFOs into single data line FIFO

## 59.2.2 Block diagram

The following block diagram also shows the module clocks.

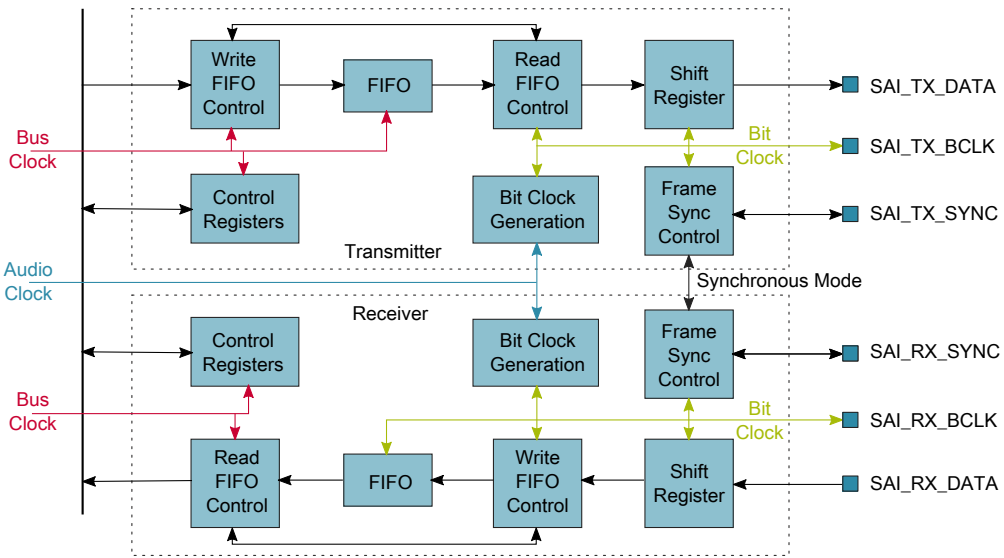



**Figure 59-1. I²S/SAI block diagram**

## 59.2.3 Modes of operation

Module power modes include Run mode, Stop modes, low-leakage stop modes, and Debug mode.

### 59.2.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

### 59.2.3.2 Stop modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit is set (TCSR[STOPE] and/or RCSR[STOPE], respectively), and provided the transmitter and/or receiver is/are using an externally generated bit clock or an Audio Master Clock that remains operating in Stop mode. The SAI transmitter and/or receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

In Stop mode, if the Transmitter Stop Enable (TCSR[STOPE]) bit is clear, the transmitter is disabled after completing the current transmit frame, and, if the Receiver Stop Enable (RCSR[STOPE]) bit is clear, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

### 59.2.3.3 Low-leakage stop modes

When entering low-leakage stop modes, the Stop Enable (TCSR[STOPE] and RCSR[STOPE]) bits are ignored and the SAI is disabled after completing the current transmit and receive Frames. Entry into stop mode is prevented (not acknowledged) while waiting for the transmitter and receiver to be disabled at the end of the current frame.

### 59.2.3.4 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. The transmitter and receiver bit clocks are not affected by Debug mode.

## 59.3 External signals

| Name | Function | I/O |
|---|---|---|
| SAI_TX_BCLK | Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated. | I/O |
| SAI_TX_SYNC | Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated | I/O |

*Table continues on the next page...*

| Name | Function | I/O |
|---|---|---|
| | synchronously by the bit clock when internally generated. | |
| SAI_TX_DATA[1:0] | Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word. | O |
| SAI_RX_BCLK | Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated. | I/O |
| SAI_RX_SYNC | Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated. | I/O |
| SAI_RX_DATA[1:0] | Receive Data. The receive data is sampled synchronously by the bit clock. | I |

## 59.4 Memory map and register definition

A read or write access to an address from offset 0x100 and above will result in a bus error.

### 59.4.1 I2S register descriptions

#### 59.4.1.1 I2S Memory map

I2S0 base address: 4003_D000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VERID) | 32 | RO | 0300_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0005_0302h |
| 8h | SAI Transmit Control Register (TCSR) | 32 | RW | 0000_0000h |
| Ch | SAI Transmit Configuration 1 Register (TCR1) | 32 | RW | 0000_0000h |
| 10h | SAI Transmit Configuration 2 Register (TCR2) | 32 | RW | 0000_0000h |
| 14h | SAI Transmit Configuration 3 Register (TCR3) | 32 | RW | 0000_0000h |
| 18h | SAI Transmit Configuration 4 Register (TCR4) | 32 | RW | 0000_0000h |
| 1Ch | SAI Transmit Configuration 5 Register (TCR5) | 32 | RW | 0000_0000h |
| 20h - 24h | SAI Transmit Data Register (TDR0 - TDR1) | 32 | WORZ | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 40h - 44h | SAI Transmit FIFO Register (TFR0 - TFR1) | 32 | RO | 0000_0000h |
| 60h | SAI Transmit Mask Register (TMR) | 32 | RW | 0000_0000h |
| 88h | SAI Receive Control Register (RCSR) | 32 | RW | 0000_0000h |
| 8Ch | SAI Receive Configuration 1 Register (RCR1) | 32 | RW | 0000_0000h |
| 90h | SAI Receive Configuration 2 Register (RCR2) | 32 | RW | 0000_0000h |
| 94h | SAI Receive Configuration 3 Register (RCR3) | 32 | RW | 0000_0000h |
| 98h | SAI Receive Configuration 4 Register (RCR4) | 32 | RW | 0000_0000h |
| 9Ch | SAI Receive Configuration 5 Register (RCR5) | 32 | RW | 0000_0000h |
| A0h - A4h | SAI Receive Data Register (RDR0 - RDR1) | 32 | RO | 0000_0000h |
| C0h - C4h | SAI Receive FIFO Register (RFR0 - RFR1) | 32 | RO | 0000_0000h |
| E0h | SAI Receive Mask Register (RMR) | 32 | RW | 0000_0000h |

## 59.4.1.2 Version ID Register (VERID)

### 59.4.1.2.1 Offset

| Register | Offset |
|---|---|
| VERID | 0h |

### 59.4.1.2.2 Function

.

### 59.4.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FEATURE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 59.4.1.2.4 Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>This read only field returns the major version number for the specification. |
| 23-16<br>MINOR | Minor Version Number<br>This read only field returns the minor version number for the specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>This read only field returns the feature set number.<br>0000000000000000b - Standard feature set. |

### 59.4.1.3 Parameter Register (PARAM)

#### 59.4.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 59.4.1.3.2 Function

.

#### 59.4.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FRAME | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FIFO | | | | 0 | | | | DATALINE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### 59.4.1.3.4 Fields

| Field | Function |
|---|---|
| 31-20 | Reserved |

*Table continues on the next page...*

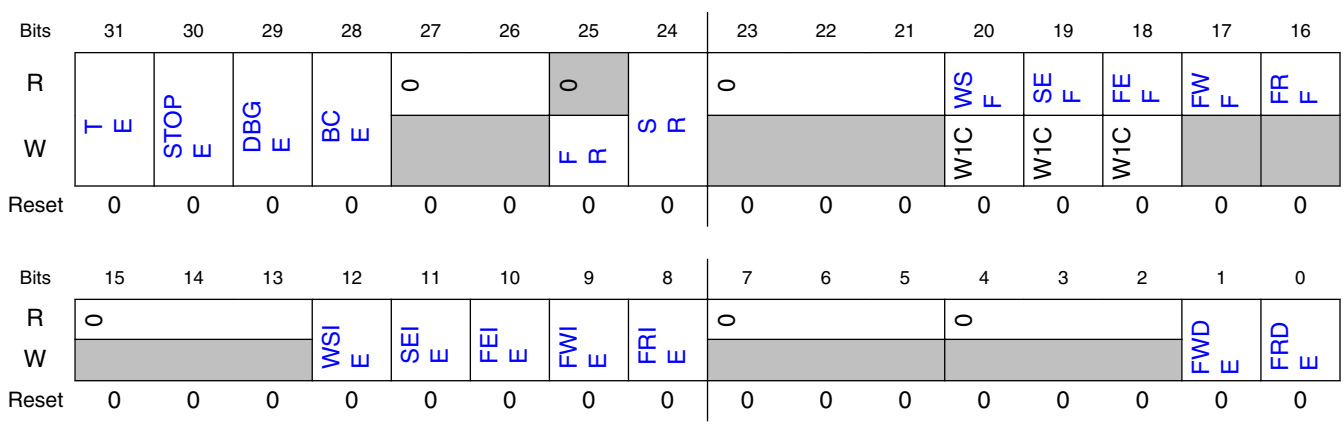
| Field | Function |
|---|---|
| — | |
| 19-16<br>FRAME | Frame Size<br>The maximum number of slots per frame is 2^FRAME. |
| 15-12<br>— | Reserved |
| 11-8<br>FIFO | FIFO Size<br>The number of words in each FIFO is 2^FIFO. |
| 7-4<br>— | Reserved |
| 3-0<br>DATALINE | Number of Datalines<br>The number of datalines implemented. |

## 59.4.1.4 SAI Transmit Control Register (TCSR)

### 59.4.1.4.1 Offset

| Register | Offset |
|---|---|
| TCSR | 8h |

### 59.4.1.4.2 Diagram




### 59.4.1.4.3 Fields

| Field | Function |
|---|---|
| 31 | Transmitter Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TE | Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame.<br>0b - Transmitter is disabled.<br>1b - Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame. |
| 30<br>STOPE | Stop Enable<br>Configures transmitter operation in Stop mode. The Stop Enable field is ignored and the transmitter is disabled in all low-leakage stop modes.<br>0b - Transmitter disabled in Stop mode.<br>1b - Transmitter enabled in Stop mode. |
| 29<br>DBGE | Debug Enable<br>Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode.<br>0b - Transmitter is disabled in Debug mode, after completing the current frame.<br>1b - Transmitter is enabled in Debug mode. |
| 28<br>BCE | Bit Clock Enable<br>Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame.<br>0b - Transmit bit clock is disabled.<br>1b - Transmit bit clock is enabled. |
| 27-26<br>— | Reserved |
| 25<br>FR | FIFO Reset<br>Empties the FIFO, and sets the FIFO read and write pointers to the same value, which may or may not be zero. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set.<br>0b - No effect.<br>1b - FIFO reset. |
| 24<br>SR | Software Reset<br>When set, resets the internal transmitter logic including the FIFO read and write pointers. Software-visible registers are not affected, except for the status registers.<br>0b - No effect.<br>1b - Software reset. |
| 23-21<br>— | Reserved |
| 20<br>WSF | Word Start Flag<br>Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag.<br>0b - Start of word not detected.<br>1b - Start of word detected. |
| 19<br>SEF | Sync Error Flag<br>Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag.<br>0b - Sync error not detected.<br>1b - Frame sync error detected. |
| 18<br>FEF | FIFO Error Flag<br>Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag.<br>0b - Transmit underrun not detected. |

*Table continues on the next page...*

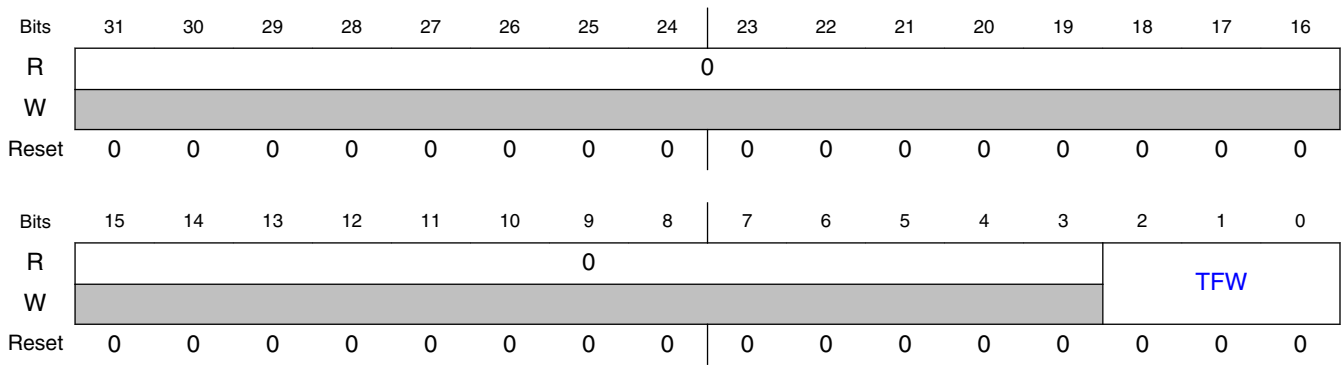
| Field | Function |
|---|---|
| | 1b - Transmit underrun detected. |
| 17<br>FWF | FIFO Warning Flag<br>Indicates that an enabled transmit FIFO is empty.<br>0b - No enabled transmit FIFO is empty.<br>1b - Enabled transmit FIFO is empty. |
| 16<br>FRF | FIFO Request Flag<br>Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark.<br>0b - Transmit FIFO watermark has not been reached.<br>1b - Transmit FIFO watermark has been reached. |
| 15-13<br>— | Reserved |
| 12<br>WSIE | Word Start Interrupt Enable<br>Enables/disables word start interrupts.<br>0b - Disables interrupt.<br>1b - Enables interrupt. |
| 11<br>SEIE | Sync Error Interrupt Enable<br>Enables/disables sync error interrupts.<br>0b - Disables interrupt.<br>1b - Enables interrupt. |
| 10<br>FEIE | FIFO Error Interrupt Enable<br>Enables/disables FIFO error interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br>Enables/disables FIFO warning interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |
| 8<br>FRIE | FIFO Request Interrupt Enable<br>Enables/disables FIFO request interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |
| 7-5<br>— | Reserved |
| 4-2<br>— | Reserved |
| 1<br>FWDE | FIFO Warning DMA Enable<br>Enables/disables DMA requests.<br>0b - Disables the DMA request.<br>1b - Enables the DMA request. |
| 0<br>FRDE | FIFO Request DMA Enable<br>Enables/disables DMA requests.<br>0b - Disables the DMA request.<br>1b - Enables the DMA request. |

## 59.4.1.5 SAI Transmit Configuration 1 Register (TCR1)

### 59.4.1.5.1 Offset

| Register | Offset |
|---|---|
| TCR1 | Ch |

### 59.4.1.5.2 Diagram




### 59.4.1.5.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>TFW | Transmit FIFO Watermark<br>Configures the watermark level for all enabled transmit channels. |

## 59.4.1.6 SAI Transmit Configuration 2 Register (TCR2)

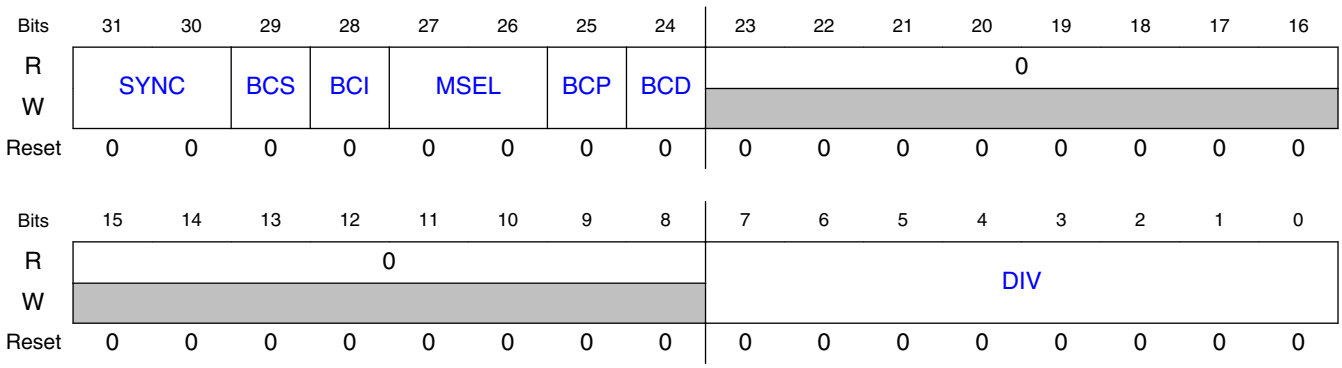
### 59.4.1.6.1 Offset

| Register | Offset |
|---|---|
| TCR2 | 10h |

### 59.4.1.6.2 Function

This register must not be altered when TCSR[TE] is set.

### 59.4.1.6.3 Diagram




### 59.4.1.6.4 Fields

| Field | Function |
|---|---|
| 31-30<br>SYNC | Synchronous Mode<br>Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver or other SAI peripheral must be configured for asynchronous operation.<br>00b - Asynchronous mode.<br>01b - Synchronous with receiver.<br>10b - Synchronous with another SAI transmitter.<br>11b - Synchronous with another SAI receiver. |
| 29<br>BCS | Bit Clock Swap<br>This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC).<br>When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC).<br>This field has no effect when synchronous to another SAI peripheral.<br>0b - Use the normal bit clock source.<br>1b - Swap the bit clock source. |
| 28<br>BCI | Bit Clock Input<br>When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. |

*Table continues on the next page...*

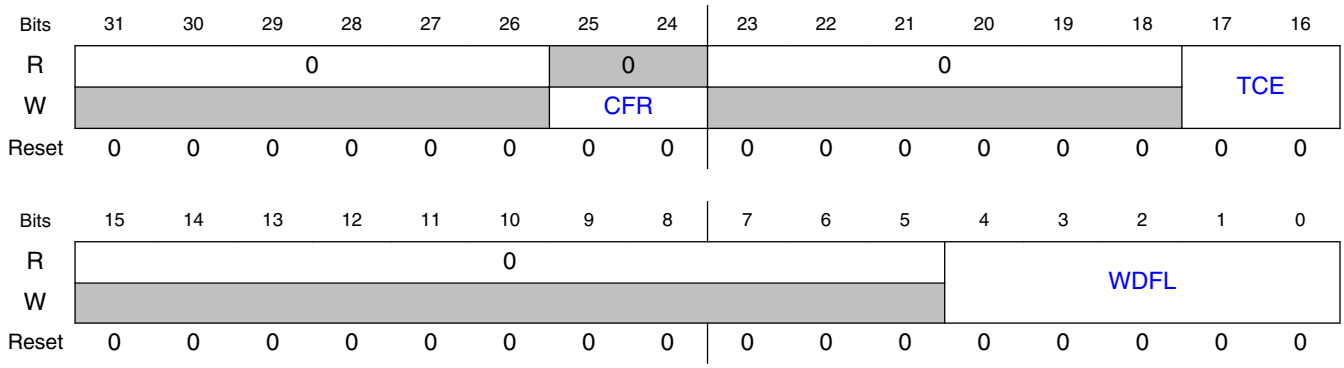
| Field | Function |
|---|---|
| | The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral .<br>0b - No effect.<br>1b - Internal logic is clocked as if bit clock was externally generated. |
| 27-26<br>MSEL | MCLK Select<br>Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock.<br>**NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option.<br>00b - Bus Clock selected.<br>01b - Master Clock (MCLK) 1 option selected.<br>10b - Master Clock (MCLK) 2 option selected.<br>11b - Master Clock (MCLK) 3 option selected. |
| 25<br>BCP | Bit Clock Polarity<br>Configures the polarity of the bit clock.<br>0b - Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge.<br>1b - Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge. |
| 24<br>BCD | Bit Clock Direction<br>Configures the direction of the bit clock.<br>0b - Bit clock is generated externally in Slave mode.<br>1b - Bit clock is generated internally in Master mode. |
| 23-8<br>— | Reserved |
| 7-0<br>DIV | Bit Clock Divide<br>Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2. |

## 59.4.1.7 SAI Transmit Configuration 3 Register (TCR3)

### 59.4.1.7.1 Offset

| Register | Offset |
|---|---|
| TCR3 | 14h |

## 59.4.1.7.2 Diagram




## 59.4.1.7.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>CFR | Channel FIFO Reset<br>Resets the FIFO pointers for a specific channel. Reading this field will always return zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set.<br>The width of CFR field = the number of transmit channels (call it N). For example, if CFR is 2 bits wide, then bit position 24 refers to transmit channel 1 FIFO pointer and bit position 25 refers to transmit channel 2 FIFO pointer. Setting bit 24 resets transmit channel 1 FIFO pointer, and setting bit 25 enables transmit channel 2 FIFO pointer. Setting bit N will reset transmit channel N FIFO pointer.<br>0b - No effect.<br>1b - Transmit data channel N FIFO is reset. |
| 23-18<br>— | Reserved |
| 17-16<br>TCE | Transmit Channel Enable<br>Enables the corresponding data channel for transmit operation. Changing TCE field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for transmit operation.<br>The width of TCE field = the number of transmit channels (call it N). For example, if TCE field is 2 bits wide, then bit position 16 refers to transmit channel 1 and bit position 17 refers to transmit channel 2. Setting bit 16 enables transmit channel 1, and setting bit 17 enables transmit channel 2. Setting bit N will enable transmit channel N.<br>0b - Transmit data channel N is disabled.<br>1b - Transmit data channel N is enabled. |
| 15-5<br>— | Reserved |
| 4-0<br>WDFL | Word Flag Configuration<br>Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set. |

## 59.4.1.8 SAI Transmit Configuration 4 Register (TCR4)

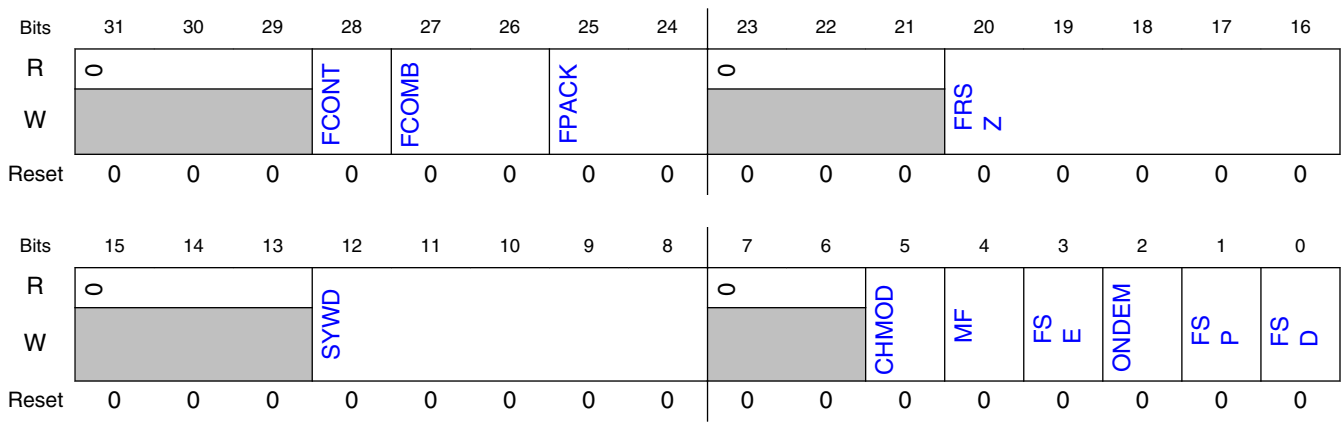
### 59.4.1.8.1 Offset

| Register | Offset |
|---|---|
| TCR4 | 18h |

### 59.4.1.8.2 Function

This register must not be altered when TCSR[TE] is set.

### 59.4.1.8.3 Diagram




### 59.4.1.8.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>FCONT | FIFO Continue on Error<br>Configures when the SAI will continue transmitting after a FIFO error has been detected.<br>0b - On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared.<br>1b - On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared. |
| 27-26<br>FCOMB | FIFO Combine Mode |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | When FIFO combine mode is enabled for FIFO writes, software writing to any FIFO data register will alternate the write among the enabled data channel FIFOs. For example, if two data channels are enabled then the first write will be performed to the first enabled data channel FIFO and the second write will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO writes will reset the pointer back to the first enabled data channel.<br>When FIFO combine mode is enabled for FIFO reads from the transmit shift registers, the transmit data channel output will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked word will be transmitted from the first enabled data channel FIFO and the second unmasked word will be transmitted from the second enabled data channel FIFO. Since the first word of the frame is always transmitted from the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels.<br>00b - FIFO combine mode disabled.<br>01b - FIFO combine mode enabled on FIFO reads (from transmit shift registers).<br>10b - FIFO combine mode enabled on FIFO writes (by software).<br>11b - FIFO combine mode enabled on FIFO reads (from transmit shift registers) and writes (by software). |
| 25-24<br>FPACK | FIFO Packing Mode<br>Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are loaded from the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO write pointer will only increment when the full 32-bit FIFO word has been written by software.<br>00b - FIFO packing is disabled<br>01b - Reserved<br>10b - 8-bit FIFO packing is enabled<br>11b - 16-bit FIFO packing is enabled |
| 23-21<br>— | Reserved |
| 20-16<br>FRSZ | Frame size<br>Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words. |
| 15-13<br>— | Reserved |
| 12-8<br>SYWD | Sync Width<br>Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame. |
| 7-6<br>— | Reserved |
| 5<br>CHMOD | Channel Mode<br>Configures if transmit data pins are configured for TDM mode or Output mode.<br>0b - TDM mode, transmit data pins are tri-stated when slots are masked or channels are disabled.<br>1b - Output mode, transmit data pins are never tri-stated and will output zero when slots are masked or channels are disabled. |
| 4<br>MF | MSB First<br>Configures whether the LSB or the MSB is transmitted first.<br>0b - LSB is transmitted first. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - MSB is transmitted first. |
| 3<br>FSE | Frame Sync Early<br>0b - Frame sync asserts with the first bit of the frame.<br>1b - Frame sync asserts one bit before the first bit of the frame. |
| 2<br>ONDEM | On Demand Mode<br>When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear.<br>0b - Internal frame sync is generated continuously.<br>1b - Internal frame sync is generated when the FIFO warning flag is clear. |
| 1<br>FSP | Frame Sync Polarity<br>Configures the polarity of the frame sync.<br>0b - Frame sync is active high.<br>1b - Frame sync is active low. |
| 0<br>FSD | Frame Sync Direction<br>Configures the direction of the frame sync.<br>0b - Frame sync is generated externally in Slave mode.<br>1b - Frame sync is generated internally in Master mode. |

## 59.4.1.9 SAI Transmit Configuration 5 Register (TCR5)

### 59.4.1.9.1 Offset

| Register | Offset |
|---|---|
| TCR5 | 1Ch |

### 59.4.1.9.2 Function

This register must not be altered when TCSR[TE] is set.

### 59.4.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WNW | | | | | 0 | | | W0W | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FBT | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.9.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-24<br>WNW | Word N Width<br>Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported. |
| 23-21<br>— | Reserved |
| 20-16<br>W0W | Word 0 Width<br>Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame. |
| 15-13<br>— | Reserved |
| 12-8<br>FBT | First Bit Shifted<br>Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First. |
| 7-0<br>— | Reserved |

## 59.4.1.10 SAI Transmit Data Register (TDR0 - TDR1)

### 59.4.1.10.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| TDRa | 20h + (a × 4h) |

#### 59.4.1.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | TDR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | TDR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 59.4.1.10.3 Fields

| Field | Function |
|---|---|
| 31-0<br>TDR | Transmit Data Register<br>Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored. |

### 59.4.1.11 SAI Transmit FIFO Register (TFR0 - TFR1)

#### 59.4.1.11.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| TFRa | 40h + (a × 4h) |

#### 59.4.1.11.2 Function

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

### 59.4.1.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WCP | 0 | | | | | | | | | | | WFP | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | RFP | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.11.4 Fields

| Field | Function |
|---|---|
| 31<br>WCP | Write Channel Pointer<br>When FIFO Combine mode is enabled for writes, indicates that this data channel is the next FIFO to be written.<br>0b - No effect.<br>1b - FIFO combine is enabled for FIFO writes and this FIFO will be written on the next FIFO write. |
| 30-20<br>— | Reserved |
| 19-16<br>WFP | Write FIFO Pointer<br>FIFO write pointer for transmit data channel. |
| 15-4<br>— | Reserved |
| 3-0<br>RFP | Read FIFO Pointer<br>FIFO read pointer for transmit data channel. |

## 59.4.1.12 SAI Transmit Mask Register (TMR)

### 59.4.1.12.1 Offset

| Register | Offset |
|---|---|
| TMR | 60h |

### 59.4.1.12.2 Function

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

### 59.4.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TWM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TWM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.12.4 Fields

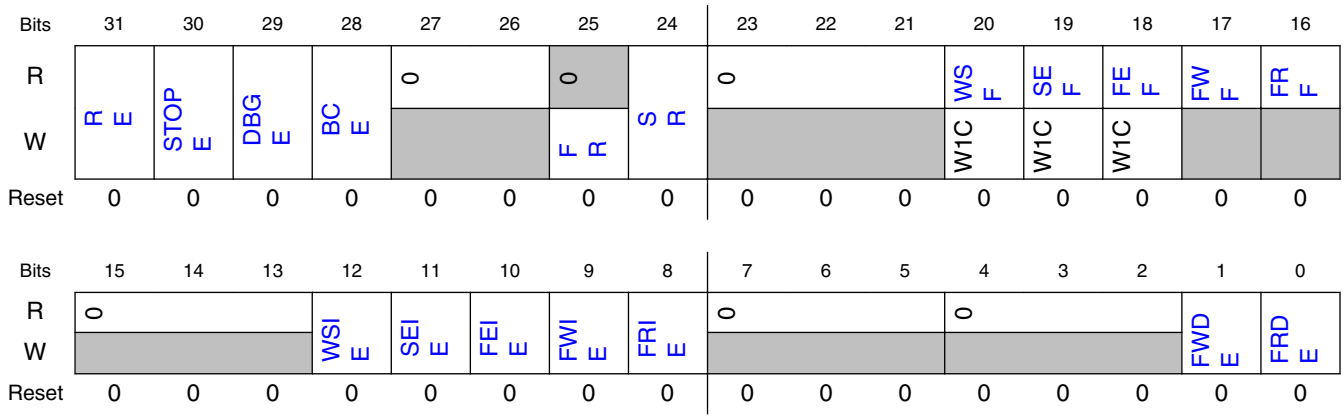
| Field | Function |
|---|---|
| 31-0<br>TWM | Transmit Word Mask<br>Configures whether the transmit word is masked (transmit data pins are tri-stated or drive zero and transmit data not read from FIFO) for the corresponding word in the frame.<br>00000000000000000000000000000000b - Word N is enabled.<br>00000000000000000000000000000001b - Word N is masked. The transmit data pins are tri-stated or drive zero when masked. |

## 59.4.1.13 SAI Receive Control Register (RCSR)

### 59.4.1.13.1 Offset

| Register | Offset |
|---|---|
| RCSR | 88h |

## 59.4.1.13.2 Diagram




## 59.4.1.13.3 Fields

| Field | Function |
|---|---|
| 31<br>RE | Receiver Enable<br>Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame.<br>0b - Receiver is disabled.<br>1b - Receiver is enabled, or receiver has been disabled and has not yet reached end of frame. |
| 30<br>STOPE | Stop Enable<br>Configures receiver operation in Stop mode. The Stop Enable field is ignored and the receiver is disabled in all low-leakage stop modes.<br>0b - Receiver disabled in Stop mode.<br>1b - Receiver enabled in Stop mode. |
| 29<br>DBGE | Debug Enable<br>Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode.<br>0b - Receiver is disabled in Debug mode, after completing the current frame.<br>1b - Receiver is enabled in Debug mode. |
| 28<br>BCE | Bit Clock Enable<br>Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame.<br>0b - Receive bit clock is disabled.<br>1b - Receive bit clock is enabled. |
| 27-26<br>— | Reserved |
| 25<br>FR | FIFO Reset<br>Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set.<br>0b - No effect.<br>1b - FIFO reset. |
| 24 | Software Reset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SR | Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers.<br>0b - No effect.<br>1b - Software reset. |
| 23-21<br>— | Reserved |
| 20<br>WSF | Word Start Flag<br>Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag.<br>0b - Start of word not detected.<br>1b - Start of word detected. |
| 19<br>SEF | Sync Error Flag<br>Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag.<br>0b - Sync error not detected.<br>1b - Frame sync error detected. |
| 18<br>FEF | FIFO Error Flag<br>Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag.<br>0b - Receive overflow not detected.<br>1b - Receive overflow detected. |
| 17<br>FWF | FIFO Warning Flag<br>Indicates that an enabled receive FIFO is full.<br>0b - No enabled receive FIFO is full.<br>1b - Enabled receive FIFO is full. |
| 16<br>FRF | FIFO Request Flag<br>Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark.<br>0b - Receive FIFO watermark not reached.<br>1b - Receive FIFO watermark has been reached. |
| 15-13<br>— | Reserved |
| 12<br>WSIE | Word Start Interrupt Enable<br>Enables/disables word start interrupts.<br>0b - Disables interrupt.<br>1b - Enables interrupt. |
| 11<br>SEIE | Sync Error Interrupt Enable<br>Enables/disables sync error interrupts.<br>0b - Disables interrupt.<br>1b - Enables interrupt. |
| 10<br>FEIE | FIFO Error Interrupt Enable<br>Enables/disables FIFO error interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br>Enables/disables FIFO warning interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |

*Table continues on the next page...*

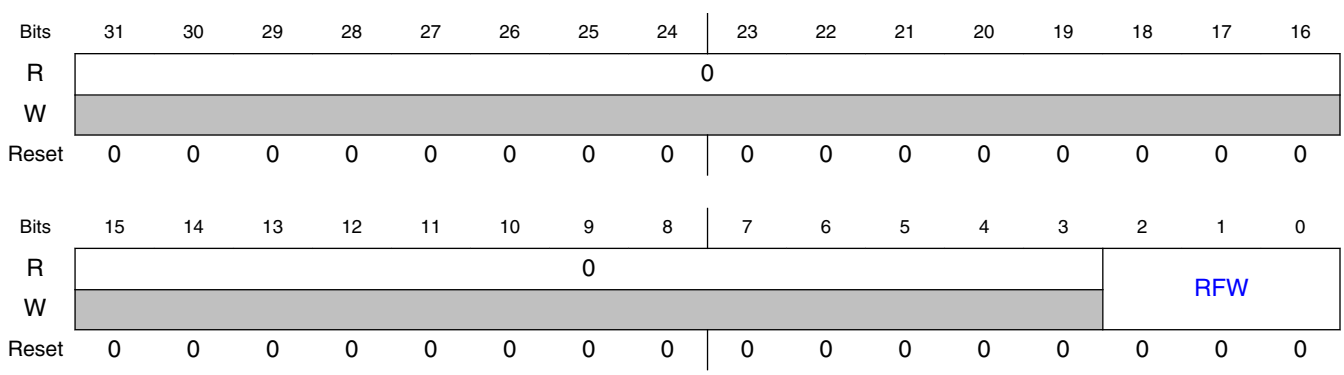
| Field | Function |
|---|---|
| 8<br>FRIE | FIFO Request Interrupt Enable<br>Enables/disables FIFO request interrupts.<br>0b - Disables the interrupt.<br>1b - Enables the interrupt. |
| 7-5<br>— | Reserved |
| 4-2<br>— | Reserved |
| 1<br>FWDE | FIFO Warning DMA Enable<br>Enables/disables DMA requests.<br>0b - Disables the DMA request.<br>1b - Enables the DMA request. |
| 0<br>FRDE | FIFO Request DMA Enable<br>Enables/disables DMA requests.<br>0b - Disables the DMA request.<br>1b - Enables the DMA request. |

## 59.4.1.14 SAI Receive Configuration 1 Register (RCR1)

### 59.4.1.14.1 Offset

| Register | Offset |
|---|---|
| RCR1 | 8Ch |

### 59.4.1.14.2 Diagram



### 59.4.1.14.3 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>RFW | Receive FIFO Watermark<br>Configures the watermark level for all enabled receiver channels. |

## 59.4.1.15 SAI Receive Configuration 2 Register (RCR2)

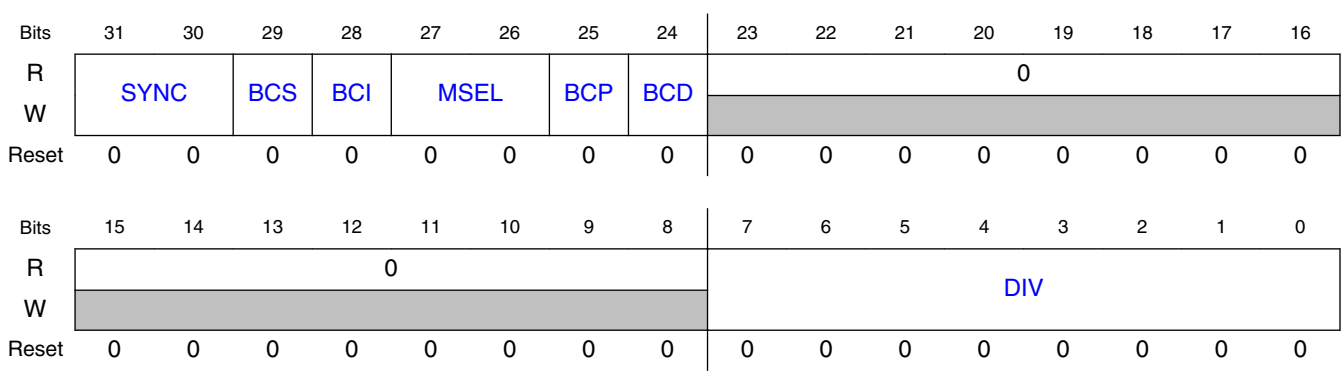
### 59.4.1.15.1 Offset

| Register | Offset |
|---|---|
| RCR2 | 90h |

### 59.4.1.15.2 Function

This register must not be altered when RCSR[RE] is set.

### 59.4.1.15.3 Diagram




### 59.4.1.15.4 Fields

| Field | Function |
|---|---|
| 31-30<br>SYNC | Synchronous Mode<br>Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter or other SAI peripheral must be configured for asynchronous operation. |

*Table continues on the next page...*

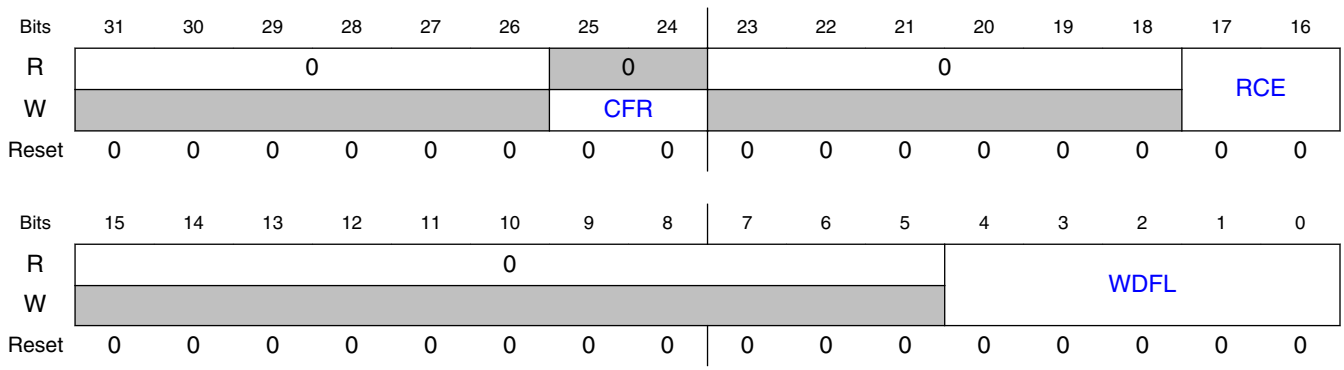
| Field | Function |
|---|---|
| | 00b - Asynchronous mode.<br>01b - Synchronous with transmitter.<br>10b - Synchronous with another SAI receiver.<br>11b - Synchronous with another SAI transmitter. |
| 29<br>BCS | Bit Clock Swap<br>This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC).<br>When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC).<br>This field has no effect when synchronous to another SAI peripheral.<br>0b - Use the normal bit clock source.<br>1b - Swap the bit clock source. |
| 28<br>BCI | Bit Clock Input<br>When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time.<br>The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral .<br>0b - No effect.<br>1b - Internal logic is clocked as if bit clock was externally generated. |
| 27-26<br>MSEL | MCLK Select<br>Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock.<br>**NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option.<br>00b - Bus Clock selected.<br>01b - Master Clock (MCLK) 1 option selected.<br>10b - Master Clock (MCLK) 2 option selected.<br>11b - Master Clock (MCLK) 3 option selected. |
| 25<br>BCP | Bit Clock Polarity<br>Configures the polarity of the bit clock.<br>0b - Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge.<br>1b - Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge. |
| 24<br>BCD | Bit Clock Direction<br>Configures the direction of the bit clock.<br>0b - Bit clock is generated externally in Slave mode.<br>1b - Bit clock is generated internally in Master mode. |
| 23-8<br>— | Reserved |
| 7-0<br>DIV | Bit Clock Divide<br>Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2. |

# 59.4.1.16 SAI Receive Configuration 3 Register (RCR3)

## 59.4.1.16.1 Offset

| Register | Offset |
|---|---|
| RCR3 | 94h |

## 59.4.1.16.2 Diagram




## 59.4.1.16.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>CFR | Channel FIFO Reset<br>Resets the FIFO pointers for a specific channel. Reading this field will always return zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set.<br>The width of CFR field = the number of receive channels (call it N). For example, if CFR is 2 bits wide, then bit position 24 refers to receive channel 1 FIFO pointer and bit position 25 refers to receive channel 2 FIFO pointer. Setting bit 24 resets receive channel 1 FIFO pointer, and setting bit 25 enables receive channel 2 FIFO pointer. Setting bit N will reset receive channel N FIFO pointer.<br>0b - No effect.<br>1b - Receive data channel N FIFO is reset. |
| 23-18<br>— | Reserved |
| 17-16<br>RCE | Receive Channel Enable<br>Enables the corresponding data channel for receive operation. Changing this field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for receive operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The width of RCE field = the number of receive channels (call it N). For example, if RCE field is 2 bits wide, then bit position 16 refers to receive channel 1 and bit position 17 refers to receive channel 2. Setting bit 16 enables receive channel 1, and setting bit 17 enables receive channel 2. Setting bit N will enable receive channel N.<br>0b - Receive data channel N is disabled.<br>1b - Receive data channel N is enabled. |
| 15-5<br>— | Reserved |
| 4-0<br>WDFL | Word Flag Configuration<br>Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set. |

## 59.4.1.17 SAI Receive Configuration 4 Register (RCR4)

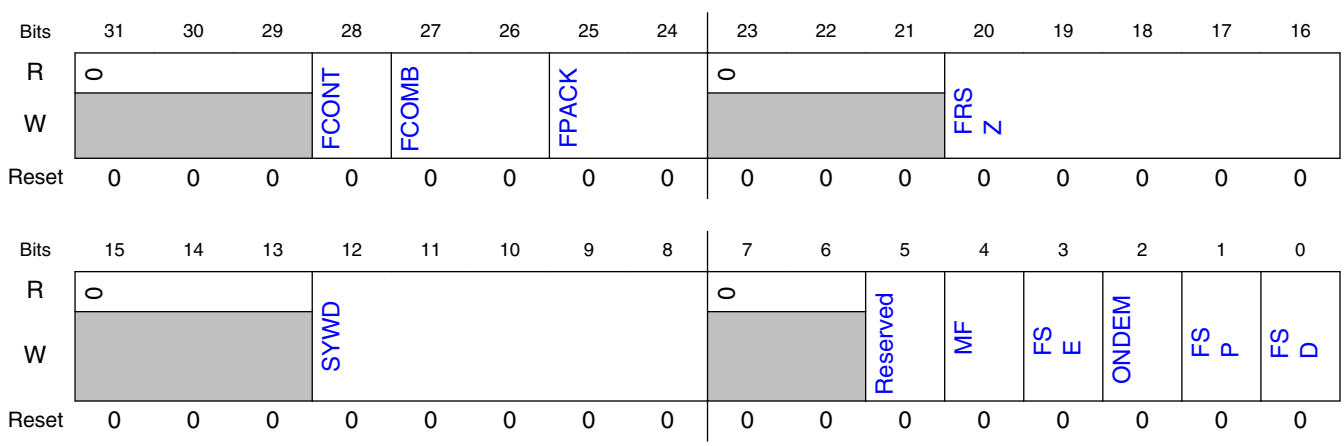
### 59.4.1.17.1 Offset

| Register | Offset |
|---|---|
| RCR4 | 98h |

### 59.4.1.17.2 Function

This register must not be altered when RCSR[RE] is set.

### 59.4.1.17.3 Diagram



## 59.4.1.17.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>FCONT | FIFO Continue on Error<br>Configures when the SAI will continue receiving after a FIFO error has been detected.<br>0b - On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared.<br>1b - On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared. |
| 27-26<br>FCOMB | FIFO Combine Mode<br>When FIFO combine mode is enabled for FIFO reads, software reading any FIFO data register will alternate the read among the enabled data channel FIFOs. For example, if two data channels are enabled then the first read will be performed to the first enabled data channel FIFO and the second read will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO reads will reset the pointer back to the first enabled data channel.<br>When FIFO combine mode is enabled for FIFO writes from the receive shift registers, the first enabled data channel input will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked received word will be stored in the first enabled data channel FIFO and the second unmasked received word will be stored in the second enabled data channel FIFO. Since the first word of the frame is always stored in the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels.<br>00b - FIFO combine mode disabled.<br>01b - FIFO combine mode enabled on FIFO writes (from receive shift registers).<br>10b - FIFO combine mode enabled on FIFO reads (by software).<br>11b - FIFO combine mode enabled on FIFO writes (from receive shift registers) and reads (by software). |
| 25-24<br>FPACK | FIFO Packing Mode<br>Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are stored to the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO read pointer will only increment when the full 32-bit FIFO word has been read by software.<br>00b - FIFO packing is disabled<br>01b - Reserved.<br>10b - 8-bit FIFO packing is enabled<br>11b - 16-bit FIFO packing is enabled |
| 23-21<br>— | Reserved |
| 20-16<br>FRSZ | Frame Size<br>Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words. |
| 15-13<br>— | Reserved |
| 12-8<br>SYWD | Sync Width |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame. |
| 7-6<br>— | Reserved |
| 5<br>— | Reserved. Software should only write zero to this bit. |
| 4<br>MF | MSB First<br>Configures whether the LSB or the MSB is received first.<br>0b - LSB is received first.<br>1b - MSB is received first. |
| 3<br>FSE | Frame Sync Early<br>0b - Frame sync asserts with the first bit of the frame.<br>1b - Frame sync asserts one bit before the first bit of the frame. |
| 2<br>ONDEM | On Demand Mode<br>When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear.<br>0b - Internal frame sync is generated continuously.<br>1b - Internal frame sync is generated when the FIFO warning flag is clear. |
| 1<br>FSP | Frame Sync Polarity<br>Configures the polarity of the frame sync.<br>0b - Frame sync is active high.<br>1b - Frame sync is active low. |
| 0<br>FSD | Frame Sync Direction<br>Configures the direction of the frame sync.<br>0b - Frame Sync is generated externally in Slave mode.<br>1b - Frame Sync is generated internally in Master mode. |

## 59.4.1.18 SAI Receive Configuration 5 Register (RCR5)

### 59.4.1.18.1 Offset

| Register | Offset |
|---|---|
| RCR5 | 9Ch |

### 59.4.1.18.2 Function

This register must not be altered when RCSR[RE] is set.

### 59.4.1.18.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WNW | | | | | 0 | | | W0W | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FBT | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.18.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-24<br>WNW | Word N Width<br>Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported. |
| 23-21<br>— | Reserved |
| 20-16<br>W0W | Word 0 Width<br>Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame. |
| 15-13<br>— | Reserved |
| 12-8<br>FBT | First Bit Shifted<br>Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First. |
| 7-0<br>— | Reserved |

## 59.4.1.19 SAI Receive Data Register (RDR0 - RDR1)

### 59.4.1.19.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| RDRa | A0h + (a × 4h) |

### 59.4.1.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.19.3 Fields

| Field | Function |
|---|---|
| 31-0<br>RDR | Receive Data Register<br>Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored. |

## 59.4.1.20 SAI Receive FIFO Register (RFR0 - RFR1)

### 59.4.1.20.1 Offset

For a = 0 to 1:

| Register | Offset |
|---|---|
| RFRa | C0h + (a × 4h) |

### 59.4.1.20.2 Function

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

### 59.4.1.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | WFP | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RCP | 0 | | | | | | | | | | | RFP | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 59.4.1.20.4 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>WFP | Write FIFO Pointer<br>FIFO write pointer for receive data channel. |
| 15<br>RCP | Receive Channel Pointer<br>When FIFO Combine mode is enabled for reads, indicates that this data channel is the next FIFO to be read.<br>0b - No effect.<br>1b - FIFO combine is enabled for FIFO reads and this FIFO will be read on the next FIFO read. |
| 14-4<br>— | Reserved |
| 3-0<br>RFP | Read FIFO Pointer<br>FIFO read pointer for receive data channel. |

## 59.4.1.21 SAI Receive Mask Register (RMR)

#### 59.4.1.21.1 Offset

| Register | Offset |
|---|---|
| RMR | E0h |

#### 59.4.1.21.2 Function

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

#### 59.4.1.21.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RWM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RWM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 59.4.1.21.4 Fields

| Field | Function |
|---|---|
| 31-0<br>RWM | Receive Word Mask<br>Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame.<br>00000000000000000000000000000000b - Word N is enabled.<br>00000000000000000000000000000001b - Word N is masked. |

## 59.5 Functional description

This section provides a complete functional description of the block.

### 59.5.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

#### 59.5.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated.

#### 59.5.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

- If both transmitter and receiver are configured for asynchronous operation, then the transmitter and receiver will each use *their own* bit clock and frame sync.
- If the *transmitter* is configured for asynchronous mode and the receiver is configured for synchronous mode, then both transmitter and receiver will use the *transmitter* bit clock and frame sync.
- If the *receiver* is configured for asynchronous mode and the transmitter is configured for synchronous mode, then both transmitter and receiver will use the *receiver* bit clock and frame sync.

Note that the software configures synchronous or asynchronous mode, and that choice selects the bit clock/frame sync used.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver is using an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

#### 59.5.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

**NOTE**

Although there is no specific minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

### 59.5.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

#### 59.5.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

#### 59.5.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI transmitter can also reset the FIFO of individual data channels by setting the appropriate TCR3[CFR] bit. This should only be done when the corresponding TCR3[TCE] bit is clear.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver can also reset the FIFO of individual data channels by setting the appropriate RCR3[CFR] bit. This should only be done when the corresponding RCR3[RCE] bit is clear.

### 59.5.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other or synchronously to other SAI peripherals.

#### 59.5.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

#### 59.5.3.2 Multiple SAI Synchronous mode

Synchronous operation between multiple SAI peripherals is not supported on all devices. This mode requires the source of the bit clock and frame sync to be configured for asynchronous operation and the remaining users of the bit clock and frame sync to be configured for synchronous operation.

Synchronous operation between multiple SAI transmitters or receivers also requires the source of the bit clock and frame sync to be enabled for any of the synchronous transmitters or receivers to also be enabled. It is recommended that the source of the bit clock and frame sync is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The separate SAI peripherals otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

### 59.5.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word
  - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

## 59.5.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 8 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

### 59.5.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 59-2 for LSB First configurations and Figure 59-3 for MSB First configurations.

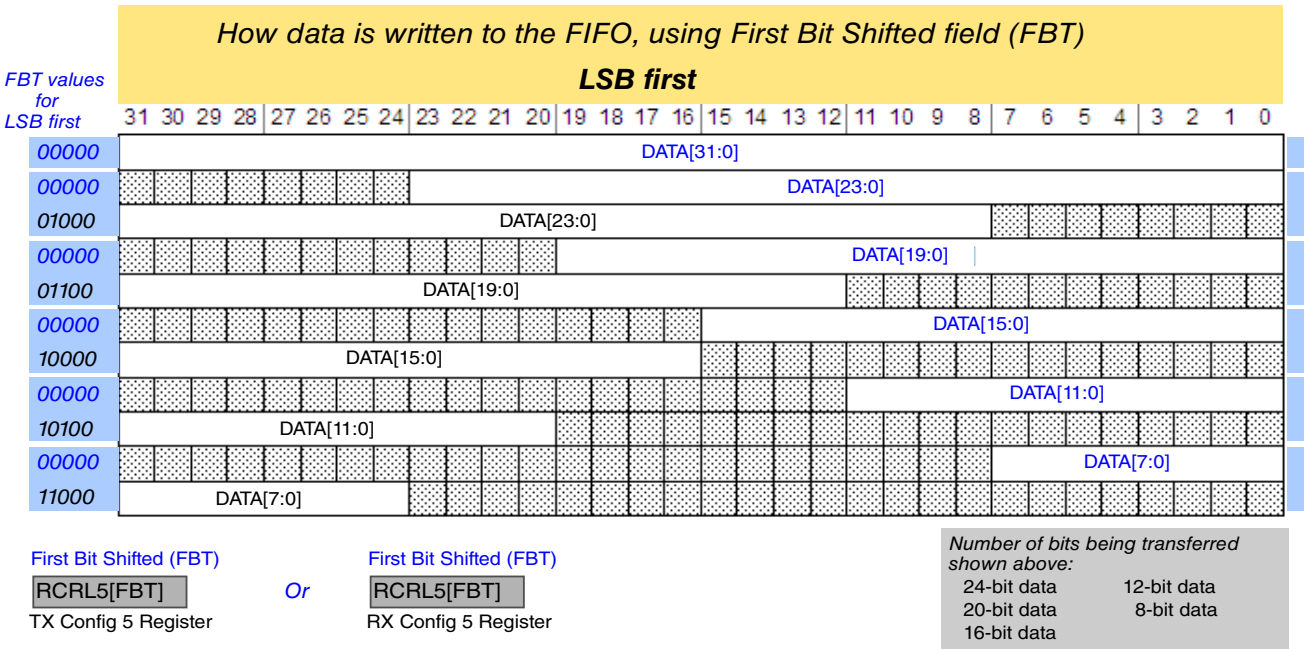



**Figure 59-2. SAI first bit shifted, LSB first**

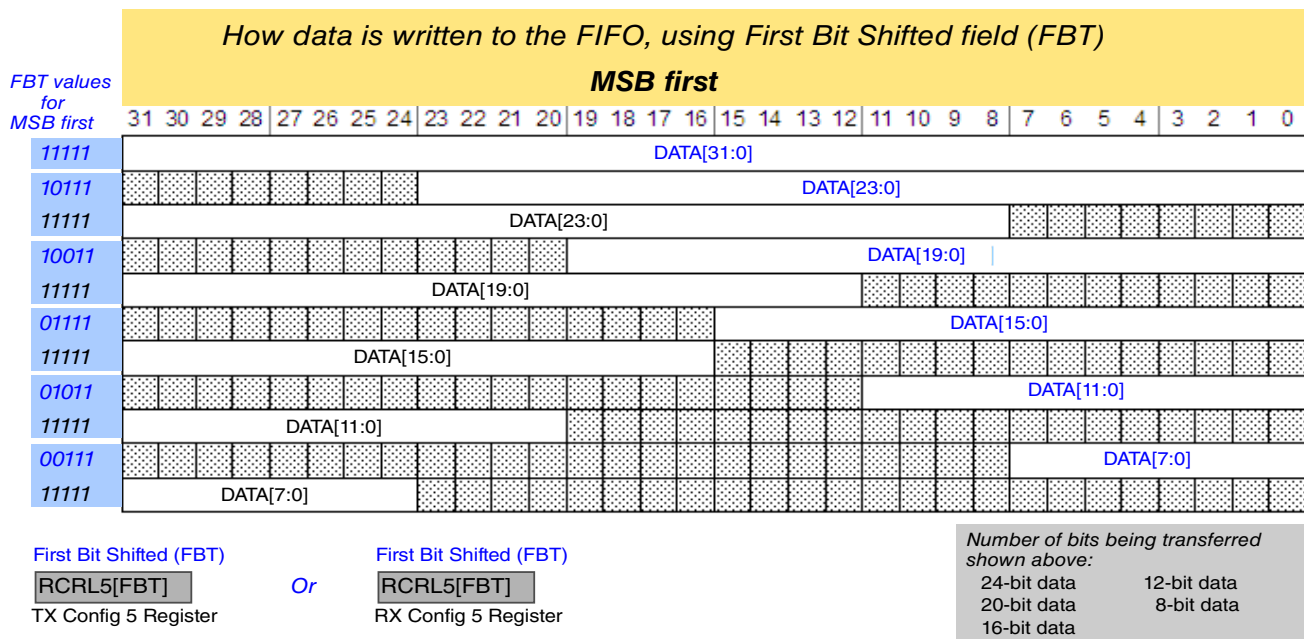



**Figure 59-3. SAI first bit shifted, MSB first**

### 59.5.5.2 FIFO pointers

When writing to a Transmit Data Register (TDR*n*), the Write FIFO Pointer (WFP) of the corresponding Transmit FIFO Register (TFR*n*) increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the Transmit Data Register and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data; 16-bit writes should only be used when transmitting up to 16-bit data.

- If the Transmit FIFO is full, then writes to a Transmit Data Register are ignored.
- If the Transmit FIFO is empty, then to avoid a FIFO underrun, the Transmit Data Register must be written at least 3 bit clocks before the start of the next unmasked word. Before enabling the transmitter, the Transmit FIFO should be initialized with data (since after the transmitter is enabled, the transmitter will start a new frame, and if no data is in the FIFO, then the transmitter will immediately give an error).

When reading a Receive Data Register (RDR*n*), the Read FIFO Pointer (RFP) of the corresponding Receive FIFO Register (RFR*n*) increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data; 16-bit reads should only be used when receiving up to 16-bit data.

- If the Receive FIFO is empty, then reads from a Receive Data Register are ignored.
- If the Receive FIFO is full, then to avoid a FIFO overrun, the Receive Data Register must be read at least 3 bit clocks before the end of an unmasked word.

### 59.5.5.3 FIFO packing

FIFO packing supports storing multiple 8-bit or 16-bit data words in one 32-bit FIFO word for the transmitter and/or receiver. While this can be emulated by adjusting the number of bits per word and number of words per frame (for example, one 32-bit word per frame versus two 16-bit words per frame), FIFO packing does not require even multiples of words per frame and fully supports word masking. When FIFO packing is enabled, the FIFO pointers only increment when the full 32-bit FIFO word has been written (transmit) or read (receive) by software, supporting scenarios where different words within each frame are loaded/stored in different areas of memory.

When 16-bit FIFO packing is enabled for transmit, the transmit shift register is loaded at the start of each frame and after every second unmasked transmit word. The first word transmitted is taken from 16-bit word at byte offset $0 (first bit is selected by TCFG5[FBT] must be configured within this 16-bit word) and the second word transmitted is taken from the 16-bit word at byte offset $2 (first bit is selected by TCSR5[FBT][3:0]). The transmitter will transmit logic zero until the start of the next word once the 16-bit word has been transmitted.

When 16-bit FIFO packing is enabled for receive, the receive shift register is stored after every second unmasked received word, and at the end of each frame if there is an odd number of unmasked received words in each frame. The first word received is stored in the 16-bit word at byte offset $0 (first bit is selected by RCFG5[FBT] and must be configured within this 16-bit word) and the second word received is stored in the 16-bit word at byte offset $2 (first bit is selected by RCSR5[FBT][3:0]). The receiver will ignore received data until the start of the next word once the 16-bit word has been received.

The 8-bit FIFO packing is similar to 16-bit packing except four words are loaded or stored into each 32-bit FIFO word. The first word is loaded/stored in byte offset $0, second word in byte offset $1, third word in byte offset $2 and fourth word in byte offset $3. The TCFG5[FBT] and/or RCFG5[FBT] must be configured within byte offset $0.

### 59.5.5.4 FIFO Combine

FIFO combining mode allows the separate FIFOs for multiple data channels to be used as a single FIFO for either software accesses or a single data channel or both. Note that the enabled data channels must be contiguous and data channel 0 must be enabled when FIFO Combine mode is enabled.

Combining FIFOs for software access (writing transmit FIFO registers, reading receive FIFO registers) allows a DMA controller or software to read or write multiple FIFOs without incrementing the address that is accessed. Once enabled, the first software access to a FIFO register will access the first enabled channel FIFO, while the second access to a FIFO register will access the second enabled channel FIFO. This continues until software accesses the last enabled channel FIFO and the pointer resets back to the first enabled channel FIFO. To reset the pointer manually, software can reset the FIFOs or disable the FIFO combining on software accesses.

Combining FIFOs for transmit data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with identical data output on each enabled data channel. The transmit shift registers for all enabled data channels are loaded at the start of each frame and every N unmasked words (where N is the number of enabled data channels). The first word transmitted is loaded from the first enabled channel FIFO, while the second word transmitted is loaded from the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always loaded from the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Combining FIFOs for receive data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with received data from channel 0 stored into each enabled data channel. The receive shift register for all enabled data channels are stored after every N unmasked words (where N is the number of enabled data channels). The first word received is stored to the first enabled channel FIFO, while the second word received is stored to the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always stored the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Note that combining FIFOs for data channels will load or store each channel FIFO at the same time. This means that FIFO error conditions are only checked every N words (where N is the number of enabled data channels) and that the FIFO warning and request flags will assert if any of the enabled data channel meets the warning flag or request flag conditions.

## 59.5.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

## 59.5.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags. Asynchronous versions of the transmitter and receiver interrupts are generated to wake up the CPU from stop mode.

### 59.5.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

### 59.5.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

### 59.5.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels will transmit zero data before TCSR[FEF] is cleared.

When TCR4[FCONT] is set, the FIFO will continue transmitting data following an underflow without software intervention. To ensure that data is transmitted in the correct order, the transmitter will continue from the same word number in the frame that caused the FIFO to underflow, but only after new data has been written to the transmit FIFO. Software should still clear the TCSR[FEF] flag, but without reinitializing the transmit FIFOs.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

When RCR4[FCONT] is set, the FIFO will continue receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver will continue from the same word number in the frame that caused the FIFO to overflow, but only after data has been read from the receive FIFO. Software should still clear the RCSR[FEF] flag, but without emptying the receive FIFOs.

The FIFO error flag can generate only an interrupt.

#### 59.5.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

#### 59.5.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

# Chapter 60
# Smart Card Interface Module (EMV SIM)

## 60.1 Chip-specific EMV SIM information

### 60.1.1 EMV_SIM overview

The Euro/Mastercard/Visa Serial Interface Module (EMV_SIM) is a module that supports ISO-7816 protocol. For more information, please see Introduction

### 60.1.2 EMV_SIM Instantiation

This device contains 1 EMV_SIM module. This section describes how each module is configured on this device.

## 60.2 Introduction

The Euro, MasterCard, Visa Subscriber Identification Module (EMV SIM) is designed to facilitate communication to Smart Cards compatible to the EMV ver4.3 standard (Book 1) and Smart Cards compatible with ISO/IEC 7816-3 Standard.

### 60.2.1 Features

The EMV SIM module supports the following features:

- Supports Smart Cards based on the EMV Standard v4.3 and ISO 7816-3 standard

- Independent clock for SIM logic (transmitter + receiver) and independent clock for register read-write interface
- 16 byte deep FIFO for transmitter and receiver
- Automatic NACK generation on parity error and receiver FIFO overflow error
- Support for both Inverse and Direct conventions
- Re-transmission of byte upon Smart Card NACK request with programmable threshold of re-transmissions
- Auto detection of Initial Character in receiver and setting of data format (inverse or direct)
- NACK detection in receiver
- Independent timers to measure character wait time, block wait time and block guard time
- Two general purpose counters available for use by software application with programmable clock selection for the counters
- DMA support available to transfer data to/from FIFOs. Programmable option available to select interrupt or DMA feature
- Programmable Prescaler to generate the desired frequency for Card Clock and Baud Rate Divisor to generate the internal ETU clocks for transmitter and receiver for any F/D ratio
- Deep sleep wake-up via Smart Card presence detect interrupt
- Manual control of all Smart Card interface signals
- Automatic power down of port logic on Smart Card presence detect
- Support for 8-bit LRC and 16-bit CRC generation for bytes sent out from transmitter and checking incoming message checksum for receiver

## 60.3 Block Diagram

Figure below shows the block diagram for the EMV SIM module.

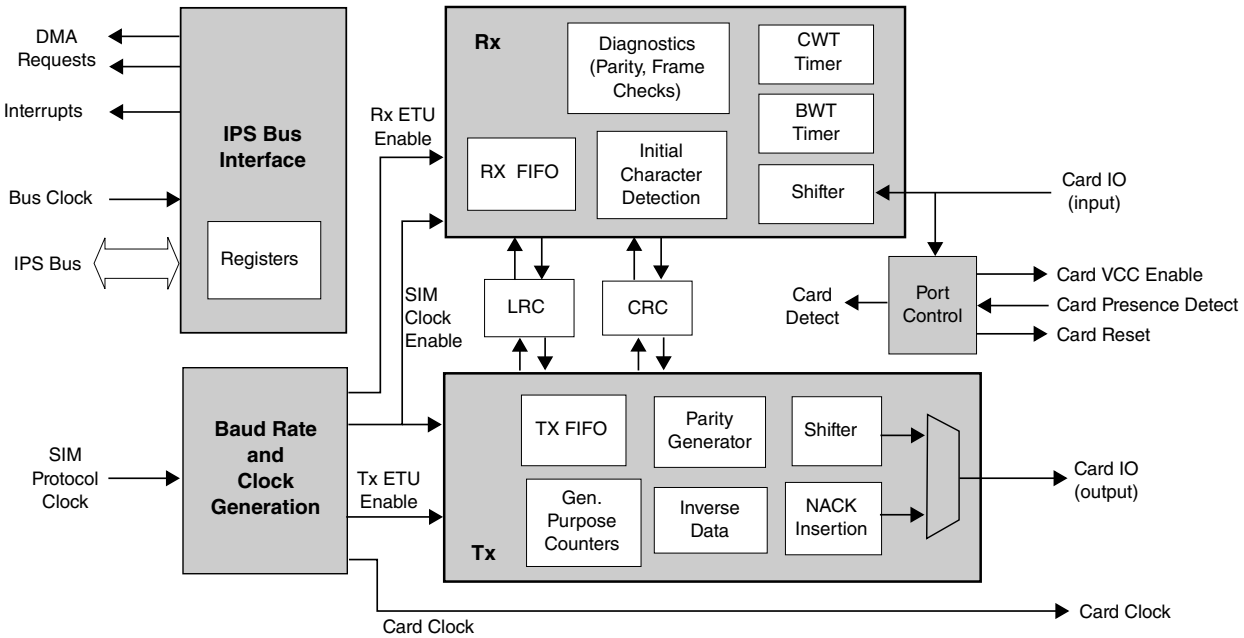



**Figure 60-1. EMV SIM Block Diagram**

## 60.4 Design Overview

The EMV SIM is designed to be compliant to the EMV ICC Specifications ver. 4.3 (dated Nov 2011) and the ISO-7816-3 standard.

The EMV SIM module is a block that is configured via the device's peripheral bus interface. The module has 32-bit aligned memory mapped registers that are used to configure and control the operations of the module. The software can write the transmit bytes or read the received bytes using either interrupt operation or DMA operation. The module can generate necessary DMA requests for the operation.

The module interfaces to the external card (Smart Card) via card clock, card reset, card $V_{CC}$ enable and IO pins. The transmitter output from the module and the receiver input to the module are connected to the same IO pin and used in the open drain configuration. A pull-up resistor must be connected to the IO pin on board to allow proper functioning.

The key functional blocks in the module are clocking, transmitter and receiver which control the interactions with the Smart Card. These blocks function on a clock that is different and asynchronous to the register read/write clock. In order for the transmitter and receiver to function properly, the respective configuration must be done before enabling the transmitter or receiver.

The clocking module generates the card clock (output to Smart Card), and the internal clocks for the transmitter and receiver. The internal clocks control the ETU period and the sampling and driving of data on the IO line. The receive operation uses an internal over-sampled clock that oversamples the received input at 16x times the ETU rate. The prescaler generates the card clock in the range of 1 to 5 MHz with near 50% duty cycle and the baud divider generates the internal ETU clocks for transmitter and receiver.

The transmitter comprises of a 16-byte deep FIFO to store the bytes to be sent to the Smart Card. These bytes are sent to the Smart Card serially by the transmitter. The data format (i.e. inverse or direct conversion) is supported and the byte is converted into the right data format before being shifted out serially. The transmitter also supports retransmission of the current byte on the reception of a NACK. The transmitter automatically inserts NACK when the receiver indicates a parity or FIFO overflow error, if configured to do so by software. The transmitter is not required to be enabled for this. The transmitter also includes timer to insert the programmable guard time between the transmitter bytes and general purpose timers to allow software performs certain measurements. The status of the transmitter operation is intimated to software via maskable interrupts.

The receiver comprises of a 16-byte deep FIFO to store the correctly received bytes from the Smart Card. Software must ensure that it reads out the data from FIFO to prevent any overflow when more bytes are sent from the card. The receiver can be configured to detect the initial character. When configured to do so, the data format (direct or inverse convention) is also determined by the receiver and appropriate bit is set in the register map. Internal timers in the receiver measure the various wait times (Character, Block and Guard). On reception of a byte, the receiver checks the byte for Parity Error (in T=0 mode) and Framing Error. It can then direct the transmitter to insert NACK on parity error detection. For a block of bytes received, the receiver checks for LRC or CRC errors (in T=1 mode). Other errors are stored as status for software to be aware of their occurrence. The status of the receiver operation is intimated to software via maskable interrupts.

In addition to the above blocks, the module also contains the 8-bit LRC and the 16-bit CRC blocks that are common to the transmitter and receiver. Both transmitter and receiver can use the LRC and CRC blocks and can insert the LRC or CRC into the byte stream or check the validity of the input stream using the LRC and CRC values received. Since transmit and receive operations are mutually exclusive, common LRC and CRC blocks for the transmitter and receiver can be used.

The Smart Card interface signals, except IO, which is controlled by the transmitter and receiver; are generated by the port control block which controls the assertion and de-assertion of the clock, reset and VCC signals to Smart Card. It can do so manually (software needs to explicitly write the necessary bits) or using auto power down wherein

the module can itself power down the Smart Card when configured to do so by software. The port control block can also detect the Smart Card presence and generate necessary interrupts to indicate the same.

## 60.5 Signal Description

The following table shows the user-accessible signals available on EMV SIM. See the chip-level specification to find out which signals are connected to the external pins.

**Table 60-1. EMV SIM Signal Description**

| Signal | Description | I/O |
|---|---|---|
| EMVSIM_SCLK | Card Clock. Clock to Smart Card. | O |
| EMVSIM_SRST | Card Reset. Reset signal to Smart Card | O |
| EMVSIM_VCC_EN | Card Power Enable. This signal controls the power to Smart Card | O |
| EMVSIM_IO | Card Data Line. Bi-directional data line. | I/O |
| EMVSIM_PD | Card Presence Detect. Signal indicating presence or removal of card | I |

## 60.6 Memory Map and Registers

### 60.6.1 EMVSIM register descriptions

The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s.The module will not check for correctness of programmed values in the registers and software must ensure that correct values are being written.

#### 60.6.1.1 EMVSIM Memory map

EMVSIM0 base address: 4003_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID Register (VER_ID) | 32 | RO | 0000_0000h |
| 4h | Parameter Register (PARAM) | 32 | RO | 0000_1010h |
| 8h | Clock Configuration Register (CLKCFG) | 32 | RW | 0000_0000h |
| Ch | Baud Rate Divisor Register (DIVISOR) | 32 | RW | 0000_0174h |
| 10h | Control Register (CTRL) | 32 | RW | 0100_0006h |
| 14h | Interrupt Mask Register (INT_MASK) | 32 | RW | 0000_FFFFh |
| 18h | Receiver Threshold Register (RX_THD) | 32 | RW | 0000_0001h |
| 1Ch | Transmitter Threshold Register (TX_THD) | 32 | RW | 0000_000Fh |
| 20h | Receive Status Register (RX_STATUS) | 32 | W1C | 0000_0000h |
| 24h | Transmitter Status Register (TX_STATUS) | 32 | W1C | 0000_00B8h |
| 28h | Port Control and Status Register (PCSR) | 32 | RW | 0100_0000h |
| 2Ch | Receive Data Read Buffer (RX_BUF) | 32 | RO | 0000_0000h |
| 30h | Transmit Data Buffer (TX_BUF) | 32 | WORZ | 0000_0000h |
| 34h | Transmitter Guard ETU Value Register (TX_GETU) | 32 | RW | 0000_0000h |
| 38h | Character Wait Time Value Register (CWT_VAL) | 32 | RW | 0000_FFFFh |
| 3Ch | Block Wait Time Value Register (BWT_VAL) | 32 | RW | FFFF_FFFFh |
| 40h | Block Guard Time Value Register (BGT_VAL) | 32 | RW | 0000_0000h |
| 44h | General Purpose Counter 0 Timeout Value Register (GPCNT0_VAL) | 32 | RW | 0000_FFFFh |
| 48h | General Purpose Counter 1 Timeout Value (GPCNT1_VAL) | 32 | RW | 0000_FFFFh |

## 60.6.1.2 Version ID Register (VER_ID)

### 60.6.1.2.1 Offset

| Register | Offset |
|---|---|
| VER_ID | 0h |

### 60.6.1.2.2 Function

Version ID for the IP. It corresponds to the version of IP being used.

#### 60.6.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 60.6.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>VER | Version ID of the module<br>EMV SIM Version Number used on the chip<br>31:16 - Major Version (example: 01.00)<br>15:0 - Minor Version (example: 01.00) |

### 60.6.1.3 Parameter Register (PARAM)

#### 60.6.1.3.1 Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### 60.6.1.3.2 Function

This register provides details on the parameter settings that were used while including this module in the chip.

### 60.6.1.3.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_FIFO_DEPTH | | | | | | | | RX_FIFO_DEPTH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### 60.6.1.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_FIFO_DEPTH | Transmit FIFO Depth<br>Value of parameter for Transmit FIFO Depth (in Bytes) |
| 7-0<br>RX_FIFO_DEPTH | Receive FIFO Depth<br>Value of parameter for Receive FIFO Depth (in Bytes) |

## 60.6.1.4 Clock Configuration Register (CLKCFG)

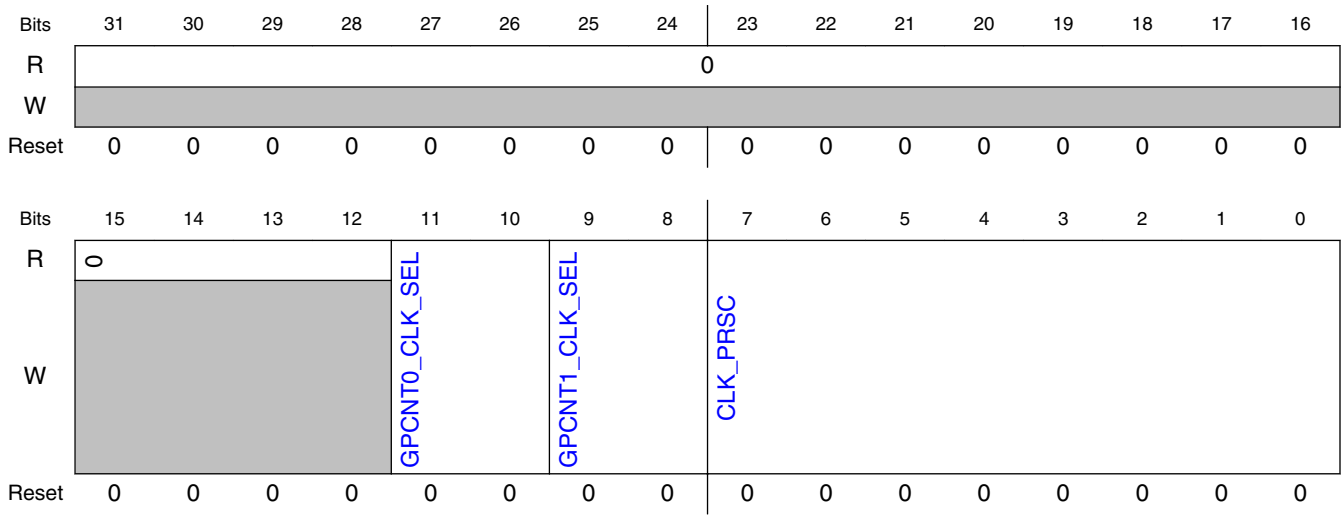
### 60.6.1.4.1 Offset

| Register | Offset |
|---|---|
| CLKCFG | 8h |

### 60.6.1.4.2 Function

This register provides configuration details to enable the right clock frequency of clocks used by EMV SIM module.

### 60.6.1.4.3 Diagram




### 60.6.1.4.4 Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-10<br>GPCNT0_CLK_SEL | General Purpose Counter 0 Clock Select<br>Selects which clock source is used by EMV SIM Module general purpose counter 0. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other register bits of the EMV SIM module (KILL_CLOCK, RCV_EN, and XMT_EN). Counter is active while RCV_EN or XMT_EN are set.<br>00b - Disabled / Reset (default)<br>01b - Card Clock<br>10b - Receive Clock<br>11b - ETU Clock (transmit clock) |
| 9-8<br>GPCNT1_CLK_SEL | General Purpose Counter 1 Clock Select<br>Selects which clock source is used by EMV SIM Module general purpose counter 1. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are active. These input clocks are enabled through other register bits of the EMV SIM module (KILL_CLOCK, RCV_EN, and XMT_EN). Counter is active while RCV_EN or XMT_EN are set.<br>00b - Disabled / Reset (default)<br>01b - Card Clock<br>10b - Receive Clock<br>11b - ETU Clock (transmit clock) |
| 7-0<br>CLK_PRSC | Clock Prescaler Value<br>The value written to this register will determine the desired card clock frequency. The Card Clock frequency is the EMV SIM Protocol (or logic) clock divided by this prescaler value. The prescaler value should be updated only when card clock is disabled and transmitter and receiver are not operational.<br>00000010b - Divide by 2 |

## 60.6.1.5 Baud Rate Divisor Register (DIVISOR)

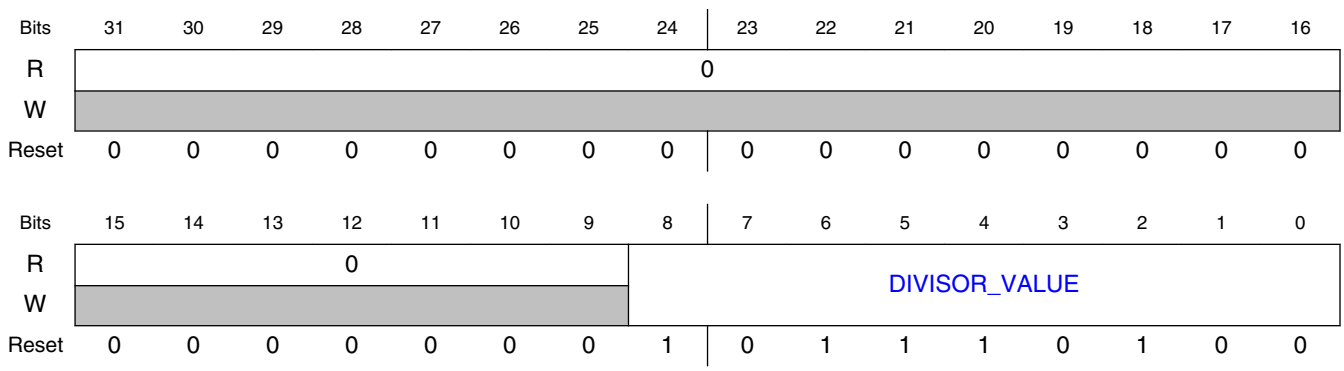
### 60.6.1.5.1 Offset

| Register | Offset |
|---|---|
| DIVISOR | Ch |

### 60.6.1.5.2 Function

This regiter configures the divisor value to generate the baud clock which will drive the card clock and also generate the transmit and receive clocks and respective ETUs.

### 60.6.1.5.3 Diagram




### 60.6.1.5.4 Fields

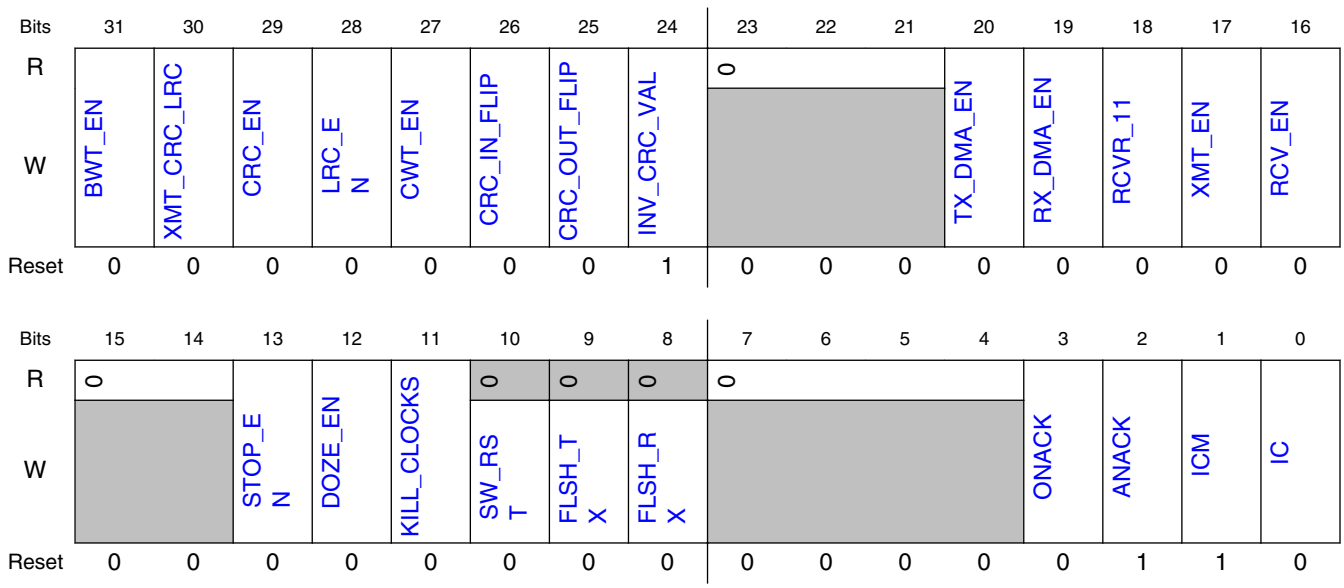
| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-0<br>DIVISOR_VALUE | Divisor (F/D) Value<br>The value written to this register will be used to generate the ETU bit period that will be used by the transmitter and receiver. The divisor value is the integer result of F/D value; where F and D are the clock rate conversion integer and baud rate adjustment integer, respectively, that required for the desired operation. The value in this register can be changed when no transaction is active with the smart card.<br>When programming this field, the fractional part of F/D should be rounded off the nearest integer.<br>000000000-000000100b - Invalid. As per ISO 7816 specification, minimum value of F/D is 5<br>101110100b - Divisor value for F = 372 and D = 1 (default) |

## 60.6.1.6 Control Register (CTRL)

### 60.6.1.6.1 Offset

| Register | Offset |
|---|---|
| CTRL | 10h |

### 60.6.1.6.2 Diagram




### 60.6.1.6.3 Fields

| Field | Function |
|---|---|
| 31<br>BWT_EN | Block Wait Time Counter Enable<br>Writing a '1' to this bit will enable the BWT and BGT functions. The BWT and BGT functions can then be individually selected using the interrupt mask.<br>0b - Disable BWT, BGT Counters (default)<br>1b - Enable BWT, BGT Counters |
| 30<br>XMT_CRC_LRC | Transmit CRC or LRC Enable<br>This bit specifies whether or not to transmit the redundancy checking data (LRC or CRC) at the end of a transmission (that is, when the FIFO becomes empty).<br>0b - No CRC or LRC value is transmitted (default)<br>1b - Transmit LRC or CRC info when FIFO empties (whichever is enabled) |
| 29 | CRC Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CRC_EN | This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the CRC_OK bit in the RX_STATUS register. Clearing this bit resets the current CRC residual value in the EMV SIM hardware.<br>0b - 16-bit Cyclic Redundancy Checking disabled (default)<br>1b - 16-bit Cyclic Redundancy Checking enabled |
| 28<br>LRC_EN | LRC Enable<br>This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the LRC_OK bit in the RX_STATUS register. Clearing this bit resets the current LRC value in the EMV SIM hardware.<br>0b - 8-bit Linear Redundancy Checking disabled (default)<br>1b - 8-bit Linear Redundancy Checking enabled |
| 27<br>CWT_EN | Character Wait Time Counter Enable<br>Enables the character wait time counter. Clearing this bit resets the counter to zero.<br>0b - Character Wait time Counter is disabled (default)<br>1b - Character Wait time counter is enabled |
| 26<br>CRC_IN_FLIP | CRC Input Byte's Bit Reversal or Flip Control<br>This bit control whether the bits in the CRC input byte will be reversed before CRC calculation or not. When set to '1', the bits within the input byte for CRC will change from 7:0 to 0:7. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly.<br>0b - Bits in the input byte will not be reversed (i.e. 7:0 will remain 7:0) before the CRC calculation (default)<br>1b - Bits in the input byte will be reversed (i.e. 7:0 will become 0:7) before CRC calculation |
| 25<br>CRC_OUT_FLIP | CRC Output Value Bit Reversal or Flip<br>This bit control whether the bits in the CRC output bytes will be reversed or not. When set to '1', the bits within the two bytes for CRC Output will change from 15:0 to {8:15,0:7}. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly.<br>0b - Bits within the CRC output bytes will not be reversed i.e. 15:0 will remain 15:0 (default)<br>1b - Bits within the CRC output bytes will be reversed i.e. 15:0 will become {8:15,0:7} |
| 24<br>INV_CRC_VAL | Invert bits in the CRC Output Value<br>This bit control whether the bits within the CRC Output value will be inverted (1's complement) or not. For CCITT CRC calculation, this bit should be set to '1'. For any other CRC using same polynomial, this bit can be set accordingly.<br>0b - Bits in CRC Output value will not be inverted.<br>1b - Bits in CRC Output value will be inverted. (default) |
| 23-21<br>— | Reserved |
| 20<br>TX_DMA_EN | Transmit DMA Enable<br>Enables assertion of DMA write request when Transmit FIFO is empty. Request is held asserted till Transmit FIFO reaches the programmed data threshold value. Transmit Data Threshold Interrupt will not be generated when this bit is asserted.<br>0b - No DMA Write Request asserted for Transmitter (default)<br>1b - DMA Write Request asserted for Transmitter |
| 19<br>RX_DMA_EN | Receive DMA Enable<br>Enables assertion of DMA read request when Receive FIFO reaches the programmed data threshold value. Request is held asserted till all the programmed threshold bytes are read out. Receiver Data Interrupt will not be generated when this bit is asserted.<br>0b - No DMA Read Request asserted for Receiver (default) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - DMA Read Request asserted for Receiver |
| 18<br>RCVR_11 | Receiver 11 ETU Mode Enable<br>Used to configure the EMV SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards.<br>0b - Receiver configured for 12 ETU operation mode (default)<br>1b - Receiver configured for 11 ETU operation mode |
| 17<br>XMT_EN | Transmitter Enable<br>Used to enable/disable the EMV SIM transmitter block. It can be set to 0 during the auto power down sequence.<br>**Note:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values.<br>0b - EMV SIM Transmitter disabled (default)<br>1b - EMV SIM Transmitter enabled |
| 16<br>RCV_EN | Receiver Enable<br>Used to enable/disable the EMV SIM receiver block. Once the transmitter has completed its operation, the software must enable the receiver using this bit. It can be set to 0 during the auto power down sequence.<br>0b - EMV SIM Receiver disabled (default)<br>1b - EMV SIM Receiver enabled |
| 15-14<br>— | Reserved |
| 13<br>STOP_EN | STOP Enable<br>Used to configure the operation of the EMV SIM module when a processor STOP instruction is executed. This bit is added to provide support for Smart Cards that do not allow the Smart Card clock to be stopped while power is applied.<br>0b - STOP instruction shuts down all EMV SIM clocks (default)<br>1b - STOP instruction shuts down all clocks except for the Smart Card Clock (SCK) (clock provided to Smart Card) |
| 12<br>DOZE_EN | Doze Enable<br>Used to configure the operation of the EMV SIM module when a processor DOZE instruction is executed.<br>0b - DOZE instruction will gate all internal EMV SIM clocks as well as the Smart Card clock when the transmit FIFO is empty (default)<br>1b - DOZE instruction has no effect on EMV SIM module |
| 11<br>KILL_CLOCKS | Kill all internal clocks<br>Used to enable/disable the clock input to the EMV SIM module. This bit will gate all clocks including the Smart Card clock regardless of the state of the STOP bit described above.<br>**Note:** This bit will have no effect on the register read write clock. Only EMV SIM logic clock is gated.<br>0b - EMV SIM input clock enabled (default)<br>1b - EMV SIM input clock is disabled |
| 10<br>SW_RST | Software Reset Bit<br>Used to reset the entire EMV SIM module. This acts the same as a hardware reset for the EMV SIM module. This bit is self-clearing and always reads 0.<br>**Note:** Software should allow a minimum of 4 Protocol clock cycles before attempting to access the EMV SIM module after a software reset.<br>0b - EMV SIM Normal operation (default)<br>1b - EMV SIM held in Reset |

*Table continues on the next page...*

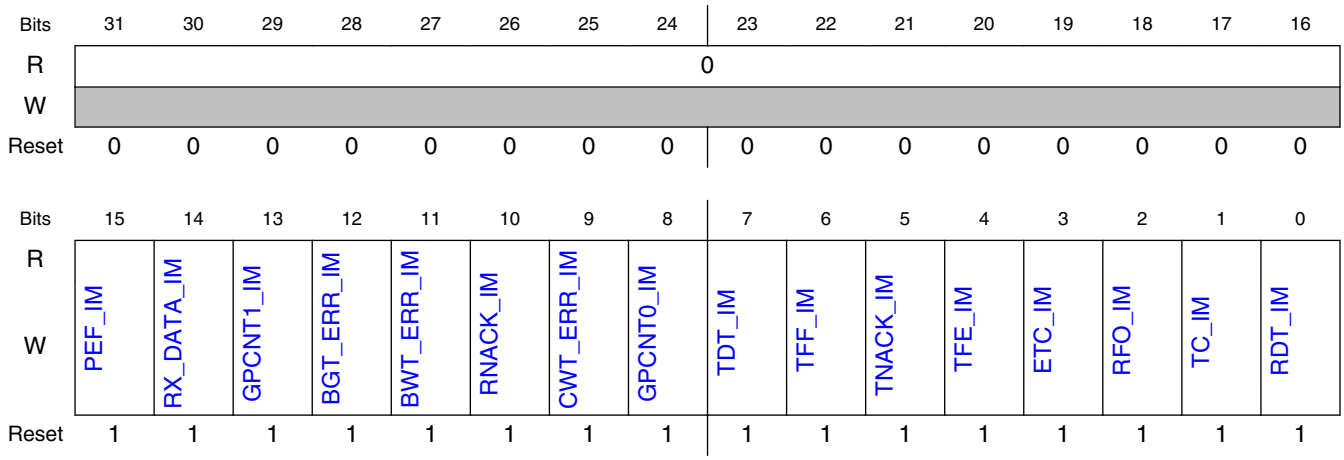
| Field | Function |
|---|---|
| 9<br>FLSH_TX | Flush Transmitter Bit<br>This bit operates as an EMV SIM transmitter reset. The receive portion of the EMV SIM module is not affected. This bit clears automatically and always reads 0.<br>0b - EMV SIM Transmitter normal operation (default)<br>1b - EMV SIM Transmitter held in Reset |
| 8<br>FLSH_RX | Flush Receiver Bit<br>This bit operates as an EMV SIM receiver reset. The transmit portion of the EMV SIM module is not affected. This bits clears automatically and always reads 0.<br>0b - EMV SIM Receiver normal operation (default)<br>1b - EMV SIM Receiver held in Reset |
| 7-4<br>— | Reserved |
| 3<br>ONACK | Overrun NACK Enable<br>Enables NACK generation when Rx FIFO is full and another message is ready for writing into the FIFO.<br>0b - NACK generation on overrun is disabled (default)<br>1b - NACK generation on overrun is enabled |
| 2<br>ANACK | Auto NACK Enable<br>Enables NACK generation for parity errors in receieved messages or when invalid initial characters are received in ICM mode.<br>0b - NACK generation on errors disabled<br>1b - NACK generation on errors enabled (default) |
| 1<br>ICM | Initial Character Mode<br>Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received.<br>0b - Initial Character Mode disabled<br>1b - Initial Character Mode enabled (default) |
| 0<br>IC | Inverse Convention<br>Used to configure the EMV SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode.<br>0b - Direction convention transfers enabled (default)<br>1b - Inverse convention transfers enabled |

## 60.6.1.7 Interrupt Mask Register (INT_MASK)

### 60.6.1.7.1 Offset

| Register | Offset |
|---|---|
| INT_MASK | 14h |

### 60.6.1.7.2 Diagram




### 60.6.1.7.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>PEF_IM | Parity Error Interrupt Mask<br>Used to enable/disable the ability of the PEF flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - PEF interrupt enabled<br>1b - PEF interrupt masked (default) |
| 14<br>RX_DATA_IM | Receive Data Interrupt Mask<br>Used to enable/disable the ability of the RX_DATA flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - RX_DATA interrupt enabled<br>1b - RX_DATA interrupt masked (default) |
| 13<br>GPCNT1_IM | General Purpose Counter 1 Timeout Interrupt Mask<br>Used to enable/disable the ability of the GPCNT1_TO flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - GPCNT1_TO interrupt enabled<br>1b - GPCNT1_TO interrupt masked (default) |
| 12<br>BGT_ERR_IM | Block Guard Time Error Interrupt<br>Used to enable/disable the ability of the BGT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - BGT_ERR interrupt enabled<br>1b - BGT_ERR interrupt masked (default) |
| 11<br>BWT_ERR_IM | Block Wait Time Error Interrupt Mask<br>Used to enable/disable the ability of the BWT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - BWT_ERR interrupt enabled<br>1b - BWT_ERR interrupt masked (default) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>RNACK_IM | Receiver NACK Threshold Interrupt Mask<br>Used to enable/disable the ability of the RTE flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - RTE interrupt enabled<br>1b - RTE interrupt masked (default) |
| 9<br>CWT_ERR_IM | Character Wait Time Error Interrupt Mask<br>Used to enable/disable the ability of the CWT_ERR flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - CWT_ERR interrupt enabled<br>1b - CWT_ERR interrupt masked (default) |
| 8<br>GPCNT0_IM | General Purpose Timer 0 Timeout Interrupt Mask<br>Used to enable/disable the ability of the GPCNT0_TO flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - GPCNT0_TO interrupt enabled<br>1b - GPCNT0_TO interrupt masked (default) |
| 7<br>TDT_IM | Transmit Data Threshold Interrupt Mask<br>Used to enable/disable the ability of the TDTF flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - TDTF interrupt enabled<br>1b - TDTF interrupt masked (default) |
| 6<br>TFF_IM | Transmit FIFO Full Interrupt Mask<br>Used to enable/disable the ability of the TFF flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - TFF interrupt enabled<br>1b - TFF interrupt masked (default) |
| 5<br>TNACK_IM | Transmit NACK Threshold Interrupt Mask<br>Used to enable/disable the ability of the TNTE flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - TNTE interrupt enabled<br>1b - TNTE interrupt masked (default) |
| 4<br>TFE_IM | Transmit FIFO Empty Interrupt Mask<br>Used to enable/disable the ability of the TFE flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - TFE interrupt enabled<br>1b - TFE interrupt masked (default) |
| 3<br>ETC_IM | Early Transmit Complete Interrupt Mask<br>Used to enable/disable the ability of the ETC flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - ETC interrupt enabled<br>1b - ETC interrupt masked (default) |
| 2<br>RFO_IM | Receive FIFO Overflow Interrupt Mask<br>Used to enable/disable the ability of the RFO flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - RFO interrupt enabled<br>1b - RFO interrupt masked (default) |

*Table continues on the next page...*

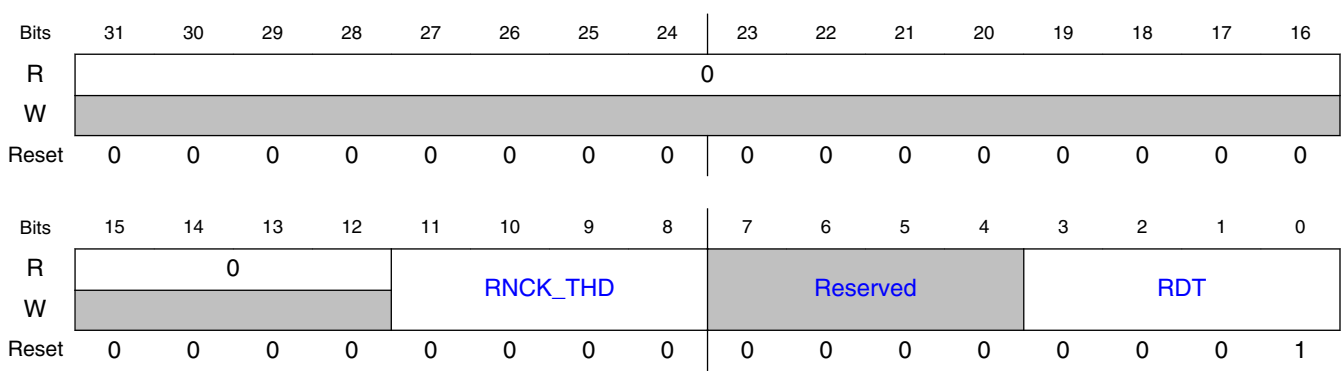
| Field | Function |
|---|---|
| 1<br>TC_IM | Transmit Complete Interrupt Mask<br>Used to enable/disable the ability of the TCF flag in the TX_STATUS register to generate EMV SIM interrupts.<br>0b - TCF interrupt enabled<br>1b - TCF interrupt masked (default) |
| 0<br>RDT_IM | Receive Data Threshold Interrupt Mask<br>Used to enable/disable the ability of the RDTF flag in the RX_STATUS register to generate EMV SIM interrupts.<br>0b - RDTF interrupt enabled<br>1b - RDTF interrupt masked (default) |

## 60.6.1.8 Receiver Threshold Register (RX_THD)

### 60.6.1.8.1 Offset

| Register | Offset |
|---|---|
| RX_THD | 18h |

### 60.6.1.8.2 Diagram




### 60.6.1.8.3 Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-8 | Receiver NACK Threshold Value |

*Table continues on the next page...*

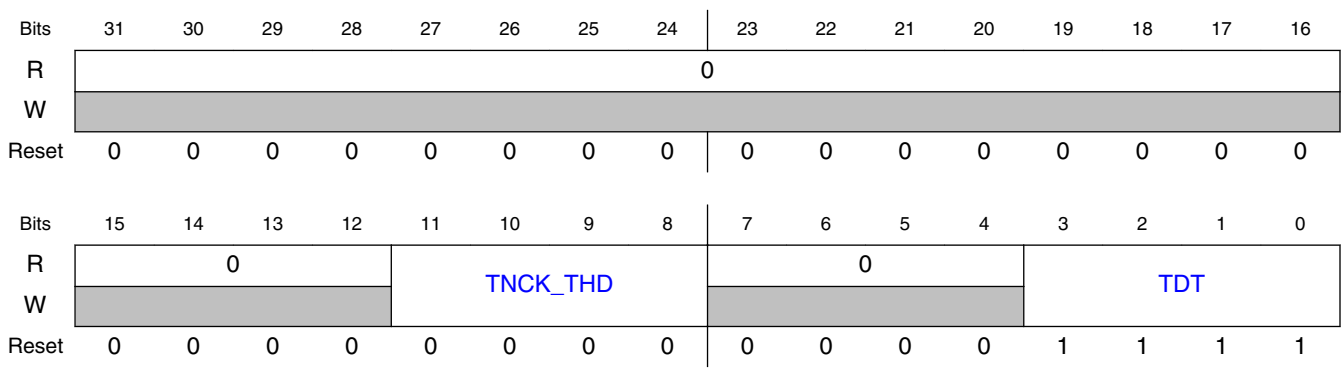
| Field | Function |
|---|---|
| RNCK_THD | Used to specify the number of consecutive NACK's transmitted by the EMV SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the EMV SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RNCK_THD has no effect.<br>0000b - Zero Threshold. RTE will not be set |
| 7-4<br>— | Reserved |
| 3-0<br>RDT | Receiver Data Threshold Value<br>Determines the number of bytes that must exist in the Receive FIFO to trigger the receive data threshold interrupt flag (RDTF). Value must not exceed the total bytes that can be stored in the Receive FIFO. |

## 60.6.1.9 Transmitter Threshold Register (TX_THD)

### 60.6.1.9.1 Offset

| Register | Offset |
|---|---|
| TX_THD | 1Ch |

### 60.6.1.9.2 Diagram




### 60.6.1.9.3 Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-8 | Transmitter NACK Threshold Value |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TNCK_THD | Used to set the NACK threshold for the transmitter. Once the threshold number set by TNCK_THD has been reached for the current byte being transmitted, the error flag TNTE in the TX_STATUS register will be set. Setting of TNTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears TNTE. To trigger TNTE, a given byte being transmitted must reach the TNCK_THD threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next.<br>0000b - TNTE will never be set; retransmission after NACK reception is disabled.<br>0001b - TNTE will be set after 1 nack is received; 0 retransmissions occurs.<br>0010b - TNTE will be set after 2 nacks are received; at most 1 retransmission occurs.<br>0011b - TNTE will be set after 3 nacks are received; at most 2 retransmissions occurs.<br>1111b - TNTE will be set after 15 nacks are received; at most 14 retransmissions occurs. |
| 7-4<br>— | Reserved |
| 3-0<br>TDT | Transmitter Data Threshold Value<br>Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the TX_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set. |

## 60.6.1.10 Receive Status Register (RX_STATUS)

### 60.6.1.10.1 Offset

| Register | Offset |
|---|---|
| RX_STATUS | 20h |

### 60.6.1.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RX_CNT | | | | 0 | | | | RX_WPTR | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | FEF | PEF | BGT_ERR | BWT_ERR | RTE | CWT_ERR | CRC_OK | LRC_OK | RDTF | RX_DATA | 0 | | | RFO |
| W | | | W1C | W1C | W1C | W1C | W1C | W1C | | | | W1C | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 60.6.1.10.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>RX_CNT | Receive FIFO Byte Count<br>These bits indicate the number of bytes stored in the receive FIFO.<br>0000b - FIFO is emtpy |
| 23-20<br>— | Reserved |
| 19-16<br>RX_WPTR | Receive FIFO Write Pointer Value<br>Value of write pointer of Receive FIFO |
| 15-14<br>— | Reserved |
| 13<br>FEF | Frame Error Flag<br>Used to indicate if the current received byte did not have proper STOP bits. This will not cause a NACK transmission.<br>This bit will not cause an interrupt assertion. It is simply an indication that a framing error has occured.<br>0b - No frame error detected<br>1b - Frame error detected |
| 12<br>PEF | Parity Error Flag<br>Used to indicate if a received byte had a parity error or not. A parity error will cause transmission of NACK if ANACK bit is set in CTRL register and PEF bit will not be asserted. When ANACK is *not* set, PEF will be asserted.<br>This bit will also not assert when ICM bit is set; however parity is checked in ICM = 1 mode. It is simply an indication that a parity error has occured.<br>0b - No parity error detected<br>1b - Parity error detected |
| 11<br>BGT_ERR | Block Guard Time Error Flag<br>Used to indicate if the block guard time was too small. The threshold is set by the block guard time register.<br>0b - Block guard time was sufficient<br>1b - Block guard time was too small |
| 10<br>BWT_ERR | Block Wait Time Error Flag<br>Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers.<br>0b - Block wait time not exceeded<br>1b - Block wait time was exceeded |
| 9<br>RTE | Received NACK Threshold Error Flag<br>Used to indicate whether the number of consecutive NACK's generated by the EMV SIM module in response to receive parity errors, for the byte being received, equals the value programmed in the RTH[3:0] in the RX_THRESHOLD register. This bit will never set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write once to clear bit. Clearing this bit also resets the internal counter for consecutive NACK's being transmitted for a given byte.<br>0b - Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0] |
| 8<br>CWT_ERR | Character Wait Time Error Flag<br>Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register.<br>0b - No CWT violation has occurred (default).<br>1b - Time between two consecutive characters has exceeded the value in CHAR_WAIT. |
| 7<br>CRC_OK | CRC Check OK Flag<br>Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRC_EN bit is set in the CTRL register. The current CRC residual can be reset by three mechanisms:<br>• Clear CRC_EN bit in CTRL register<br>• Set XMT_EN bit in ENABLE register<br>• Automatically by hardware when ETC flag is set at the end of a transmission.<br>0b - Current CRC value does not match remainder.<br>1b - Current calculated CRC value matches the expected result. |
| 6<br>LRC_OK | LRC Check OK Flag<br>Used to indicate when the calculated 8-bit LRC value is correct for the current input data stream. The value is calculated across all received characters from the point the LRC_EN bit is set in the CTRL register. The current LRC residual can be reset by three mechanisms:<br>• Clear LRC_EN bit in CTRL register<br>• Set XMT_EN bit in ENABLE register<br>• Automatically by hardware when ETC flag is set at the end of a transmission.<br>0b - Current LRC value does not match remainder.<br>1b - Current calculated LRC value matches the expected result (i.e. zero). |
| 5<br>RDTF | Receive Data Threshold Interrupt Flag<br>Interrupt flag asserted when total bytes in the Receive FIFO equal or is greater than the programmed receive threshold RDT[3:0]. The RDTF flag will be set any time the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT[3:0].<br>The flag can be cleared by reading enough bytes out of the receive FIFO so as to bring the number of bytes left in the FIFO below the RDT[3:0] level. Another way to clear the flag is to set the RDT[3:0] level higher than the number of unread bytes currently in the FIFO.<br>The RDTF flag will create an interrupt if the RDT_IM bit in the INT_MASK register is cleared.<br>0b - Number of unread bytes in receive FIFO less than the value set by RDT[3:0] (default).<br>1b - Number of unread bytes in receive FIFO greater or than equal to value set by RDT[3:0]. |
| 4<br>RX_DATA | Receive Data Interrupt Flag<br>Interrupt asserted when a new data byte is received and entered into the Receive FIFO.<br>0b - No new byte is received<br>1b - New byte is received ans stored in Receive FIFO |
| 3-1<br>— | Reserved |
| 0<br>RFO | Receive FIFO Overflow Flag<br>Used to indicate that the EMV SIM was unable to store received data due to already unread bytes in the FIFO (FIFO full or almost full). It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the Smart Card which implies that no data has actually been lost. In this case, the RFO flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set RFO flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost |

| Field | Function |
|---|---|
| | (including the byte that caused the bit to be set). The RFO flag will cause an interrupt if the RFO_IM bit in the INT_MASK register. The RFO flag is a write-one-to-clear bit.<br>0b - No overrun error has occurred (default)<br>1b - A byte was received when the received FIFO was already full |

## 60.6.1.11 Transmitter Status Register (TX_STATUS)

### 60.6.1.11.1 Offset

| Register | Offset |
|---|---|
| TX_STATUS | 24h |

### 60.6.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TX_CNT | | | | 0 | | | | TX_RPTR | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | GPCNT1_TO | GPCNT0_TO | TDTF | TFFF | TCF | ETCF | TFE | 0 | | TNTE |
| W | | | | | | | W1C | W1C | | W1C | W1C | W1C | W1C | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

### 60.6.1.11.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>TX_CNT | Transmit FIFO Byte Count<br>These bits indicate the number of bytes stored in the transmit FIFO.<br>0000b - FIFO is emtpy |
| 23-20 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 19-16<br>TX_RPTR | Transmit FIFO Read Pointer<br>Value of the read pointer of transmit FIFO. |
| 15-10<br>— | Reserved |
| 9<br>GPCNT1_TO | General Purpose Counter 1 Timeout Flag<br>Used to indicate when the General Purpose Counter 1 has reached the GPCNT1_VAL value in the GPCNT_VAL register.<br>0b - GPCNT1_VAL time not reached, or bit has been cleared. (default)<br>1b - General Purpose counter has reached the GPCNT1_VAL value |
| 8<br>GPCNT0_TO | General Purpose Counter 0 Timeout Flag<br>Used to indicate when the General Purpose Counter 0 has reached the GPCNT0_VAL value in the GPCNT_VAL register.<br>0b - GPCNT0_VAL time not reached, or bit has been cleared. (default)<br>1b - General Purpose counter has reached the GPCNT0_VAL value |
| 7<br>TDTF | Transmit Data Threshold Flag<br>Interrupt flag asserted when total bytes in the Transmit FIFO is less than or equal to the programmed transmit data threshold TDT[3:0]. The TDTF flag will be set any time the number of bytes in the transmit FIFO is less than or equal to the value set by TDT[3:0]. The flag can be cleared by writing enough bytes into the transmit FIFO so as to take the number of bytes in the FIFO above the TDT[3:0] level. Another way to clear the flag is to set the TDT[3:0] level lower than the number of bytes currently in the FIFO.The TDTF flag will create an interrupt if the TDT_IM bit in the INT_MASK register is cleared.<br>0b - Number of bytes in FIFO is greater than TDT[3:0], or bit has been cleared<br>1b - Number of bytes in FIFO is less than or equal to TDT[3:0] (default) |
| 6<br>TFF | Transmit FIFO Full Flag<br>Used to indicate when the Transmit FIFO has been written with maximum number of bytes that it can store. Subsequent writes may get ignored. The TFF bit can only be cleared by writing 1 to this bit or setting the FLUSH_XMT or SOFT_RESET bit in the RESET_CNTL register.<br>0b - Transmit FIFO Full condition has *not* occurred (default)<br>1b - A Transmit FIFO Full condition has occurred |
| 5<br>TCF | Transmit Complete Flag<br>Used to indicate whether the EMV SIM transmitter is ready for a new transmission. The TC flag becomes set when the guard time has expired after the last byte in the transmit FIFO (if XMT_CRC_LRC = 0) or the LRC or CRC byte (if XMT_CRC_LRC bit = 1) has been transmitted. The TC flag will create an interrupt if TC_IM in the INT_MASK register is low. The TC bit is a write-one-to-clear bit.<br>0b - Transmit pending or in progress<br>1b - Transmit complete (default) |
| 4<br>ETCF | Early Transmit Complete Flag<br>Used to indicate that the EMV SIM transmitter has finished sending the last byte in transmit FIFO (if XMT_CRC_LRC = 0) or finished sending the LRC or CRC byte (if XMT_CRC_LRC = 1). This bit differs from the TC bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if ETC_IM in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit.<br>0b - Transmit pending or in progress<br>1b - Transmit complete (default) |
| 3<br>TFE | Transmit FIFO Empty Flag<br>Used to indicate that the EMV SIM transmit FIFO has been emptied. This bit will be set when the last byte in the transmit FIFO has been transferred out of the EMV SIM transmitter to the Smart Card successfully. TFE will also be set when the TNTE flag is set. The TFE flag will create an interrupt if TFE_IM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. |

*Table continues on the next page...*

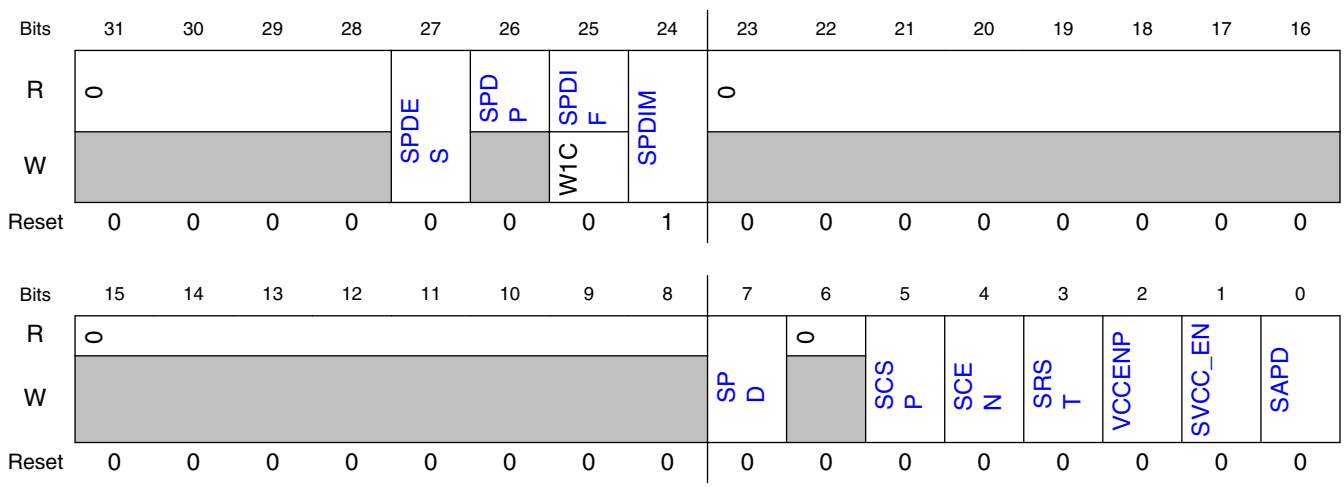
| Field | Function |
|---|---|
| | 0b - Transmit FIFO is not empty<br>1b - Transmit FIFO is empty (default) |
| 2-1<br>— | Reserved |
| 0<br>TNTE | Transmit NACK Threshold Error Flag<br>Used to indicate the transmit NACK threshold has been reached. When TNTE is high, no further transmissions will be done until the TNTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The TNTE flag will create an interrupt if TNACK_IM in the INT_MASK register is low. The TNTE bit is a write-one-to-clear bit.<br>0b - Transmit NACK threshold has not been reached (default)<br>1b - Transmit NACK threshold reached; transmitter frozen |

## 60.6.1.12 Port Control and Status Register (PCSR)

### 60.6.1.12.1 Offset

| Register | Offset |
|---|---|
| PCSR | 28h |

### 60.6.1.12.2 Diagram




### 60.6.1.12.3 Fields

| Field | Function |
|---|---|
| 31-28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27<br>SPDES | SIM Presence Detect Edge Select<br>Controls which edge of the Smart Card Presence Detect pin is used to detect the presence of the Smart Card.<br>0b - Falling edge on the pin (default)<br>1b - Rising edge on the pin |
| 26<br>SPDP | Smart Card Presence Detect Pin Status<br>This bit reflects the state of the Smart Card Presence Detect pin. It is not a latched register bit, but instead a synchronized version of the state of the pin itself.<br>0b - SIM Presence Detect pin is logic low<br>1b - SIM Presence Detectpin is logic high |
| 25<br>SPDIF | Smart Card Presence Detect Interrupt Flag<br>Status flag to indicate that an insertion or removal of a Smart Card has been detected on port. Can create an interrupt to the MCU if SPDIM is low and/or initiate the auto power down sequence if SAPD bit is '1'. Write a '1' to this bit to clear.<br>0b - No insertion or removal of Smart Card detected on Port (default)<br>1b - Insertion or removal of Smart Card detected on Port |
| 24<br>SPDIM | Smart Card Presence Detect Interrupt Mask<br>Interrupt mask for the presence detect interrupt flag (SPDIF).<br>0b - SIM presence detect interrupt is enabled<br>1b - SIM presence detect interrupt is masked (default) |
| 23-8<br>— | Reserved |
| 7<br>SPD | Auto Power Down Control<br>Writing a '1' to this location will start the autopower down sequence provided the auto power down feature is enabled by writing '1' to SAPD bit of this register. The SPD bit will autoclear when power down is complete.<br>0b - No effect (default)<br>1b - Start Auto Powerdown or Power Down is in progress |
| 6<br>— | Reserved |
| 5<br>SCSP | Smart Card Clock Stop Polarity<br>Used to control the polarity of the idle EMV SIM clock when the clock is disabled by SCEN.<br>0b - Clock is logic 0 when stopped by SCEN<br>1b - Clock is logic 1 when stopped by SCEN |
| 4<br>SCEN | Clock Enable for Smart Card<br>Used to enable/disable the clock to the Smart Card. It can be forced low by hardware during the auto power down sequence.<br>0b - Smart Card Clock Disabled<br>1b - Smart Card Clock Enabled |
| 3<br>SRST | Reset to Smart Card<br>Used to control state of reset line to the Smart Card. It can be forced low by hardware during the auto power down sequence. Smart Card reset signals are active low.<br>0b - Smart Card Reset is asserted (default)<br>1b - Smart Card Reset is de-asserted |
| 2<br>VCCENP | VCC Enable Polarity Control |

*Table continues on the next page...*

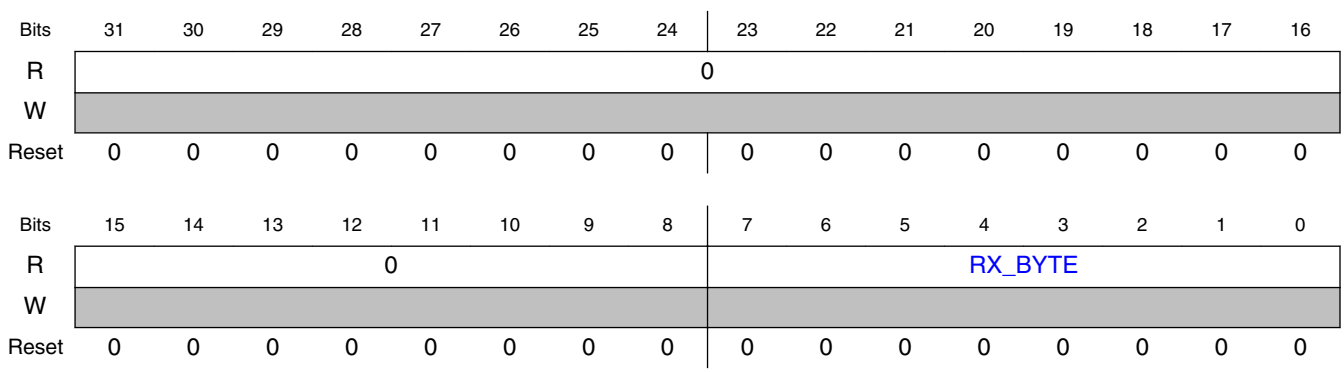
| Field | Function |
|---|---|
| | Used to control the polarity of the SVEN output pad via the SVCC_EN bit. When set to '1', this bit will invert the value of SVCC_EN bit of this register and output to SVEN output pad of device.<br>0b - VCC_EN is active high. Polarity of SVCC_EN is unchanged.<br>1b - VCC_EN is active low. Polarity of SVCC_EN is inverted. |
| 1<br>SVCC_EN | Vcc Enable for Smart Card<br>Used to control the state of the SVEN pin on Smart Card port. The SVEN pin controls the Smart Card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. This bit is XORed with the VCCENP bit to control the polarity of the SVEN output pad.<br>0b - Smart Card Voltage disabled (default)<br>1b - Smart Card Voltage enabled |
| 0<br>SAPD | Auto Power Down Enable<br>Used to enable/disable the auto power down feature. This bit must be set prior to the auto power conditions are met. This bit will be return to 0 at the end of the auto power down sequence. This bit controls the auto power down function which can be initiated by:<br>• Setting the SPD bit in this register by software<br>• Assertion of the RTE bit in the RX_STATUS register<br>• Assertion of the SPDIF bit when the interrupt is used for detecting card removal<br><br>Manual Power Down via software by writing to the SCEN, SRST, and VCC_EN can still be carried out irrespective of the value of this bit.<br>0b - Auto power down disabled (default)<br>1b - Auto power down enabled |

## 60.6.1.13 Receive Data Read Buffer (RX_BUF)

### 60.6.1.13.1 Offset

| Register | Offset |
|---|---|
| RX_BUF | 2Ch |

### 60.6.1.13.2 Diagram



### 60.6.1.13.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>RX_BYTE | Receive Data Byte Read<br>Provides the byte value from the top of the receive FIFO. Each read access to this register will increment the read pointer of the Receive FIFO. |

## 60.6.1.14 Transmit Data Buffer (TX_BUF)

### 60.6.1.14.1 Offset

| Register | Offset |
|---|---|
| TX_BUF | 30h |

### 60.6.1.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | TX_BYTE | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 60.6.1.14.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>TX_BYTE | Transmit Data Byte |

| Field | Function |
|---|---|
| | Write to this register to fill the transmit FIFO with the bytes to be transmitted. The Tx FIFO can be written to only when the transmitter is enabled. Writing to this register while transmitter is disabled will lead to ignoring of all write accesses to this register. Reads to this register will return zeros.<br>**NOTE:** Writing more data to the transmit FIFO than it can hold, will cause a Transmit FIFO Full (TFF) error. |

## 60.6.1.15 Transmitter Guard ETU Value Register (TX_GETU)

### 60.6.1.15.1 Offset

| Register | Offset |
|---|---|
| TX_GETU | 34h |

### 60.6.1.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | GETU | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 60.6.1.15.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>GETU | Transmitter Guard Time Value in ETU<br>Used to control the number of additional Elementary Time Units (ETUs) inserted between bytes transmitted by the EMV SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the EMV SIM receiver. A value of 0x00 inserts no additional ETUs, while a value of 0xFE inserts 254 additional ETUs. A value of 0xFF subtracts one ETU by reducing the number of STOP bits from two to one.<br>00000000b - no additional ETUs inserted (default)<br>00000001b - 1 additional ETU inserted |

| Field | Function |
|---|---|
| | 11111110b - 254 additional ETUs inserted<br>11111111b - Subtracts one ETU by reducing the number of STOP bits from two to one |

## 60.6.1.16 Character Wait Time Value Register (CWT_VAL)

### 60.6.1.16.1 Offset

| Register | Offset |
|---|---|
| CWT_VAL | 38h |

### 60.6.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CWT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 60.6.1.16.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>CWT | Character Wait Time Value<br>The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF |

## 60.6.1.17 Block Wait Time Value Register (BWT_VAL)

### 60.6.1.17.1 Offset

| Register | Offset |
|---|---|
| BWT_VAL | 3Ch |

### 60.6.1.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BWT | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BWT | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 60.6.1.17.3 Fields

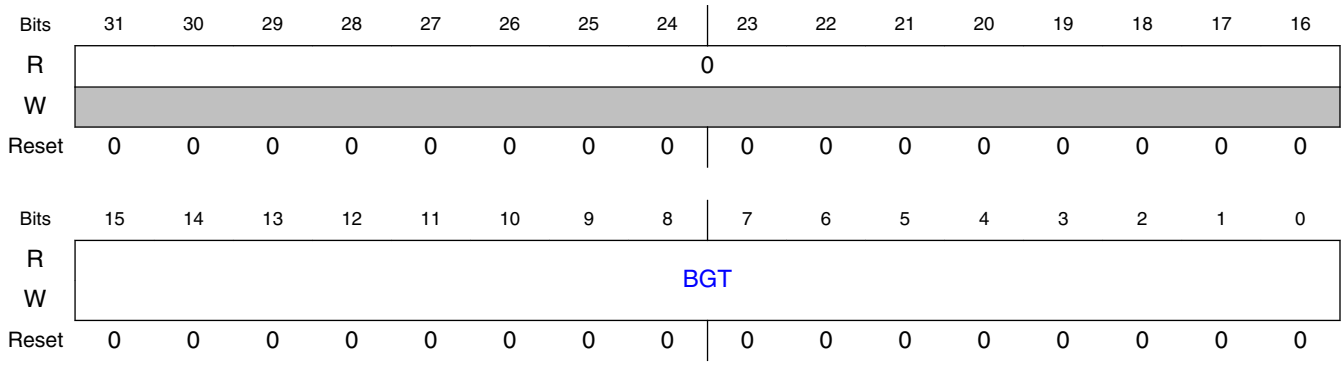
| Field | Function |
|---|---|
| 31-0<br>BWT | Block Wait Time Value<br>The time from START bit of last byte sent from the EMV SIM module to the START bit of the first byte sent from the Smart Card must be less than the value in this register. If it is not, then the BWT_TO flag is set. |

## 60.6.1.18 Block Guard Time Value Register (BGT_VAL)

### 60.6.1.18.1 Offset

| Register | Offset |
|---|---|
| BGT_VAL | 40h |

### 60.6.1.18.2 Diagram




### 60.6.1.18.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>BGT | Block Guard Time Value<br>The value in this register is the block guard time. Time from START bit of last byte sent from the EMV SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set. |

## 60.6.1.19 General Purpose Counter 0 Timeout Value Register (GPCNT0_VAL)

### 60.6.1.19.1 Offset

| Register | Offset |
|---|---|
| GPCNT0_VAL | 44h |

### 60.6.1.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GPCNT0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 60.6.1.19.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>GPCNT0 | General Purpose Counter 0 Timeout Value<br>The value written to this register will be used to compare to the general purpose counter 0 in the EMV SIM module. Once the General purpose counter reaches this value, the GPCNT0 flag in the TX_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. |

## 60.6.1.20 General Purpose Counter 1 Timeout Value (GPCNT1_VAL)

### 60.6.1.20.1 Offset

| Register | Offset |
|---|---|
| GPCNT1_VAL | 48h |

### 60.6.1.20.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GPCNT1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 60.6.1.20.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>GPCNT1 | General Purpose Counter 1 Timeout Value<br>The value written to this register will be used to compare to the general purpose counter in the EMV SIM module. Once the General purpose counter reaches this value, the GPCNT1 flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. |

# 60.7 Functional Description

## 60.7.1 Initialization

### 60.7.1.1 Configuring EMV SIM

The first operation that must be done is to configure the EMV SIM interface signals to allow the EMV SIM Transmit or Receive operation to commence. The following steps describe a typical way to enable the Smart Card:

- Configure prescaler by programming the desired value in PRSC register to generate the card clock. The card clock frequency is calculated as:
  - $f_{SIM}$ / (PRSC[7:0]); where $f_{SIM}$ is the EMV SIM Clock Frequency
  - Prescaler value must be set such that the maximum card clock frequency specification set by EMV & ISO-7816 specifications are not violated

- Configure the DIVISOR_VALUE in the Baud Rate Divisor Register. This value determines the ETU period to be used by the transmitter or receiver
- Enable power to the Smart Card by setting the VCC_EN bit in PCSR register
- Enable the Smart Card clock by setting the SCEN bit in PCSR register
- Release Smart Card reset using the SRST bit in PCSR register to bring the Smart Card out of reset
- Program the power down functionality i.e. if Auto Power Down of Smart Card is required or not
- Select the Data Format type for the EMV SIM
  - Either set the IC bit to the desired data format (inverse or direct) OR
  - Set the ICM bit to allow automatic detection of Initial Character. IC bit will be updated to desired data format upon valid Initial Character detection
- Configure if the EMV SIM module is enabled to run in STOP and DOZE modes by setting the STOP_EN and DOZE_EN bits as desired
- Enable DMA request generation for Transmitter or Receiver as desired

Once the above configuration is complete, the receiver or transmitter can be configured depending on the operation to be carried out on the Smart Card.

**NOTE**

XMT_EN and RCV_EN bits should not be changed in the same write cycle. Software should attempt separate write accesses to CTRL register for updating these bits, one by one. For example, if XMT_EN = 1 and RCV_EN = 0 and in order to enable the receiver, first write to CTRL register should make XMT_EN = 0 and second write to CTRL register should make RCV_EN = 1.

### 60.7.1.2 Configuring Receiver

The following is a list of configurations that need to be performed (after the "Configuring EMV SIM" step) in order to configure the EMV SIM receiver for operation:

- Enable NACK generation capability
  - To generate NACK on parity errors and invalid initial characters, set the ANACK bit in CTRL register
  - To generate NACK on receiver FIFO overflow error, set the ONACK bit in CTRL register
- Configure the desired threshold for NACK generation and data threshold in the RX_THD register. These are the thresholds at which the RTE and RDTF flags will be set, respectively.
  - If auto power down is desired on RTE flag assertion, then write to SAPD bit PCSR to initiate the Smart Card power down

- Configure the timeout values for Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter
- Enable necessary interrupts by clearing respective bits in the INT_MASK register
- Configure the General Purpose Counter as needed and select desired clock source (GPCNT0/1_CLKSEL in the CLKCFG register)
- Enable Receiver by setting RCV_EN bit to 1 in CTRL register
- Receiver will enter Initial Character Detection mode if ICM bit is set or enter normal receive mode if ICM bit is not set
- Enable LRC or CRC as desired

#### 60.7.1.3 Configuring Transmitter

The following is a list of configurations that need to be performed (after the "Configuring EMV SIM" step) in order to configure the EMV SIM transmitter for operation:

- Select desired re-transmission threshold for NACKed characters in TX_THD register
  - This is the threshold at which the TNTE flag will be set by using the TNACK_THD[3:0] bits
- Configure the desired Guard Time between the characters transmitted by writing to the GETU[7:0] bits in TX_GETU register
- Configure the desired data threshold in TX_THD register.
- Enable necessary interrupts by clearing respective bits in the INT_MASK register
- Configure the General Purpose Counter as needed and select desired clock source (GPCNT0/1_CLKSEL in the CLKCFG register)
- Enable Transmitter by setting the XMT_EN in the CTRL register
- Enable LRC & CRC as desired
- Program bytes to be transmitted into the Tx FIFO. If DMA is enabled, DMA request will get asserted automatically.

**NOTE**

Tx FIFO can be written to after the transmitter is enabled (XMT_EN = 1). Writing to Tx FIFO while transmitter is disabled will not be successful and write access will be ignored.

**NOTE**

The Transmit and Receive operations are mutually exclusive and should not be enabled together.

### 60.7.2 Smart Card Interface and Control

The EMV SIM module controls the all the interface signals for the Smart Card. The Smart Card clock is generated using the prescaler (PRSC register) and the ETU periods are generated using the baud rate divisor (DIVISOR register). The Smart Card Reset and Voltage Enable are controlled by software through the PCSR register. The message bytes are transmitted and received on the Card IO device pin which is used in open drain configuration.

### 60.7.2.1 Smart Card Presence Detect

The Smart Card Presence Detect pin (SIMPD) allows for detection of the insertion or removal of a Smart Card. The SPDES bit in the PCSR register allows the software to configure which edge of the SIMPD pin will cause a presence detect interrupt flag to be asserted.

An internal pull-down is required on SIMPD pin to allow a high to low transition on this pin when the Smart Card is removed.

### 60.7.2.2 Powering Up

The first step to communicating with a Smart Card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the Smart Card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the Smart Card by setting the VCC_EN bit in the PCSR register.
2. Select the appropriate clock frequency for the Smart Card by programming the CLKCFG and DIVISOR registers.
3. Enable the clock to the Smart Card by setting the SCEN bit in the PCSR register.
4. Remove the card from reset by setting the SRST bit in the PCSR register.

### 60.7.2.3 Automatic Power Down

Smart Cards require a proper sequence of operations to be followed to allow them to be powered down. The power down sequence as per ISO 7816 specifications is:

- Smart Card Reset (RST) pin transitions from high to low
- Smart Card Clock (CLK) is turned off to low
- Smart Card IO (IO) is transitioned from high impedance to low
- Smart Card Voltage Enable (VCC_EN) to turned off (pin driven low)

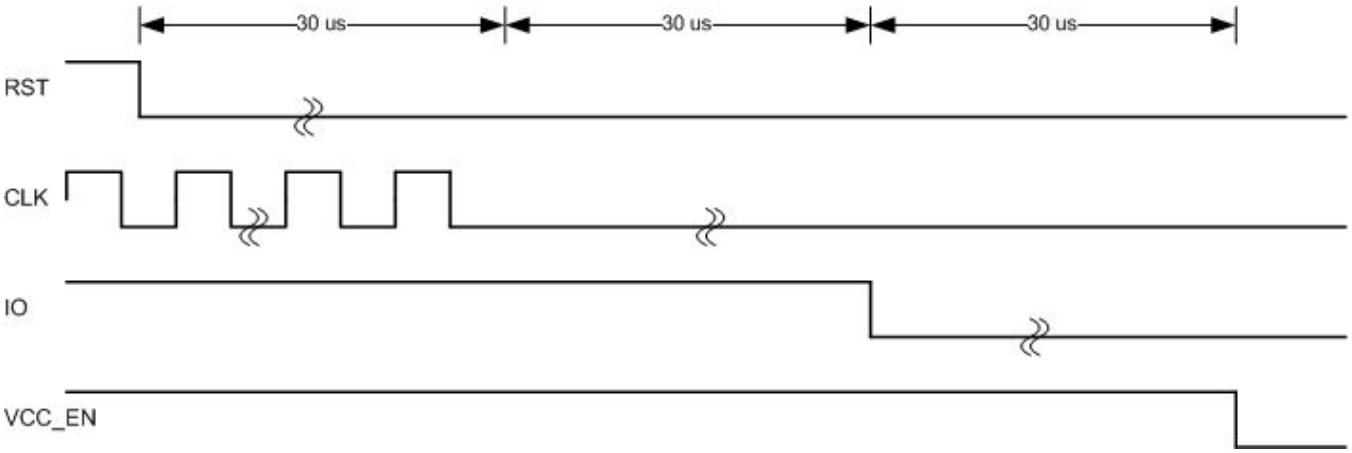



**Figure 60-2. Auto Power Down Sequence**

These operations can be done manually by software by writing to appropriate bits (SCEN, SRST and VCC_EN) in the PCSR register or automatically by hardware under the following conditions provided the auto power down feature is enabled in the PCSR register.

- Setting the SPD bit in PSCR register by software
- Assertion of RTE bit in RX_STATUS register
- Assertion of SPDIF bit in PCSR register when configured to detect card removal

**NOTE**

The SPDES bit in PCSR register determines which edge transition of the Smart Card Presence Detect pin is used for Smart Card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the edge specified by SPDES bit will cause the following: SPDIF to be set; if the SPDIM mask is clear, an interrupt to the CPU; and if SAPD in the PCSR register is set, an auto power down sequence to begin. If Smart card insertion is expected, SAPD can be set low to avoid the auto power down sequence from falsely triggering. The bit SPDP can be used to determine the current state of the Smart Card Presence Detect pin.

**NOTE**

The auto power down operation requires a low frequency clock (usually 32 kHz clock source) to be active on the device.

The auto power down sequence will clear the following register bits upon power down sequence completion:

- SPD bit in PCSR register
- SCEN bit in PCSR register

- SRST bit in PCSR register
- VCC_EN bit in PCSR register
- XMT_EN and RCV_EN bits in CTRL register

## 60.7.3 EMV SIM Receiver

The EMV SIM Receiver is designed to receive all message bytes and store valid bytes into the receive FIFO. It performs checks on the incoming messages and indicates the transmitter to enable NACK insertion for each message. In addition, the receiver automatically detects NACKs received when the transmitter is sending out message bytes.

### 60.7.3.1 Data Sampling

The incoming data is internally synchronized and sampled using a 16x oversampled clock, called the EMV SIM logic clock. The frequency of this clock is 16 times the ETU Clock (1/ETU Period). The whole receiver works on this EMV SIM logic clock.

### 60.7.3.2 Initial Character Detection

The EMV SIM module supports the detection of the Initial Character and determine the data format when it is placed in the initial character mode by setting the ICM bit in CTRL register. When placed in this mode, the EMV SIM module samples in the input line and waits for one of the initial characters to be detected. Once detected, the ICM bit is cleared and the data format bit, IC bit, is set to appropriate value depending on the data format detected using the initial character.

The two possible data formats are inverse convention and direct convention. Figures below illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the bits is flipped (i.e. MSB for LSB) and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

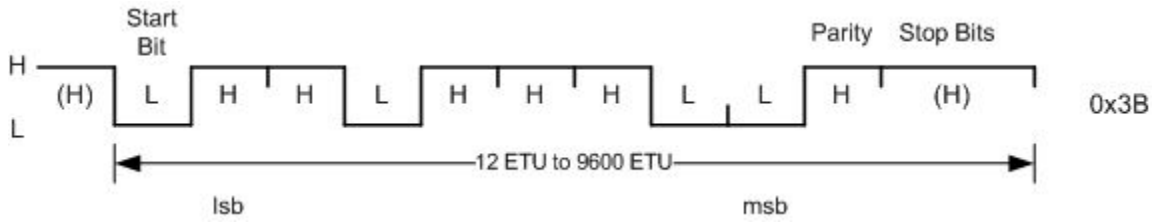



**Figure 60-3. Initial Character in Direct Conversion**

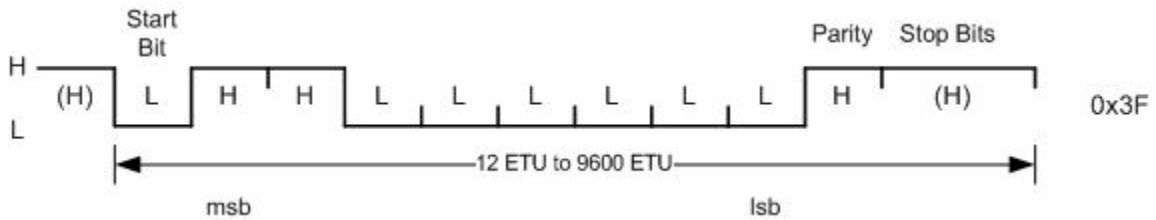



**Figure 60-4. Initial Character in Inverse Conversion**

To place the EMV SIM into initial character mode, set the ICM bit in the CTRL register. Once a valid initial character is received, the IC bit in the CTRL register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the EMV SIM is currently using. The 0x3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a 0x3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

The receiver remains in initial character mode until disabled by software or a valid initial character is detected. Upon detection of valid character, the receiver enters data receive mode. It is recommended to use one of the general purpose counter as a timeout during initial character mode.

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as 0x3F).

**NOTE**

There is a condition where a parity error in the initial byte for direct convention (0x3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, 0x3F). The EMV SIM module will not recognize this as a valid

initial character for inverse convention and will set the PEF flag.

### 60.7.3.3 Receiver Diagnostics

The EMV SIM module performs diagnostic checks on the incoming messages. These checks include parity check and message frame checks.

***Parity Error Detection:*** The receiver is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct.

When a parity error is detected on a given byte, the parity error flag (PEF bit) is asserted in the RX_STATUS register if NACK generation on errors is disabled (ANACK = 0 in CTRL register). The parity error flag does not generate an interrupt to the system, however, the module can cause the transmitter to transmit a NACK (if ANACK = 1 in CTRL register) without software needed to enable transmitter for this and request re-transmission of the current byte.

***Framing Error Detection:*** The receiver is responsible for detecting framing errors in the received data. Data is always transmitted with 2 STOP bits, except when in 11 ETU mode, where there is one STOP bit. The STOP bits are defined as high state at the 11th and 12th bit positions in the received data. In either mode, the receiver expects to see a high at the 11th bit position in the received data. If a low is detected at the 11th bit position, the receiver indicates a framing error for that byte.

When a framing error is detected on a given byte, the FEF bit is set in the RX_STATUS register. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the Smart Card asking for a retransmission of the corrupted data.

***LRC & CRC:*** The receiver also checks the LRC or CRC on the incoming block of bytes (if LRC or CRC is enabled). The result of the CRC or LRC check is updated in the RX_STATUS register once the complete message block is received.

### 60.7.3.4 NACK Detection

During transmission of message bytes, the receiver automatically checks for NACKs sent by the Smart Card. The software does not need to enable receiver for this. The NACK is detected by sampling the IO line at 11 ETUs after the falling edge of the START bit. Once a NACK is detected, the transmitter is signaled and the transmitter will retransmit the current byte after two ETU time (as per ISO 7816 specification).

### 60.7.3.5 Using EMV SIM Receiver with "T=1" Smart Cards

The EMV SIM module provides hardware support for “T=1” type Smart Cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the EMV SIM receiver are as follows:

- 11 ETU Characters
  - “T=1” cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The EMV SIM module provides the RCVR11 bit in the CTRL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the Smart Card. The value of CWT can range from 12 ETU to 32779 ETU. The EMV SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the Smart Card.
- Block Wait Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The block wait time must not exceed a value that is programmable. The EMV SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The block guard must be greater than a value that is programmable. The EMV SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The EMV SIM module provides hardware support for both the LRC and CRC operations.

## 60.7.4 EMV SIM Transmitter

The transmitter is responsible for reading the message bytes from the Tx FIFO, adding the LRC or CRC at the end of the message block and re-transmit any byte on reception of a NACK. It assists the receive operation by inserting NACKs when directed by the receiver.

### 60.7.4.1 Message Transmission

The transmitter operation does not start until the Tx FIFO is empty. Software can configure the EMV SIM module and enable the transmitter but unless a byte is written into the Tx FIFO, no transmission will start. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO. Refer Message Handling section for more details on using the Tx FIFO.

The transmitter continuously transmits the messages from the Tx FIFO. Upon NACK detection, the transmitter resends the same byte until the Transmitter NACK Threshold limit is reached. When TNTE is set, the EMV SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO are lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until TNTE is cleared by software. The only way to clear TNTE is to write a 1 to the TNTE bit in the TX_STATUS register. It is possible to disable the detection of NACKs from the Smart Card by setting TNCK_THD[3:0] to 0x0. By setting TNCK_THD[3:0] to 0x1, it is possible to disable all retransmissions while still setting TNTE on the first NACK received. In general, TNTE is set on the NACK that causes the threshold set by TNCK_THD[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

If all bytes are successfully transmitted and the Tx FIFO is empty, the transmitter send the check byte (i.e. LRC or CRC) if the XMT_CRC_LRC and LRC_EN or CRC_EN bits are set. The transmitter sends each byte to the LRC or CRC block to allow the check byte generation.

At the end of each byte, the transmitter inserts the programmed Guard ETUs. Refer to Protocol Timers section for more details on Guard Time Counter.

### 60.7.4.2 NACK Insertion

The transmitter is responsible for driving the output low during the STOP bit time to signify an error was detected in the received data from the Smart Card. This logic responds to a NACK request generated by the receiver block.

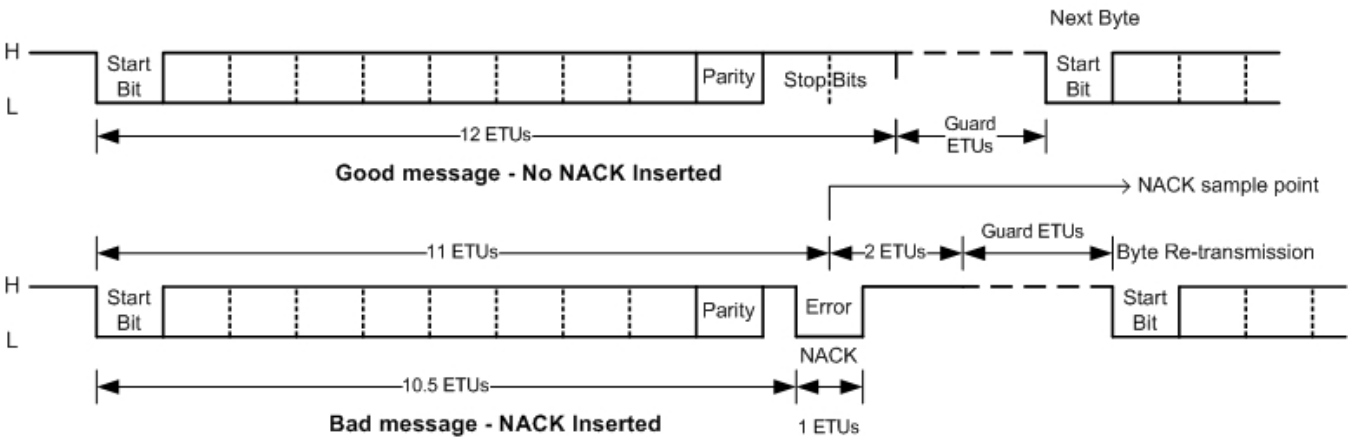



**Figure 60-5. NACK Insertion**

The NACK generation logic is also responsible for keeping track of the number of NACKs received during a transmit operation. The receiver detects NACKs generated by the Smart Card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (TNCK_THD[3:0]), an interrupt flag (TNTE) is generated, the transmit FIFO is flushed, and the transmitter is disabled.

### 60.7.4.3 Using EMV SIM Transmitter with "T=1" Smart Cards

The EMV SIM module provides hardware support for “T=1” type Smart Cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the EMV SIM transmitter are as follows:

- 11 ETU Characters
  - The EMV SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (0xFF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the

programmable guard time in the GETU register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The EMV SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.

- Block Waiting Time
  - The block waiting time (BWT) is defined as the maximum time between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The EMV SIM transmitter provides two General Purpose Counter(s) either of which can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the start bits of the last character of a transmitted block and the first character of the next received block. The value of BGT is 22 ETU. The EMV SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The EMV SIM module provides hardware support for both the LRC and CRC operation.

## 60.7.5 LRC and CRC

### 60.7.5.1 LRC Block

The EMV SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 Smart Cards that support LRC. This block can be enabled through the LRC_EN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRC_OK bit is set in the RX_STATUS register. During transmission, the LRC block Exclusive-ORs

each character that is transmitted with the current value of the LRC. If the XMT_CRC_LRC bit in the CTRL register is set, the LRC value will automatically be sent by the transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCR_EN bit in the CTRL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the LRC value is automatically reset by the EMV SIM hardware. Finally, when setting the XMT_EN bit, the EMV SIM hardware resets the LRC value.

Summary of configuration needed to use LRC block:

- Use the LRC_EN bit in CTRL register to enable the LRC block
- Use the XMT_CRC_LRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent.

### 60.7.5.2 CRC Block

The EMV SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 Smart Cards that support CRC. This block can be enabled through the CRC_EN bit in the CTRL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial used for calculating the CRC is $G(x) = x^{16} + x^{12} + x^5 + 1$. The CRC register is initialized to all "1" before the data is shifted in. The CRC block can be configured to compute using the ISO 7816 CRC method or the 16-bit CCITT CRC method. The software can select the following bits: CRC_IN_FLIP, CRC_OUT_FLIP, and INV_CRC_VAL of the CTRL register.

- To select CCITT CRC method, set
  - CRC_IN_FLIP = 1
  - CRC_OUT_FLIP = 1
  - INV_CRC_VAL = 1
- To select normal CRC method (i.e. ISO 7816 CRC method), set
  - CRC_IN_VAL = 0
  - CRC_OUT_VAL = 0
  - INV_CRC_VAL = 1

During transmission, the CRC block updates the current value of the CRC using each character. If the XMT_CRC_LRC bit in the CTRL register is set, the CRC value will automatically be formatted as per the selected CRC method (CCITT or normal) and sent by the EMV SIM transmitter as the final two characters when the transmit FIFO empties. At the end of the reception of a block of characters, the residual from the CRC calculation is compared and the CRC_OK bit is set in the RX_STATUS register.

**NOTE**

Setting any other value for CRC_IN_FLIP, CRC_OUT_FLIP and INV_CRC_VAL will cause CRC value to be formatted accordingly; however the CRC will not be compliant to the standard CRC used in Smart Cards.

The CRC value can be reset/initialized in multiple ways. Clearing the CRC_EN bit in the CTRL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the CRC value is automatically reset by the EMV SIM hardware. Finally, when setting the XMT_EN bit, the EMV SIM hardware resets the CRC value.

Summary of configuration needed to use CRC block:

- Use CRC_IN_FLIP, CRC_OUT_FLIP and INV_CRC_VAL bits in CTRL register to select the desired CRC method (CCITT or normal)
- Use the CRC_EN bit in CTRL register to enable the CRC block
- Use the XMT_CRC_LRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent.

## 60.7.6 Message Handling

The EMV SIM module has FIFOs on both transmit and receive side for handling all message bytes.

### 60.7.6.1 Transmit FIFO

A 16-byte deep FIFO is implemented in the transmitter. The transmit FIFO can be written into by the software but reads to this FIFO will return zeros. To take care of clock synchronization, each write access will have wait states inserted by the module but this is transparent to the software. The Tx FIFO can be written to while the transmitter is enabled, however, the transmitter does not start transmitting while the FIFO is empty. A write to the Tx FIFO will initiate the transmit operation. Software can store more bytes into the Tx FIFO until the programmed threshold is reached. Software can enable DMA operation by setting TX_DMA_EN = 1 in the CTRL register. On setting this bit, a DMA request will be asserted and remain asserted till the number of bytes in FIFO reached the programmed threshold value (TDT[3:0] in the TX_THD register).

The transmit FIFO can be flushed by setting the FLSH_TX bit in the CTRL register. A transmit NACK threshold error (TNTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the FLSH_TX operation.

### 60.7.6.2 Receive FIFO

A 16-byte deep FIFO is implemented in the receiver. Since more than 16-bytes can be received in the FIFO, the software must ensure that the Rx FIFO is periodically read. The Rx FIFO can be read by software via normal CPU accesses on interrupt (RX_DATA interrupt flag) or via DMA by setting the DMA_RX_EN bit in CTRL register. The software cannot write to this register. A write access may terminate in a transfer error.

The Rx FIFO is loaded by the receiver when it receives an error free message byte with correct parity. The FIFO stores the message byte only. When the total bytes in the Rx FIFO equals the programmed threshold value (RDT[3:0] in RX_THD register), the RDTF bit is asserted in the RX_STATUS register. An interrupt will be asserted if the RDT_IM bit in the INT_MASK is cleared. If DMA access to RX_FIFO is enabled, the DMA request to read the Rx FIFO will be asserted when the threshold flag is set and will clear when all the bytes are read out from the Rx FIFO.

The receive FIFO can be flushed by setting the FLSH_RX bit in the CTRL register. The flush operation resets the receiver except for receiver control registers.

**NOTE**

Since the register bits and the receiver logic are in different clock domains, the software needs to make sure to allow clock synchronization time when sampling the RDTF bit. One proposed method could be to read one byte at a time (from Rx FIFO) when RDTF is asserted, allow a time of 3 bus clock cycles after Rx FIFO read and then check the status of the RDTF bit and repeat as necessary. Alternatively, one could set the receiver threshold one more than actually needed.

***Rx FIFO Overflow Detection:*** When a byte is received by the receiver and the Rx FIFO is not read and already contains 16 bytes, a FIFO overflow error will be asserted (RFO bit set to 1 in RX_STATUS register). The received byte will be discarded leaving the FIFO with the first 16 bytes received. If the ONACK bit in the CTRL register is asserted, the

transmitter will generate a NACK for the overflow condition every time a new byte is received that cannot be stored in the Rx FIFO. The EMV SIM module will continually send NACKs till the Rx FIFO is read or NACK transmit threshold is reached.

## 60.7.7 Protocol Timers

The EMV SIM module has several 16-bit and 32-bit timers to allow the software perform protocol timing checks and measurements like:

- Guard Time
- Character Wait Time
- Block Wait Time
- Block Guard Time
- Work Wait Time

In addition to the above, the EMV SIM module also contains two 32-bit general purpose counters which can be used by software for any application specific time measurement.

### 60.7.7.1 Transmit Guard Time

The Transmit Guard time is the number of ETUs inserted by the transmitter between each character being transmitted. The number of ETUs is controlled by programming the TX_GETU register. The Guard Time is measured as the time (in ETUs) between the STOP bits of previous character and the START bit of next character.

For a Guard Time programmed as 0xFF in GETU[7:0] bits of TX_GETU register, the transmitter inserts one STOP bit instead of two.

### 60.7.7.2 Character Wait Time

The EMV SIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters (i.e. time between start bits in the consecutive characters received). When enabled, the Character Wait Time Counter (CWT) will not start counting until after the START bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETUs between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear. The CWT can be configured as follows:

- Program the CWT_VAL register to the value that needs to be enforced
- Activate CWT function by setting the CWT_EN bit in the CTRL register
- Enable the CWT_ERR interrupt by clearing the CWT_IM bit in the INT_MASK register
- If an interrupt occurs, then the software should read the RX_STATUS register to determine if the error was caused by a CWT_ERR violation. The software should clear the CWT_ERR interrupt by writing a “1” to the CWT_ERR bit in the RX_STATUS register
- Software can disable the CWT counter by clearing the CWT_EN bit in the CTRL register at any time

#### 60.7.7.3 Block Wait Time and Block Guard Time

A 32-bit timer is used to measure both Block Wait Time (BWT) and Block Guard Time (BGT). The BWT and BGT timer are used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If this time measured is larger than the value in BWT_VAL register, then the BWT_ERR flag will be set and an interrupt generated. If the time is less than the value in the BGT_VAL register, then the BGT_ERR flag will be set and an interrupt generated. The block wait timer can be configured as follows:

- Program the BWT_VAL and BGT_VAL registers to the value that need to be enforced
- Activate the block wait timer by writing '1' to BWT_EN bit in the CTRL register
- Enable the BWT_ERR and BGT_ERR interrupts by clearing the BWT_IM and BGT_IM bits in the INT_MASK register
- If an interrupt occurs, then the software should read the RX_STATUS register to determine if the error was caused by a BWT_ERR or BGT_ERR violation. The software should clear the interrupt status bit by writing a “1” to the BWT_ERR bit or BGT_ERR, respectively, in the RX_STATUS register
- Software can disable the BWT counter by clearing the BWT_EN bit in the CTRL register at any time

**NOTE**

BWT should be enabled after all bytes are written to in Tx FIFO. This is necessary to avoid the condition where writes to Tx FIFO are very slow (more than 1 ETU time apart) and if BWT is enabled earlier, then BWT error can trigger falsely.

### 60.7.7.4 Work Wait Time

There is no separate counter to measure the work wait time (WWT). Work wait time is a combination of BWT and CWT. To measure WWT, software must configure both CWT_VAL and BWT_VAL to the value to enforced. When measuring WWT, the BGT timer will not be used so it should be masked (inactive) by the BGT_IM bit. Below is the sequence to program the EMV SIM to send and receive data while checking WWT:

- Program both the CWT_VAL and the BWT_VAL registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
- Activate both the CWT and BWT functions by setting the CWT_EN and BWT_EN bit in the CTRL register
- Enable the CWT_ERR and BWT_ERR interrupts by clearing the CWT_IM and BWT_IM bits in the INT_MASK register
- Program the EMV SIM module to enable Receiver or Transmitter as desired by application
- If an interrupt occurs, then the software should consider that a WWT violation if the CWT_ERR or BWT_ERR bits are asserted. The software should clear the CWT_ERR or BWT_ERR interrupt by writing a “1” to the BWT_ERR or CWT_ERR bit in the RX_STATUS register

### 60.7.7.5 General Purpose Timers

The EMV SIM module provides TWO 16-bit counters for use, when timing events during Smart Card communication. The clock source for the counter is selectable between three sources: Card Clock, Receiver Clock (16 times the ETU clock) or ETU Clock (for Transmitter). The GPCNT0/1_CLKSEL[1:0] bits in the CTRL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the SCEN, KILL_CLOCKS, RCV_EN and XMT_EN bits provided in the CTRL register.

To run the counters from the card clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register
- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- 'SCEN = 1' in the PCSR register
- ‘GPCNT_CLKSET[1:0] = 01’ in the CLKCFG register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counters from the receive clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register.

- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- ‘GPCNT0/1_CLKSET[1:0] = 10’ in the CLKCFG register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counters from the ETU (transmit) clock source the following conditions must be met:

- ‘KILL_CLOCKS = 0’ in the CTRL register.
- ‘RCV_EN = 1’ or XMT_EN = 1 in the CTRL register
- ‘GPCNT0/1_CLKSET[1:0] = 11’ in the CLKCFG register

The counters will begin to count at the ETU clock rate as soon as these conditions are met.

The counters can be reset by setting GPCNT0/1_CLKSEL[1:0] to 00. A 16-bit comparator value (GPCNT0/1_VAL register) is provided that allows the software to select a count value at which the interrupt flags (GPCNT0/1_TO bit in TX_STATUS register) can be set and an interrupt generated if the mask (GPCNT0/1_IM bits in INT_MASK register) are clear.

The following sequence should be followed to configure both general purpose timers:

- Program counter comparator using the GPCNT0/1_VAL register
- Enable the conditions described above using the CTRL register
- Select desired clock source for the General Purpose Counter using the CLKCFG register
- Enable interrupts in the INT_MASK register

## 60.7.8 Answer To Reset (ATR) Detection

The first communication that occurs between the Smart Card and the EMV SIM module will be a block of data sent from the Smart Card to the EMV SIM module after the card is powered up and the card reset is removed (As described in Smart Card Interface and Control Section). This block is called the Answer To Reset (ATR). To receive the ATR, the EMV SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 spec, both “T=0” and “T=1” cards will communicate initially using 12 ETU character durations.

The following steps provide a suggested approach to configure the EMV SIM module to receive the ATR.

- Clear RCVR11 bit in the CTRL register

- Set ANACK bit in the CTRL register to enable NACK generation. The ISO 7816-3 spec allows the EMV SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

**NOTE**

The Europay MasterCard and VISA (EMV) cards are similar to “T=1”, but do not allow the EMV SIM module to NACK during the initial communication

- Enable RDTF and RFO interrupts by setting RDT_IM and RFO_IM bits in INT_MASK register. These interrupts are enabled to notify the software when characters are received. A threshold can be set for the number of characters to receive before generating an interrupt.
- Set desired threshold for received characters by writing RDT in RX_THD register

The EMV SIM should be setup to perform in initial character mode during ATR detection. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the Smart Card.

- Set Initial Character Mode by setting the ICM bit in the CTRL register

The ISO 7816-3 spec requires that Smart Cards meet certain timing restrictions. One of these is the time from the de-assertion of the card reset to the beginning of the ATR sequence. The EMV SIM module's General Purpose Counter can be used to verify that the Smart Card begins its ATR within the 400 to 40,000 card clock cycle range.

- Set the General Purpose Counter 0 or 1 Timeout Value to 0x9C40 using GPCNT0/1_VAL register.
- Enable General Purpose Counter Interrupt by clearing GPCNT0/1_IM in INT_MASK register.
- Enable General Purpose Counter by programming the GPCNT0/1_CLKSEL[1:0] bits to 01, so that the card clock is used for counting.

The ISO7816-3 spec states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

- Set Character Wait Time Counter Comparator to 9600 using the CWT_VAL register
- Enable the Character Wait Time Counter Interrupt by clearing CWT_ERR_IM in INT_MASK register
- Enable the Character Wait Time Counter by setting the CWT_EN bit in the CTRL register

The last step in preparing for ATR reception is to enable the receiver.

- Set RCV_EN bit in CTRL register

The EMV SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (RX_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 spec (See Section Smart Card Interface and Control). Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 spec for details).

#### 60.7.8.1 Programming Considerations for T=0 Smart Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values of DIVISOR register based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of GETU[7:0] in the TX_GETU register.
- Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CTRL register.
- Adjust the stop clock polarity by modifying the value of the SCSP bit in the PCSR register.
- Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the TNCK_THD[3:0] bits in the TX_THD register.
- Adjust the level for the Receive NACK threshold by modifying the RNCK_THD[3:0] bits in the RX_THD register.

If a negotiation with the Smart Card is desired, the software sends a PPS response to the Smart Card. To send the response, the following steps should be performed:

- Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the TX_THD register.
- Clear all transmit interrupt flags in the TX_STATUS register by writing a one to the bits of the register.
- Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the Smart Card.
- Enable the transmitter by setting the XMT_EN bit in the CTRL register.
- Write the characters to be sent as response (max 16) to the transmit FIFO using the TX_BUF register.

At this point, the EMV SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the Smart Card.

Once the transmission is complete, the EMV SIM module should be completely configured for standard operation with the T=0 Smart Card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

### 60.7.8.2 Programming Considerations for T=1 Smart Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

- Adjust the baud rate by changing the values of DIVISOR register based on the values of 'F' and 'D' received in the ATR.
- Adjust the guard time between characters by changing the value of GETU[7:0] in the TX_GETU register. Setting GETU[7:0] to 0xFF configures the SIM transmitter for 11 ETU transmissions.
- Disable NACK capability by clearing the ONACK and ANACK bits in the CTRL register. T=1 cards do not allow NACKs.
- Adjust the stop clock polarity by modifying the value of the SCSP bit in the PCSR register.
- Set Character Wait Time Counter Timeout Value to value specified in the ATR by using the CWT_VAL register.
- Enable the Character Wait Time Counter Interrupt by clearing CWT_ERR_IM in INT_MASK register.
- Enable the Character Wait Time Counter by setting the CWT_EN bit in the CTRL register.
- Enable CRC or LRC error checking according to the ATR information by setting either the CRC_EN or LRC_EN bit in the CTRL register. These bits should never be set at the same time!

For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the Smart Card is desired, the software will send a PPS response to the Smart Card. Otherwise, the protocol is initiated with a block transfer from the EMV SIM module. In order to send the response or the first block, the following steps should be performed:

- Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the TX_THD register.

- Clear all transmit interrupt flags in the TX_STATUS register by writing a one to the bits of the register.
- Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 characters are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the Smart Card.
- Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_CRC_LRC bit in the CTRL register.

**NOTE**

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_CRC_LRC bit during the PPS exchange.

- Enable the transmitter by setting the XMT_EN bit in the CTRL register.
- Write the characters to be sent as response (max 16) to the transmit FIFO using the TX_BUF register.

At this point, the EMV SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the Smart Card.

Once the transmission is complete, the EMV SIM module should be completely configured for standard operation with the T=1 Smart Card. The software can continue to service RDTF interrupts for received characters, and TDTF interrupts for transmitted characters.

# Chapter 61
# Universal Serial Bus Full Speed Controller (USBFS)

## 61.1 Chip-specific USB information

In this chapter, the term IRC48M refers to the Fast Internal Reference Clock (FIRC) discussed in the other chapters.

This MCU only supports Full Speed Device Mode.

### 61.1.1 USB wakeup

When the USB detects that there is no activity on the USB bus for more than 3 ms, the INT_STAT[SLEEP] bit is set. This bit can cause an interrupt and software decides the appropriate action.

Waking from a low power mode (except in LLS/VLLS mode where USB is not powered) occurs through an asynchronous interrupt triggered by activity on the USB bus. Setting the USBTRC0[USBRESMEN] bit enables this function.

The following wakeup functionality is also supported:

- In Stop/VLPS, the USB controller can generate an interrupt on VREGIN detection.
- In LLS/VLLS except VLLS1/0, VREGIN is an input pin to the LLWU and a transition on it can generate a wakeup.
- In LLS/VLLS, USB0_DP and USB0_DM are input pins to the LLWU and a transition on those pins can generate a wakeup provided the USB is powered and in host mode.

### 61.1.2 LPUART over USB capability

This device supports a connection whereby LPUART2 may be connected to the FS USB physical layer (and the USB controller disconnected) to allow simple debugging capabilities for applications which do not have direct physical access to LPUART2.

By setting the USBx_USBCTRL[UARTSEL] bit, LPUART2 will be connected to the FS USB physical layer. This configures the FS USB DP/DM signals to be configured as normal LPUART2 signals, which do not operate in differential mode.

## 61.1.3 USB Power Distribution

This chip includes an internal 5 V to 3.3 V USB regulator that powers the USB transceiver or the MCU (depending on the application).

### 61.1.3.1 AA/AAA cells power supply

The chip can be powered by two AA/AAA cells. In this case, the MCU is powered through VDD which is within the 1.8 to 3.0 V range. After USB cable insertion is detected, the USB regulator is enabled to power the USB transceiver.

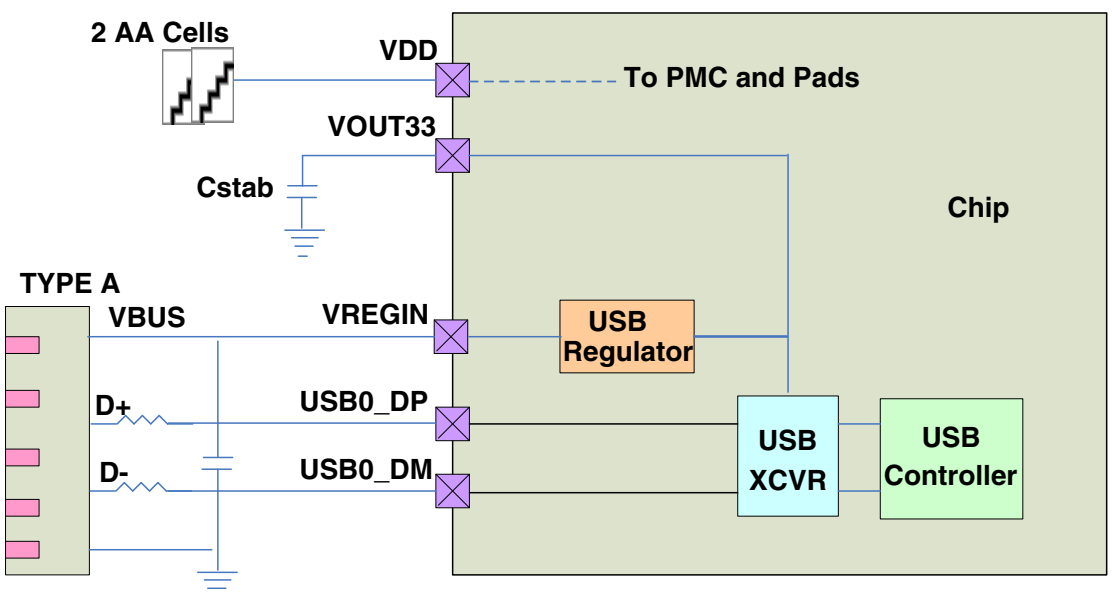



**Figure 61-1. USB regulator AA cell usecase**

### 61.1.3.2 Li-Ion battery power supply

The chip can also be powered by a single Li-ion battery. In this case, VOUT33 is connected to VDD. The USB regulator must be enabled by default to power the MCU. When connected to a USB host, the input source of this regulator is switched to the USB bus supply from the Li-ion battery.

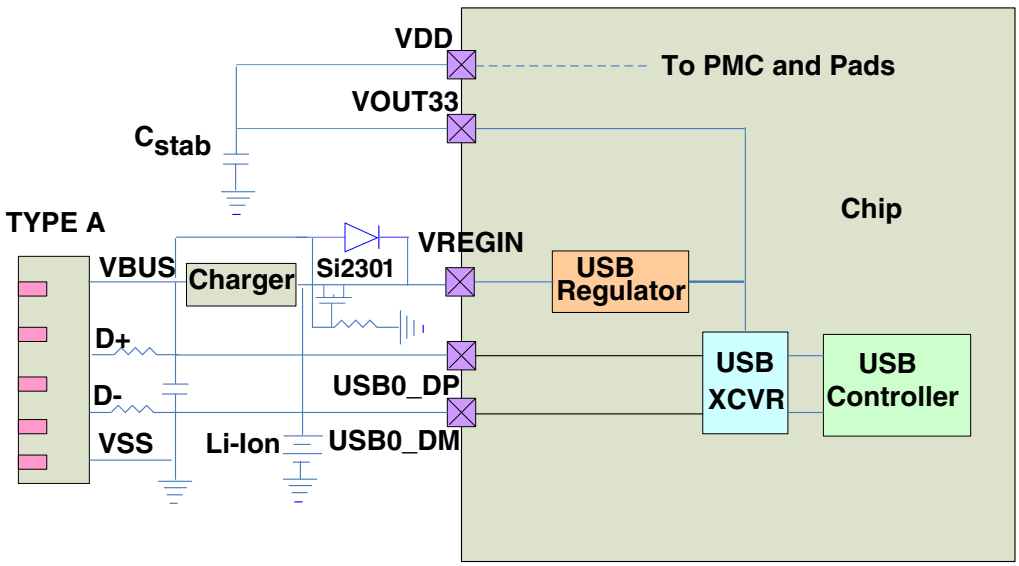



**Figure 61-2. USB regulator Li-ion usecase**

### 61.1.3.3 USB bus power supply

The chip can also be powered by the USB bus directly. In this case, VOUT33 is connected to VDD. The USB regulator must be enabled by default to power the MCU, then to power USB transceiver or external sensor.

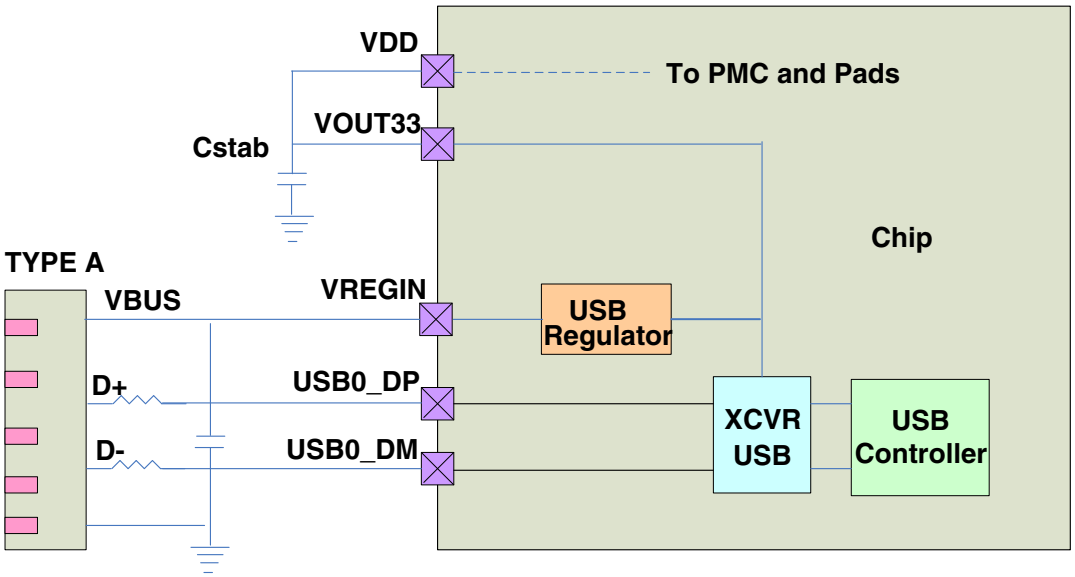



**Figure 61-3. USB regulator bus supply**

## 61.1.4 USB controller configuration

This section summarizes how the module has been configured in the chip.

**NOTE**

When USB is not used in the application, it is recommended to connect VOUT33 and VREGIN together and tie to ground through a 10 kΩ resistor. Do not tie directly to ground, because this causes a latch-up risk.

## 61.1.5 Keep Alive Mode and IRC48M operation

### 61.1.5.1 Entering Keep Alive mode

**To enter Keep Alive mode:**

1. Enter either VLPS mode or Stop mode, ensuring that the USB 48 MHz functional clock source remains enabled in either mode (VLPS mode or Stop mode):
   - **To enter VLPS mode:** set
     - KEEP_ALIVE_CTRL[KEEP_ALIVE_EN] = 1 (Keep Alive mode enable)
     - CLK_RECOVER_IRC_EN[IRC_EN] = 1 (IRC48M enable bit)
     - SPM_CORELPCNFG[CORELPCNFG_BGEN] = 1 (Bandgap Enable In VLPx Operation)

    - SPM_CORELPCNFG[SBDIS] = 1 (Source Bias Disable)
    - SPM_CORELPCNFG[WBDIS] = 1 (Well Bias Disable)
  - **To enter Stop mode:** set only
    - KEEP_ALIVE_CTRL[KEEP_ALIVE_EN] = 1 (Keep Alive mode enable)
    - CLOCK_RECOVER_IRC_EN[IRC_EN] = 1 (IRC48M enable bit)
2. To initialize the BDT in USB RAM, within each BDT set OWN=0 for every endpoint, except for EP0. The buffer address for EP0 must be configured to an address within the USB SRAM, ensuring that received data is placed within the USB SRAM. The USB cannot access other memory locations within USB Keep Alive mode.
3. Enable the wakeup interrupt, by setting KEEP_ALIVE_CTRL[WAKE_INT_EN] = 1.
4. Set KEEP_ALIVE_WKCTRL[WAKE_ON_THIS] appropriately to configure what received token will generate the wakeup from Keep Alive mode. Software then enters Stop mode and the USB is in Keep Alive mode.

### 61.1.5.2 Exiting Keep Alive mode

Any enabled interrupt generates a wakeup from STOP/VLPS mode; this causes an exit from USB Keep Alive mode. The USB can also be configured to generate an interrupt when it receives a particular token (for example, SETUP), as configured by KEEP_ALIVE[WAKE_ON_THIS]. Other USB interrupts can also remain enabled and can be generated in USB Keep Alive mode.

### 61.1.5.3 Device mode IRC48M operation

The following are the IRC48M initialization code steps:

1. Select the clock source and configure the clock divider for FIRC: SCG_FIRCDIV[FIRCDIV1] = 1, SCG_FIRCCFG[RANGE] = 0.
2. Enable FIRC: SCG_FIRCCSR [FIRCEN] = 1
3. Select FIRC as the system clock source: SCG_RCCR[SCS] = 0011

For chip-specific details, see the USBFS controller clocking information in the *Clock Distribution* chapter.

## 61.2 Introduction

This chapter describes the USB full speed controller. The USBFS controller provides functionality for implementing a USB2.0 full-speed compliant peripheral and interfaces to a USBFS transceiver.

**NOTE**

This chapter describes the following registers that have similar names: USB_CTL, USB_CTRL, and USB_CONTROL. These are all separate registers.

### 61.2.1 References

The following publications are referenced in this document. For copies or updates to these specifications, see the USB Implementers Forum, Inc. website at http://www.usb.org.

- *Universal Serial Bus Specification, Revision 2.0*, 2000, with amendments including those listed below
- *Errata for "USB Revision 2.0 April 27, 2000" as of 12/7/2000*
- *Errata for "USB Revision 2.0 April 27, 2000" as of May 28, 2002*
- *Pull-up / Pull-down Resistors* (USB Engineering Change Notice)
- *Suspend Current Limit Changes* (USB Engineering Change Notice)
- *Device Capacitance* (USB Engineering Change Notice)
- *USB 2.0 Connect Timing Update* (USB Engineering Change Notice as of April 4, 2013)
- *USB 2.0 VBUS Max Limit* (USB Engineering Change Notice)
- *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*, Revision 2.0 version 1.1a, July 27, 2012
- *Maximum VBUS Voltage* (USB OTGEH Engineering Change Notice)
- *Universal Serial Bus Micro-USB Cables and Connectors Specification*, Revision 1.01, 2007

## 61.2.2 USB—Overview

The USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB software provides a uniform view of the system for all application software, hiding implementation details to make application software more portable. It manages the dynamic attach and detach of peripherals.

There is only one host in any USB system. The USB interface to the host computer system is referred to as the Host Controller.

There may be multiple USB devices in any system such as human interface devices, speakers, printers, etc. USB devices present a standard USB interface in terms of comprehension, response, and standard capability.

The host initiates transactions to specific peripherals, whereas the device responds to control transactions. The device sends and receives data to and from the host using a standard USB data format. USB 2.0 full-speed /low-speed peripherals operate at 12Mbit/s or 1.5 Mbit/s.

For additional information, see the USB 2.0 specification.

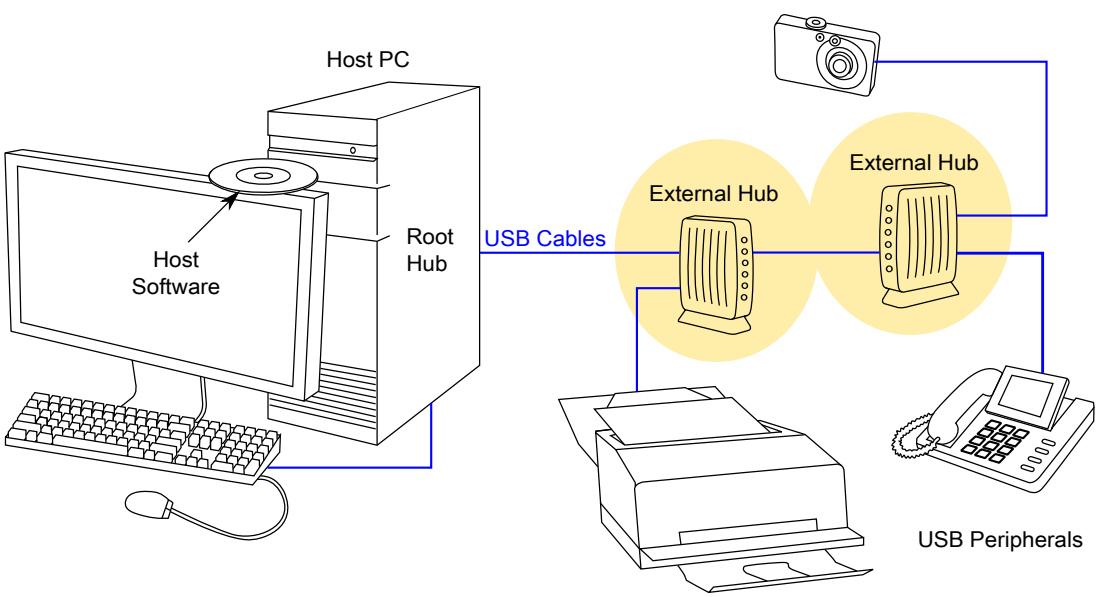



**Figure 61-4. Example USB 2.0 system configuration**

### 61.2.3 USBFS Features

- USB 1.1 and 2.0 compatible FS device controller
- 16 bidirectional endpoints
- DMA or FIFO data stream interfaces
- Low-power consumption
- IRC48M with clock-recovery is supported to eliminate the 48 MHz crystal. It is used for USB device-only implementation.
- Keep-alive feature is supported to power down system bus and CPU. USB can respond to IN with NAK and wake up for SETUP/OUT.

## 61.3 Functional description

USBFS communicates with the processor core through status registers, control registers, and data structures in memory.

### 61.3.1 Data Structures

To efficiently manage USB endpoint communications, USBFS implements a Buffer Descriptor Table (BDT) in system memory. See Figure 61-6.

### 61.3.2 On-chip transceiver required external components

USB system operation requires external components to ensure that driver output impedance, eye diagram, and VBUS cable fault tolerance requirements are met. DM and DP I/O pads must connect through series resistors (approximately 33 Ω each) to the USB connector on the application printed circuit board (PCB). Additionally, signal quality optimizes when these 33 Ω resistors are mounted closer to the processor than to the USB board-level connector. The USB transceiver includes:

- An internal 1.5 kΩ pullup resistor on the USB_DP line for full-speed device (controlled by USB_CONTROL[DPPULLUPNONOTG]
- Internal 15 kΩ pulldown resistors on the USB_DP and USB_DM signals, which are primarily intended for Host mode operation but are also useful in Device mode as explained below.

**NOTE**

For device operation, the internal 15 kΩ pulldowns should be enabled to keep the DP and DM ports in a known quiescent state when the VBUS detection software determines that the USB connection is not activated, including the cases when no cable to a host is present or when the USB port is not used. The internal 15 kΩ pulldowns should be controlled by USB_CTRL[PDE] in this case.

The following diagram provides an overview of device-only connections. For more details, see the *Kinetis Peripheral Module Quick Reference*(KQRUG).

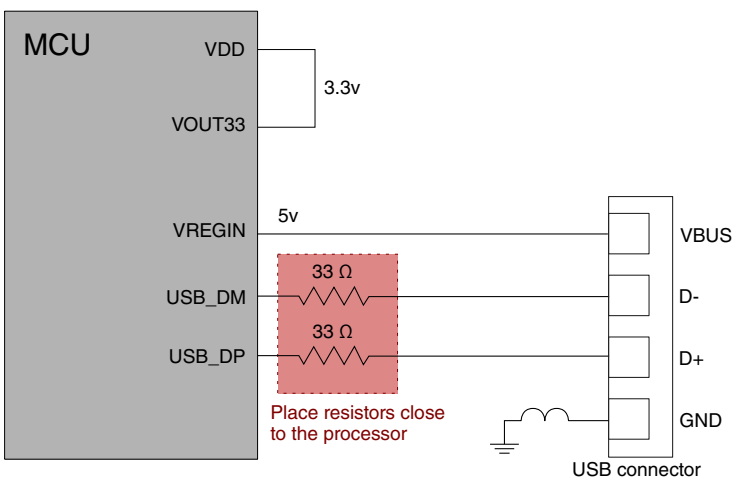



**Figure 61-5. Typical Device-only diagram (bus-powered)**

## 61.4 Programmers interface

This section discusses the major components of the programming model for the USB module.

### 61.4.1 Buffer Descriptor Table

To efficiently manage USB endpoint communications USBFS implements a Buffer Descriptor Table (BDT) in system memory.

- The BDT resides on a 512-byte boundary in system memory and is pointed to by the BDT Page Registers.
- Every endpoint direction requires two 8-byte Buffer Descriptor (BD) entries.

Therefore, a system with 16 fully bidirectional endpoints would require 512 bytes of system memory to implement the BDT. The two BD entries allows for an EVEN BD and ODD BD entry for each endpoint direction, which allows the microprocessor to process one BD while USBFS is processing the other BD. Double buffering BDs in this way allows USBFS to transfer data easily at the maximum throughput provided by USB.

Software should manage buffers for USBFS by updating the BDT when needed. This allows USBFS to efficiently manage data transmission and reception, while the microprocessor performs communication overhead processing and other function-dependent applications.

Because the buffers are shared between the microprocessor and USBFS, a simple semaphore mechanism is used to distinguish who is allowed to update the BDT and buffers in system memory. A semaphore (the OWN bit) is cleared to 0 when the BD entry is owned by the microprocessor.

- When OWN bit is 0, the microprocessor is allowed read and write access to the BD entry and the buffer in system memory.
- When OWN bit is 1, the BD entry and the buffer in system memory are owned by USBFS. USBFS now has full read and write access and the microprocessor must not modify the BD or its corresponding data buffer.

The BD also contains indirect address pointers to where the actual buffer resides in system memory. This indirect address mechanism is shown in the following diagram.

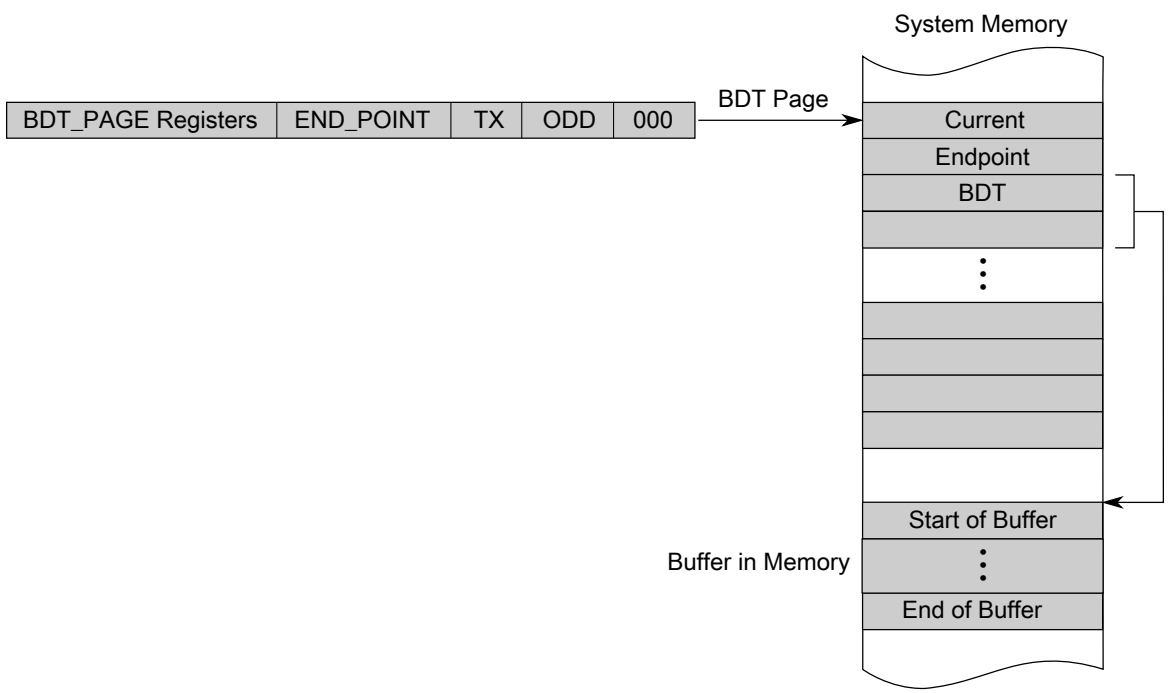



**Figure 61-6. Buffer descriptor table**

## 61.4.2 USB data transfers—Receive (Rx) and Transmit (Tx)

A USBFS core-centric nomenclature is used to describe the direction of the data transfer between USBFS core and USB. Throughout this chapter:

- "RX" (or "receive") describes transfers that move data from USB to memory.
- "TX" (or "transmit") describes transfers that move data from memory to USB.

The following table shows the data directions for USBFS.

**Table 61-1. Data direction for USB host or USB target**

| RX | TX |
|---|---|
| OUT or SETUP | IN |

## 61.4.3 Addressing BDT entries

An understanding of the addressing mechanism of the Buffer Descriptor Table is useful when accessing endpoint data via USBFS or microprocessor:

- The BDT occupies up to 512 bytes of system memory.
- 16 bidirectional endpoints can be supported with a full BDT of 512 bytes.
- 16 bytes are needed for each USB endpoint direction.
- Applications with fewer than 16 endpoints require less RAM to implement the BDT.
- The BDT Page Registers (BDT_PAGE) point to the starting location of the BDT.
- The BDT must be located on a 512-byte boundary in system memory.
- All enabled TX and RX endpoint BD entries are indexed into the BDT to allow easy access via USBFS or MCU core.

When a USB token on an enabled endpoint is received, USBFS uses its integrated DMA controller to interrogate the BDT. USBFS reads the corresponding endpoint BD entry to determine whether it (USBFS) owns the BD and corresponding buffer in system memory.

To compute the entry point into the BDT, the BDT_PAGE registers are concatenated with the current endpoint and the TX and ODD fields, to form a 32-bit address. This address mechanism is shown in the following tables:

**Table 61-2. BDT Address Calculation**

| 31:24 | 23:16 | 15:9 | 8:5 | 4 | 3 | 2:0 |
|---|---|---|---|---|---|---|
| BDT_PAGE_03 | BDT_PAGE_02 | BDT_PAGE_01[7:1] | Endpoint | TX | ODD | 000 |

**Table 61-3. BDT address calculation fields**

| Field | Description |
|---|---|
| BDT_PAGE | BDT_PAGE registers in the Control Register Block |
| ENDPOINT | ENDPOINT field from the USB TOKEN |
| TX | 1 for transmit transfers; 0 for receive transfers |
| ODD | It corresponds to the buffer currently in use; the buffers are used in a ping-pong fashion. Maintained within the USBFS SIE. |

## 61.4.4 Buffer Descriptors (BDs)

A buffer descriptor provides endpoint buffer control information for USBFS and the processor. The Buffer Descriptors have different meanings, depending on whether it is the USBFS or the processor reading the BD in memory.

The USBFS controller uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Whether to release ownership upon packet completion
- No address increment (FIFO mode)
- Whether data toggle synchronization is enabled
- How much data is to be transmitted or received
- Where the buffer resides in system memory

The processor uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received
- Where the buffer resides in system memory

The format for the BD is shown in the following table.

## Table 61-4. Buffer descriptor format

| 31:26 | 25:16 | 15:8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| RSVD | BC (10 bits) | RSVD | OWN | DATA0/1 | KEEP/ TOK_PID[3] | NINC/ TOK_PID[2] | DTS/ TOK_PID[1] | BDT_STALL/ TOK_PID[0] | 0 | 0 |
| Buffer Address (32-Bits) | | | | | | | | | | |

## Table 61-5. Buffer descriptor fields

| Field | Description |
|---|---|
| 31–26<br>RSVD | Reserved |
| 25–16<br>BC | Byte Count<br>Represents the 10-bit byte count. The USBFS SIE changes this field upon the completion of a RX transfer with the byte count of the data received. |
| 15–8<br>RSVD | Reserved |
| 7<br>OWN | Determines whether the processor or USBFS currently owns the buffer. Except when KEEP = 1, the SIE hands ownership back to the processor after completing the token by clearing this bit.<br>This must always be the last byte of the BD that the processor updates when it initializes a BD.<br>0 The processor has access to the BD. USBFS ignores all other fields in the BD.<br>1 USBFS has access to the BD. While USBFS owns the BD, the processor should not modify any other fields in the BD. |
| 6<br>DATA0/1 | Defines whether a DATA0 field (DATA0/1=0) or a DATA1 (DATA0/1=1) field was transmitted or received. It is unchanged by USBFS. |
| 5<br>KEEP/<br>TOK_PID[3] | This bit has two functions (KEEP or TOK_PID[3]):<br>• KEEP bit—When written by the processor, it serves as the KEEP bit. Typically, this bit is 1 with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed; the data is simply transferred to or from the FIFO. When KEEP is set, normally the NINC bit is also set to prevent address increment.<br>0 Allows USBFS to release the BD when a token has been processed.<br>1 This bit is unchanged by USBFS. Bit 3 of the current token PID is written back to the BD by USBFS.<br>• TOK_PID[3]—If the OWN bit is also set, then the BD remains owned by USBFS indefinitely; when written by USB, it serves as the TOK_PID[3] bit. Typically, this bit is 1 with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed; the data is simply transferred to or from the FIFO. When KEEP is set, normally the NINC bit is also set to prevent address increment.<br>0 or 1 Bit 3 of the current token PID is written back to the BD by USBFS. |
| 4<br>NINC/<br>TOK_PID[2] | No Increment (NINC)<br>Disables the DMA engine address increment. This forces the DMA engine to read or write from the same address. This is useful for endpoints when data needs to be *read from* or *written to* a single location (such as a FIFO). Typically this bit is set with the KEEP bit for ISO endpoints that are interfacing to a FIFO.<br>0 USBFS writes bit 2 of the current token PID to the BD. |

*Table continues on the next page...*

**Table 61-5. Buffer descriptor fields (continued)**

| Field | Description |
|---|---|
| | 1 This bit is unchanged by USBFS. |
| 3<br>DTS/<br>TOK_PID[1] | Setting this bit enables USBFS to perform Data Toggle Synchronization.<br>• If KEEP = 0, then bit 1 of the current token PID is written back to the BD.<br>• If KEEP = 1, then this bit is unchanged by USBFS.<br><br>0 Data Toggle Synchronization is disabled.<br><br>1 Enables USBFS to perform Data Toggle Synchronization. |
| 2<br>BDT_STALL<br>TOK_PID[0] | Setting this bit causes USBFS to issue a STALL handshake if a token is received by the SIE that would use the BDT in this location. The BDT is not consumed by the SIE (the owns bit remains set and the rest of the BDT is unchanged) when a BDT_STALL bit is set.<br>• If KEEP = 0, then bit 0 of the current token PID is written back to the BD.<br>• If KEEP = 1, then this bit is unchanged by USBFS.<br><br>0 No stall issued.<br><br>1 The BDT is not consumed by the SIE (the OWN bit remains set and the rest of the BDT is unchanged).<br><br>Setting BDT_STALL also causes the corresponding USB_ENDPT*n*[EPSTALL] bit to set. This causes USBOTG to issue a STALL handshake for both directions of the associated endpoint. To clear the stall condition:<br>1. Clear the associated USB_ENDPT*n*[EPSTALL] bit.<br>2. Write the BDT to clear OWN and BDT_STALL. |
| TOK_PID[n] | Bits [5:2] can also represent the current token PID. The current token PID is written back in to the BD by USBFS when a transfer completes. The values written back are the token PID values from the USB specification:<br>• 0x1 for an OUT token.<br>• 0x9 for an IN token.<br>• 0xD for a SETUP token. |
| 1–0<br>Reserved | Reserved, should read as zeroes. |
| ADDR[31:0] | Address<br><br>Represents the 32-bit buffer address in system memory; this address must also be 32-bit aligned. These bits are unchanged by USBFS. |

## 61.4.5 USB transaction

When USBFS transmits or receives data, it computes the BDT address using the address generation shown in "Addressing Buffer Descriptor Entries" table.

If OWN =1, then the following process occurs:

1. USBFS reads the BDT.
2. The SIE transfers the data via the DMA to or from the buffer pointed to by the ADDR field of the BD.
3. When the TOKEN is complete, USBFS updates the BDT, and if KEEP=0, then USBFS changes the OWN bit to 0.

4. The STAT register is updated and the TOK_DNE interrupt is set.
5. When the processor processes the TOK_DNE interrupt, it reads from the status register all the information needed to process the endpoint.
6. At this point, the processor allocates a new BD, so that additional USB data can be transmitted or received for that endpoint, and then processes the last BD.

The following figure shows a timeline of how a typical USB token is processed after the BDT is read and OWN=1.

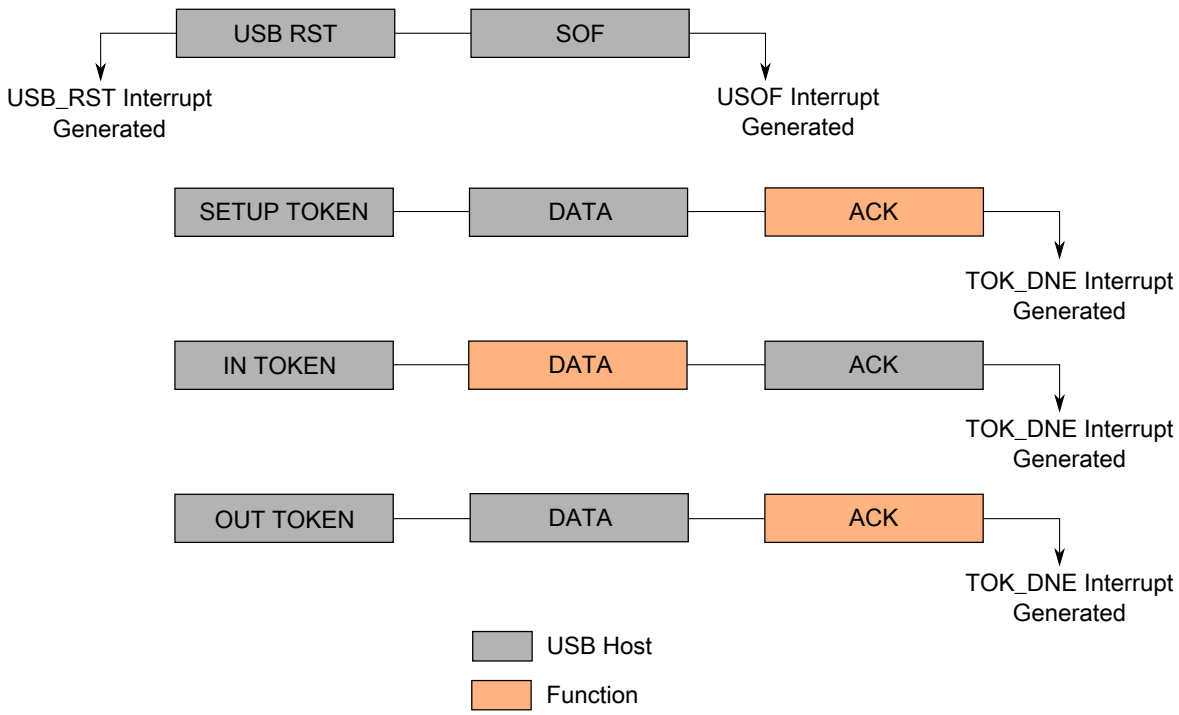



**Figure 61-7. USB token transaction**

The USB has two sources for the DMA overrun error: memory latency and oversized packets.

- **Memory Latency:** The memory latency may be too high and cause the receive FIFO to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.
- **Oversized Packets:** The packet received may be larger than the negotiated *MaxPacket* size. Typically, this is caused by a software bug. For DMA overrun errors due to oversized data packets, the USB specification is ambiguous, because it assumes correct software drivers on both sides. NAKing the packet can result in retransmission of the already oversized packet data. Therefore, in response to oversized packets, the USB core continues ACKing the packet for non-isochronous transfers.

**Table 61-6. USB responses to DMA overrun errors**

| Errors due to Memory Latency | Errors due to Oversized Packets |
|---|---|
| Non-Acknowledgment (NAK) or Bus Timeout (BTO) — See bit 4 in "Error Interrupt Status Register (ERRSTAT)" as appropriate for the class of transaction. | Continues acknowledging (ACKing) the packet for non-isochronous transfers. |
| — | The data written to memory is clipped to the MaxPacket size so as not to corrupt system memory. |
| The DMAERR bit is set in the ERRSTAT register for host and device modes of operation. Depending on the values of the INTENB and ERRENB register, the core may assert an interrupt to notify the processor of the DMA error. | Asserts ERRSTAT[DMAERR] ,which can trigger an interrupt and TOKDNE interrupt fires. Note: The TOK_PID field of the BDT is not 1111 because the DMAERR is not due to latency. |
| • In Host mode: the TOKDNE interrupt is generated and the TOK_PID field of the BDT is 1111 to indicate the DMA latency error. Host mode software can decide to retry or move to next scheduled item.<br>• In device mode: the BDT is not written back nor is the TOKDNE interrupt triggered because it is assumed that a second attempt is queued and will succeed in the future. | The packet length field written back to the BDT is the MaxPacket value that represents the length of the clipped data actually written to memory. |
| From here, the software can decide an appropriate course of action for future transactions, such as stalling the endpoint, canceling the transfer, disabling the endpoint, and others. | |

## 61.5 Memory map/Register definitions

This section provides the memory map and detailed descriptions of all USB interface registers.

### 61.5.1 USBFS memory mapping

The address space occupied by the USBFS registers is followed by 2048 bytes of embedded RAM used in Keep Alive mode. This USB RAM can also be used in Run mode.

### 61.5.2 USB register descriptions

#### 61.5.2.1 USB Memory map

USB0 base address: 4004_5000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Peripheral ID register (PERID) | 8 | RO | 04h |
| 4h | Peripheral ID Complement register (IDCOMP) | 8 | RO | FBh |
| 8h | Peripheral Revision register (REV) | 8 | RO | 33h |
| Ch | Peripheral Additional Info register (ADDINFO) | 8 | RO | 01h |
| 1Ch | OTG Control register (OTGCTL) | 8 | RW | 00h |
| 80h | Interrupt Status register (ISTAT) | 8 | W1C | 00h |
| 84h | Interrupt Enable register (INTEN) | 8 | RW | 00h |
| 88h | Error Interrupt Status register (ERRSTAT) | 8 | W1C | 00h |
| 8Ch | Error Interrupt Enable register (ERREN) | 8 | RW | 00h |
| 90h | Status register (STAT) | 8 | RO | 00h |
| 94h | Control register (CTL) | 8 | RW | 00h |
| 98h | Address register (ADDR) | 8 | RW | 00h |
| 9Ch | BDT Page register 1 (BDTPAGE1) | 8 | RW | 00h |
| A0h | Frame Number register Low (FRMNUML) | 8 | RW | 00h |
| A4h | Frame Number register High (FRMNUMH) | 8 | RW | 00h |
| B0h | BDT Page Register 2 (BDTPAGE2) | 8 | RW | 00h |
| B4h | BDT Page Register 3 (BDTPAGE3) | 8 | RW | 00h |
| C0h | Endpoint Control register (ENDPT0) | 8 | RW | 00h |
| C4h | Endpoint Control register (ENDPT1) | 8 | RW | 00h |
| C8h | Endpoint Control register (ENDPT2) | 8 | RW | 00h |
| CCh | Endpoint Control register (ENDPT3) | 8 | RW | 00h |
| D0h | Endpoint Control register (ENDPT4) | 8 | RW | 00h |
| D4h | Endpoint Control register (ENDPT5) | 8 | RW | 00h |
| D8h | Endpoint Control register (ENDPT6) | 8 | RW | 00h |
| DCh | Endpoint Control register (ENDPT7) | 8 | RW | 00h |
| E0h | Endpoint Control register (ENDPT8) | 8 | RW | 00h |
| E4h | Endpoint Control register (ENDPT9) | 8 | RW | 00h |
| E8h | Endpoint Control register (ENDPT10) | 8 | RW | 00h |
| ECh | Endpoint Control register (ENDPT11) | 8 | RW | 00h |
| F0h | Endpoint Control register (ENDPT12) | 8 | RW | 00h |
| F4h | Endpoint Control register (ENDPT13) | 8 | RW | 00h |
| F8h | Endpoint Control register (ENDPT14) | 8 | RW | 00h |
| FCh | Endpoint Control register (ENDPT15) | 8 | RW | 00h |
| 100h | USB Control register (USBCTRL) | 8 | RW | C0h |
| 104h | USB OTG Observe register (OBSERVE) | 8 | RO | 50h |
| 108h | USB OTG Control register (CONTROL) | 8 | RW | 00h |
| 10Ch | USB Transceiver Control register 0 (USBTRC0) | 8 | RW | 00h |
| 124h | Keep Alive mode control (KEEP_ALIVE_CTRL) | 8 | RW | 08h |
| 128h | Keep Alive mode wakeup control (KEEP_ALIVE_WKCTRL) | 8 | RW | 01h |
| 12Ch | Miscellaneous Control register (MISCCTRL) | 8 | RW | 00h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 130h | Peripheral mode stall disable for endpoints 7 to 0 in IN direction (STALL_IL_DIS) | 8 | RW | 00h |
| 134h | Peripheral mode stall disable for endpoints 15 to 8 in IN direction (STALL_IH_DIS) | 8 | RW | 00h |
| 138h | Peripheral mode stall disable for endpoints 7 to 0 in OUT direction (STALL_OL_DIS) | 8 | RW | 00h |
| 13Ch | Peripheral mode stall disable for endpoints 15 to 8 in OUT direction (STALL_OH_DIS) | 8 | RW | 00h |
| 140h | USB Clock recovery control (CLK_RECOVER_CTRL) | 8 | RW | 00h |
| 144h | IRC48MFIRC oscillator enable register (CLK_RECOVER_IRC_EN) | 8 | RW | 01h |
| 154h | Clock recovery combined interrupt enable (CLK_RECOVER_INT_EN) | 8 | RW | 10h |
| 15Ch | Clock recovery separated interrupt status (CLK_RECOVER_INT_STATUS) | 8 | W1C | 00h |

## 61.5.2.2 Peripheral ID register (PERID)

### 61.5.2.2.1 Offset

| Register | Offset |
|---|---|
| PERID | 0h |

### 61.5.2.2.2 Function

Reads back the value of 0x04. This value is defined for the USB peripheral.

### 61.5.2.2.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | ID | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### 61.5.2.2.4 Fields

| Field | Function |
|---|---|
| 7-6 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 5-0<br>ID | Peripheral Identification<br>This field always reads 0x4h. |

## 61.5.2.3 Peripheral ID Complement register (IDCOMP)

### 61.5.2.3.1 Offset

| Register | Offset |
|---|---|
| IDCOMP | 4h |

### 61.5.2.3.2 Function

Reads back the complement of the Peripheral ID register. For the USB peripheral, the value is 0xFB.

### 61.5.2.3.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 1 | | NID | | | | | |
| W | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

### 61.5.2.3.4 Fields

| Field | Function |
|---|---|
| 7-6<br>— | Reserved<br>These bits always read as "1" each. |
| 5-0<br>NID | NID<br>The one's complement of PERID[ID] bits. |

## 61.5.2.4 Peripheral Revision register (REV)

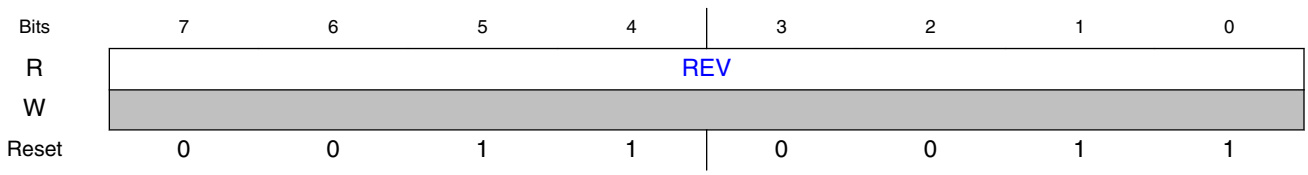
#### 61.5.2.4.1 Offset

| Register | Offset |
|---|---|
| REV | 8h |

#### 61.5.2.4.2 Function

Contains the revision number of the USB module.

#### 61.5.2.4.3 Diagram




#### 61.5.2.4.4 Fields

| Field | Function |
|---|---|
| 7-0<br>REV | Revision<br>The revision number of the USB Core. |

### 61.5.2.5 Peripheral Additional Info register (ADDINFO)

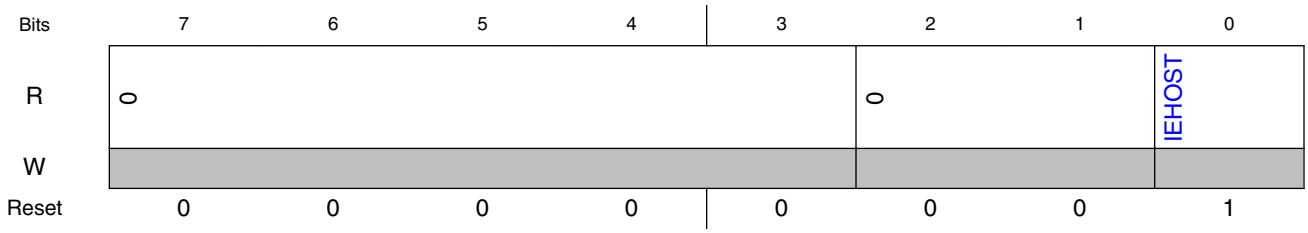
#### 61.5.2.5.1 Offset

| Register | Offset |
|---|---|
| ADDINFO | Ch |

#### 61.5.2.5.2 Function

Reads back the value of the Host Enable bit.

#### 61.5.2.5.3 Diagram




#### 61.5.2.5.4 Fields

| Field | Function |
|---|---|
| 7-3<br>— | Reserved |
| 2-1<br>— | Reserved |
| 0<br>IEHOST | Host mode enable bit<br>This bit is set to "1" if Host mode is enabled. |

### 61.5.2.6 OTG Control register (OTGCTL)

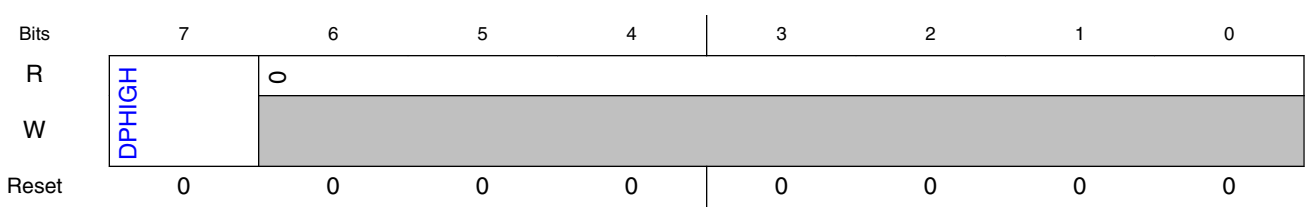
#### 61.5.2.6.1 Offset

| Register | Offset |
|---|---|
| OTGCTL | 1Ch |

#### 61.5.2.6.2 Function

Controls the operation of VBUS and Data Line termination resistors.

#### 61.5.2.6.3 Diagram



#### 61.5.2.6.4 Fields

| Field | Function |
|---|---|
| 7<br>DPHIGH | D+ Data Line pullup resistor enable<br>0b - D+ pullup resistor is not enabled<br>1b - D+ pullup resistor is enabled |
| 6-0<br>— | Reserved |

### 61.5.2.7 Interrupt Status register (ISTAT)

#### 61.5.2.7.1 Offset

| Register | Offset |
|---|---|
| ISTAT | 80h |

#### 61.5.2.7.2 Function

Contains fields for each of the interrupt sources within the USB Module. Each of these fields are qualified with their respective interrupt enable bits.

- After an interrupt bit has been set to "1", the interrupt bit can only be cleared by writing a "1" to the interrupt bit.
- This register contains the value of 0x00 after a reset.

#### 61.5.2.7.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | STALL | 0 | RESUME | SLEEP | TOKDNE | SOFTOK | ERROR | USBRST |
| W | W1C | | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 61.5.2.7.4 Fields

| Field | Function |
|---|---|
| 7<br>STALL | Stall Interrupt<br>In Device mode, STALL bit is asserted when a STALL handshake is sent by the SIE. |
| 6<br>— | Reserved |
| 5<br>RESUME | RESUME<br>RESUME bit is set when a K-state is observed on the DP/DM signals for 2.5 µs. When not in Suspend mode, this interrupt must be disabled. |
| 4<br>SLEEP | Sleep<br>SLEEP bit is set when the USB Module detects a constant idle on the USB bus for 3 ms. The sleep timer is reset by activity on the USB bus. |
| 3<br>TOKDNE | Current token processing<br>TOKDNE bit is set when the current token being processed has completed. The processor must immediately read the STATUS (STAT) register to determine the EndPoint and BD used for this token. Clearing TOKDNE bit:<br>• causes STAT to be cleared<br>• or the STAT holding register to be loaded into the STAT register. |
| 2<br>SOFTOK | Start Of Frame (SOF) token<br>SOFTOK bit is set when the USB Module receives a Start Of Frame (SOF) token. |
| 1<br>ERROR | Error<br>ERROR bit is set when any of the error conditions within Error Interrupt Status (ERRSTAT) register occur. The processor must then read the ERRSTAT register to determine the source of the error. |
| 0<br>USBRST | USB Reset<br>USBRST bit is set when the USB Module has decoded a valid USB reset. This informs the processor that it should write 0x00 into the address register and enable endpoint 0.<br>USBRST is set after a USB reset has been detected for 2.5 microseconds. USBRST is not asserted again until the USB reset condition has been removed and the USB reset condition is reasserted. |

## 61.5.2.8 Interrupt Enable register (INTEN)

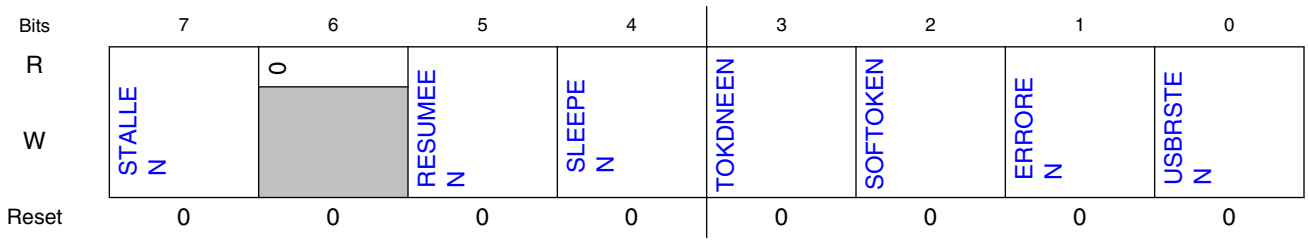
### 61.5.2.8.1 Offset

| Register | Offset |
|---|---|
| INTEN | 84h |

### 61.5.2.8.2 Function

Contains enable fields for each of the interrupt sources within the USB Module. Setting any of these bits enables the respective interrupt source in the ISTAT register. After a reset, INTEN register contains value 0x00.

### 61.5.2.8.3 Diagram




### 61.5.2.8.4 Fields

| Field | Function |
|---|---|
| 7<br>STALLEN | STALL Interrupt Enable<br>0b - Diasbles the STALL interrupt.<br>1b - Enables the STALL interrupt. |
| 6<br>— | Reserved |
| 5<br>RESUMEEN | RESUME Interrupt Enable<br>0b - Disables the RESUME interrupt.<br>1b - Enables the RESUME interrupt. |
| 4<br>SLEEPEN | SLEEP Interrupt Enable<br>0b - Disables the SLEEP interrupt.<br>1b - Enables the SLEEP interrupt. |
| 3<br>TOKDNEEN | TOKDNE Interrupt Enable<br>0b - Disables the TOKDNE interrupt.<br>1b - Enables the TOKDNE interrupt. |
| 2<br>SOFTOKEN | SOFTOK Interrupt Enable<br>0b - Disbles the SOFTOK interrupt.<br>1b - Enables the SOFTOK interrupt. |
| 1<br>ERROREN | ERROR Interrupt Enable<br>0b - Disables the ERROR interrupt.<br>1b - Enables the ERROR interrupt. |
| 0<br>USBRSTEN | USBRST Interrupt Enable<br>0b - Disables the USBRST interrupt.<br>1b - Enables the USBRST interrupt. |

# 61.5.2.9 Error Interrupt Status register (ERRSTAT)

## 61.5.2.9.1 Offset

| Register | Offset |
|---|---|
| ERRSTAT | 88h |

## 61.5.2.9.2 Function

Contains enable bits for each of the error sources within the USB Module. Each of these bits are qualified with their respective error enable bits.

- All bits of this register are logically OR'd together and the result placed in the ERROR bit of the ISTAT register.
- After an interrupt bit has been set it may only be cleared by writing a one to the respective interrupt bit.
- Each bit is set as soon as the error condition is detected; therefore, the interrupt does not typically correspond with the end of a token being processed.
- After a reset, ERRSTAT register contains value 0x00.

## 61.5.2.9.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | BTSER R | OWNERR | DMAERR | BTOER R | DFN8 | CRC16 | CRC5EOF | PIDER R |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 61.5.2.9.4 Fields

| Field | Function |
|---|---|
| 7<br>BTSERR | Bit stuff error<br>BTSERR bit is set when a bit stuff error is detected. If BTSERR is set, then the corresponding packet is rejected due to the error. |
| 6<br>OWNERR | OWNERR<br>OWNERR bit is valid when the USB Module is operating in Peripheral mode (CTL[HOSTMODEEN] = 0). OWNERR is set (to 1) if the USB Module requires a new BD for SETUP, ISO IN, or ISO OUT transfer while a new BD is not available. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 5<br>DMAERR | DMAERR<br>DMAERR bit is set if the USB Module has requested a DMA access to read a new BDT, but has not been given the bus before it needs to receive or transmit data.<br>• If processing a TX transfer, this would cause a transmit data underflow condition.<br>• If processing a RX transfer, this would cause a receive data overflow condition.<br><br>DMAERR interrupt is useful when developing device arbitration hardware for the microprocessor and the USB module, to minimize bus request and bus grant latency. DMAERR bit is also set if a data packet to or from the host is larger than the buffer size allocated in the BDT; in this case, the data packet is truncated as it is put in buffer memory. |
| 4<br>BTOERR | Bus turnaround timeout error<br>BTOERR bit is set when a bus turnaround timeout error occurs. The USB module contains a bus turnaround timer that keeps track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of a IN TOKEN. If more than 16 bit times are counted from the previous EOP before a transition from IDLE, then a bus turnaround timeout error occurs. |
| 3<br>DFN8 | Data field not 8 bits (in length)<br>DFN8 bit is set if the data field received was not 8 bits in length. USB Specification 1.0 requires that data fields be an integral number of bytes. If the data field was not an integral number of bytes, then DFN8 bit is set (to 1). |
| 2<br>CRC16 | CRC16 error<br>CRC16 bit is set (to 1) when a data packet is rejected due to a CRC16 error. |
| 1<br>CRC5EOF | CRC5 error or end of frame error<br>This error interrupt has two functions:<br>• When the USB Module is operating in Peripheral mode (CTL[HOSTMODEEN] = 0), this interrupt detects CRC5 errors in the token packets generated by the host. If CRC5EOF is 1, then the token packet was rejected due to a CRC5 error.<br>• When the USB Module is operating in Host mode (CTL[HOSTMODEEN] = 1), this interrupt detects End Of Frame (EOF) error conditions. This type of error occurs when the USB Module is transmitting or receiving data and the SOF counter reaches zero. This interrupt is useful when developing USB packet scheduling software, to ensure that no USB transactions cross the start of the next frame. |
| 0<br>PIDERR | PID error<br>PIDERR bit is set when the PID check field fails. |

## 61.5.2.10 Error Interrupt Enable register (ERREN)

### 61.5.2.10.1 Offset

| Register | Offset |
|---|---|
| ERREN | 8Ch |

### 61.5.2.10.2 Function

Contains enable bits for each of the error interrupt sources within the USB module.

- Setting any of these bits enables the respective interrupt source in ERRSTAT.
- Each bit is set as soon as the error condition is detected; therefore, the interrupt does not typically correspond with the end of a token being processed.
- After a reset, ERREN register contains value 0x00.

### 61.5.2.10.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | BTSERREN | OWNERREN | DMAERREN | BTOERREN | DFN8EN | CRC16EN | CRC5EOFEN | PIDERREN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.10.4 Fields

| Field | Function |
|---|---|
| 7<br>BTSERREN | BTSERR Interrupt Enable<br>0b - Disables the BTSERR interrupt.<br>1b - Enables the BTSERR interrupt. |
| 6<br>OWNERREN | OWNERR Interrupt Enable<br>This field is valid when the USB module is operating in Peripheral mode (CTL[HOSTMODEEN] = 0).<br>0b - Disables the OWNERR interrupt.<br>1b - Enables the OWNERR interrupt. |
| 5<br>DMAERREN | DMAERR Interrupt Enable<br>0b - Disables the DMAERR interrupt.<br>1b - Enables the DMAERR interrupt. |
| 4<br>BTOERREN | BTOERR Interrupt Enable<br>0b - Disables the BTOERR interrupt.<br>1b - Enables the BTOERR interrupt. |
| 3<br>DFN8EN | DFN8 Interrupt Enable<br>0b - Disables the DFN8 interrupt.<br>1b - Enables the DFN8 interrupt. |
| 2<br>CRC16EN | CRC16 Interrupt Enable<br>0b - Disables the CRC16 interrupt.<br>1b - Enables the CRC16 interrupt. |
| 1<br>CRC5EOFEN | CRC5/EOF Interrupt Enable<br>0b - Disables the CRC5/EOF interrupt.<br>1b - Enables the CRC5/EOF interrupt. |
| 0<br>PIDERREN | PIDERR Interrupt Enable<br>0b - Disables the PIDERR interrupt.<br>1b - Enters the PIDERR interrupt. |

## 61.5.2.11 Status register (STAT)

### 61.5.2.11.1 Offset

| Register | Offset |
|---|---|
| STAT | 90h |

### 61.5.2.11.2 Function

Reports the transaction status within the USB module.

When the processor's interrupt controller has received a TOKDNE, the Interrupt Status Register must be read to determine the status of the previous endpoint communication. The data in the status register is valid when the TOKDNE interrupt is asserted. The Status register is actually a read window into a status FIFO maintained by the USB module. When the USB module uses a BD, it updates the Status register.

If another USB transaction is performed before the TOKDNE interrupt is serviced, then the USB module stores the status of the next transaction in the STAT FIFO. Thus, STAT is actually a 4-byte FIFO that allows the processor core to process one transaction while the SIE is processing the next transaction. Clearing the TOKDNE bit in the ISTAT register causes the SIE to update STAT with the contents of the next STAT value. If the data in the STAT holding register is valid, then the SIE immediately reasserts to TOKDNE interrupt.

### 61.5.2.11.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ENDP | | | | TX | ODD | 0 | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.11.4 Fields

| Field | Function |
|---|---|
| 7-4<br>ENDP | ENDP<br>This 4-bit field encodes the endpoint address that received or transmitted the previous token. This allows the processor core to determine the BDT entry that was updated by the last USB transaction. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 3<br>TX | Transmit Indicator<br>0b - The most recent transaction was a receive operation.<br>1b - The most recent transaction was a transmit operation. |
| 2<br>ODD | Odd bank<br>ODD bit is set (to 1) if the last buffer descriptor *that was updated*, was in the odd bank of the BDT. |
| 1-0<br>— | Reserved |

## 61.5.2.12 Control register (CTL)

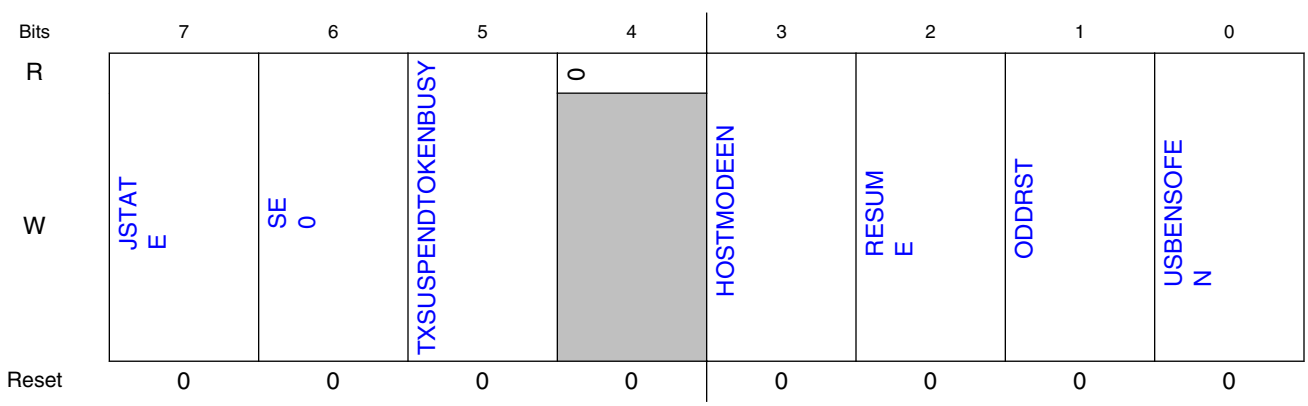
### 61.5.2.12.1 Offset

| Register | Offset |
|---|---|
| CTL | 94h |

### 61.5.2.12.2 Function

Provides various control and configuration information for the USB module.

### 61.5.2.12.3 Diagram




### 61.5.2.12.4 Fields

| Field | Function |
|---|---|
| 7 | Live USB differential receiver JSTATE signal |

*Table continues on the next page...*

| Field | Function |
|---|---|
| JSTATE | The polarity of this signal is affected by the current state of LSEN . |
| 6<br>SE0 | Live USB Single-Ended Zero signal |
| 5<br>TXSUSPENDTOKENBUSY | TXSUSPEND or TOKENBUSY<br>In Device mode, TXD_SUSPEND is set when the SIE has disabled packet transmission and reception. Clearing TXD_SUSPEND bit allows the SIE to continue token processing. TXD_SUSPEND bit is set to 1 by the SIE when a SETUP Token is received, allowing software to dequeue any pending packet transactions in the BDT before resuming token processing. |
| 4<br>— | Reserved |
| 3<br>HOSTMODEEN | Host mode enable<br>0b - USB Module operates in Device mode.<br>1b - USB Module operates in Host mode. In Host mode, the USB module performs USB transactions under the programmed control of the host processor. |
| 2<br>RESUME | Resume<br>When set to 1, RESUME bit enables the USB Module to execute resume signaling, this allows the USB Module to perform remote wake-up. Software must set RESUME to 1 for the required amount of time, and then clear RESUME to 0. If HOSTMODEEN is set, then the USB module appends a Low Speed End-of-Packet to the Resume signaling when the RESUME bit is cleared (to 0). |
| 1<br>ODDRST | ODDRST<br>Setting this bit to 1 resets all the BDT ODD ping/pong fields to 0, which then specifies the EVEN BDT bank. |
| 0<br>USBENSOFEN | USB Enable<br>Setting USBENSOFEN bit to 1 enables the USB-FS to operate; clearing USBENSOFEN bit (to 0) disables the USB-FS. Setting USBENSOFEN bit to 1 causes the SIE to reset all of its ODD bits to the BDTs; therefore, setting USBENSOFEN bit resets much of the logic in the SIE.<br>0b - Disables the USB Module.<br>1b - Enables the USB Module. |

## 61.5.2.13 Address register (ADDR)

### 61.5.2.13.1 Offset

| Register | Offset |
|---|---|
| ADDR | 98h |

### 61.5.2.13.2 Function

In Peripheral mode (CTL[HOSTMODEEN] = 0), ADDR holds the unique USB address that the USB module decodes. CTL[USBEN] must be 1.

After the reset input becomes active or the USB module decodes a USB reset signal, the Address register is reset to 0x00. This action initializes the Address register to decode address 0x00, as required by the USB specification.

### 61.5.2.13.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | ADDR | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.13.4 Fields

| Field | Function |
|---|---|
| 7<br>— | Reserved |
| 6-0<br>ADDR | USB Address<br>Defines the USB address that the USB module decodes in Peripheral mode. |

## 61.5.2.14 BDT Page register 1 (BDTPAGE1)

### 61.5.2.14.1 Offset

| Register | Offset |
|---|---|
| BDTPAGE1 | 9Ch |

### 61.5.2.14.2 Function

Provides address bits 15 through 9 of the base address where the current Buffer Descriptor Table (BDT) resides in system memory. See Buffer Descriptor Table. The 32-bit BDT Base Address is always aligned on 512-byte boundaries, so bits 8 through 0 of the base address are always zero.

### 61.5.2.14.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | BDTBA | | | | | | | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.14.4 Fields

| Field | Function |
|---|---|
| 7-1<br>BDTBA | BDTBA<br>Provides address bits 15 through 9 of the BDT base address. |
| 0<br>— | Reserved |

## 61.5.2.15 Frame Number register Low (FRMNUML)

### 61.5.2.15.1 Offset

| Register | Offset |
|---|---|
| FRMNUML | A0h |

### 61.5.2.15.2 Function

The Frame Number registers (low and high) contain the 11-bit frame number. These registers are updated with the current frame number whenever a SOF TOKEN is received.

### 61.5.2.15.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FRM | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 61.5.2.15.4 Fields

| Field | Function |
|---|---|
| 7-0<br>FRM | FRM<br>This 8-bit field and the 3-bit field in the Frame Number Register High are used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. |

### 61.5.2.16 Frame Number register High (FRMNUMH)

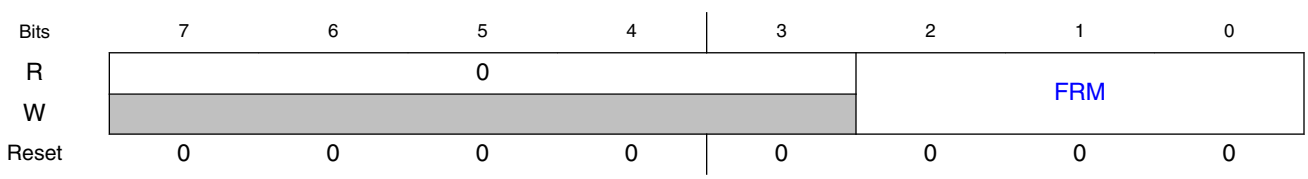
#### 61.5.2.16.1 Offset

| Register | Offset |
|---|---|
| FRMNUMH | A4h |

#### 61.5.2.16.2 Function

The Frame Number registers (low and high) contain the 11-bit frame number. These registers are updated with the current frame number whenever a SOF TOKEN is received.

#### 61.5.2.16.3 Diagram




#### 61.5.2.16.4 Fields

| Field | Function |
|---|---|
| 7-3<br>— | Reserved |
| 2-0<br>FRM | FRM<br>This 3-bit field and the 8-bit field in the Frame Number Register Low are used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. |

## 61.5.2.17 BDT Page Register 2 (BDTPAGE2)

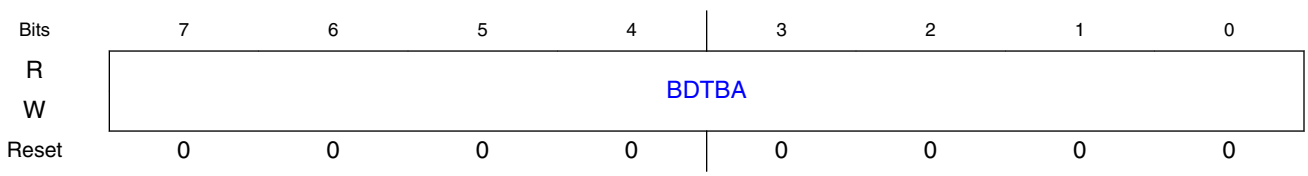
### 61.5.2.17.1 Offset

| Register | Offset |
|---|---|
| BDTPAGE2 | B0h |

### 61.5.2.17.2 Function

Contains an 8-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. See Buffer Descriptor Table.

### 61.5.2.17.3 Diagram




### 61.5.2.17.4 Fields

| Field | Function |
|---|---|
| 7-0<br>BDTBA | BDTBA<br>Provides address bits 23 through 16 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory. |

## 61.5.2.18 BDT Page Register 3 (BDTPAGE3)

### 61.5.2.18.1 Offset

| Register | Offset |
|---|---|
| BDTPAGE3 | B4h |

#### 61.5.2.18.2 Function

Contains an 8-bit value used to compute the address where the current Buffer Descriptor Table (BDT) resides in system memory. See Buffer Descriptor Table.

#### 61.5.2.18.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | BDTBA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 61.5.2.18.4 Fields

| Field | Function |
|---|---|
| 7-0<br>BDTBA | BDTBA<br>Provides address bits 31 through 24 of the BDT base address that defines the location of Buffer Descriptor Table resides in system memory. |

### 61.5.2.19 Endpoint Control register (ENDPT0 - ENDPT15)

#### 61.5.2.19.1 Offset

For a = 0 to 15:

| Register | Offset |
|---|---|
| ENDPTa | C0h + (a × 4h) |

#### 61.5.2.19.2 Function

Contains the endpoint control bits for each of the 16 endpoints available within the USB module for a decoded address. The format for these registers is shown in the following figure.

- Endpoint 0 (ENDPT0) is associated with control pipe 0, which is required for all USB functions. Therefore, after a USBRST interrupt occurs, the processor core should set ENDPT0 to contain 0x0D. In Host mode, ENDPT0 is used to determine the handshake, retry and low speed characteristics of the host transfer. Common

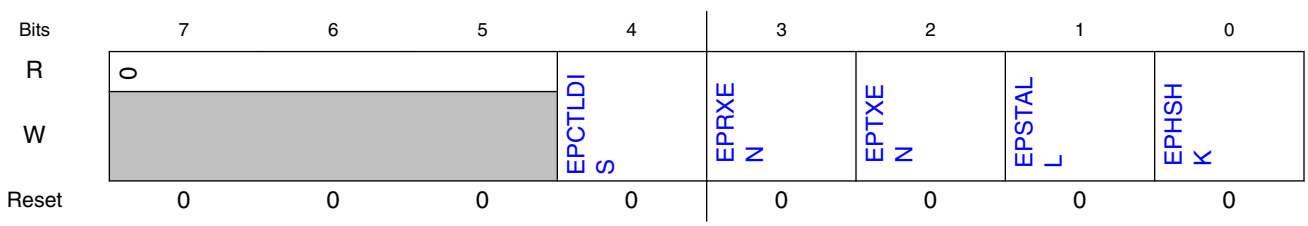
values to use for ENDPT0 in Host mode are 0x4D for Control, Bulk, and Interrupt transfers, and 0x4C for Isochronous transfers.

- For Control, Bulk and Interrupt transfers, the EPHSHK bit should be 1. For Isochronous transfers, EPHSHK bit should be 0.
- The 3 bits EPCTLDIS, EPRXEN, and EPTXEN define if an endpoint is enabled and also define the direction of the endpoint. The endpoint enable/direction control is defined in the next table.

**Table 61-7. Endpoint enable and direction control**

| EPCTLDIS | EPRXEN | EPTXEN | Endpoint enable/direction control |
|---|---|---|---|
| X | 0 | 0 | Disable endpoint |
| X | 0 | 1 | Enable endpoint for Tx transfers only |
| X | 1 | 0 | Enable endpoint for Rx transfers only |
| 1 | 1 | 1 | Enable endpoint for Rx and Tx transfers |
| 0 | 1 | 1 | Enable Endpoint for RX and TX as well as control (SETUP) transfers. |

## 61.5.2.19.3 Diagram




## 61.5.2.19.4 Fields

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4<br>EPCTLDIS | Control (SETUP) transfer disable<br>• When EPCTLDIS bit is set to 1, it disables control (SETUP) transfers.<br>• When EPCTLDIS bit is cleared (to 0), control transfers are enabled, but only if the EPRXEN and EPTXEN bits are also set. See Table 61-7. |
| 3<br>EPRXEN | Endpoint for RX transfers enable<br>When EPRXEN bit is set to 1, it enables the endpoint for RX transfers. See Table 61-7. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>EPTXEN | Endpoint for TX transfers enable<br>When EPTXEN bit is set to 1, it enables the endpoint for TX transfers. See Table 61-7. |
| 1<br>EPSTALL | Endpoint stalled<br>When EPSTALL bit is set to 1, it indicates that the endpoint is stalled.<br>• EPSTALL bit has priority over all other control bits in this register, but the EPSTALL bit is only valid if EPTXEN=1 or EPRXEN=1.<br>• Any access to this endpoint causes the USB Module to return a STALL handshake.<br>• After an endpoint is stalled, intervention is required from the Host Controller. |
| 0<br>EPHSHK | Endpoint handshaking enable<br>When EPHSHK bit is set to 1, it enables an endpoint to perform handshaking during a transaction to this endpoint. EPHSHK bit is generally 1 unless the endpoint is Isochronous. |

## 61.5.2.20 USB Control register (USBCTRL)

### 61.5.2.20.1 Offset

| Register | Offset |
|---|---|
| USBCTRL | 100h |

### 61.5.2.20.2 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SUSP | PDE | UARTCHLS | UARTSEL | 0 | 0 | | |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.20.3 Fields

| Field | Function |
|---|---|
| 7<br>SUSP | Suspend<br>Places the USB transceiver into the Suspend state.<br>0b - USB transceiver is not in the Suspend state.<br>1b - USB transceiver is in the Suspend state. |
| 6<br>PDE | Pulldown enable<br>Enables the weak pulldowns on the USB transceiver. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - Weak pulldowns are disabled on D+ and D–.<br>1b - Weak pulldowns are enabled on D+ and D–. |
| 5<br>UARTCHLS | UART Signal Channel Select<br>UARTCHLS field is valid only when USB signals are selected to be used as UART signals.<br>0b - USB DP/DM signals are used as UART TX/RX.<br>1b - USB DP/DM signals are used as UART RX/TX. |
| 4<br>UARTSEL | UART Select<br>Selects USB signals to be used as UART signals.<br>0b - USB signals are not used as UART signals.<br>1b - USB signals are used as UART signals. |
| 3<br>— | Reserved |
| 2-0<br>— | Reserved |

## 61.5.2.21 USB OTG Observe register (OBSERVE)

### 61.5.2.21.1 Offset

| Register | Offset |
|---|---|
| OBSERVE | 104h |

### 61.5.2.21.2 Function

Provides visibility on the state of the pull-ups and pull-downs at the transceiver, which is useful when interfacing to an external OTG control module via a serial interface.

### 61.5.2.21.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DPPU | DPPD | 0 | DMPD | 0 | | | |
| W | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

#### 61.5.2.21.4 Fields

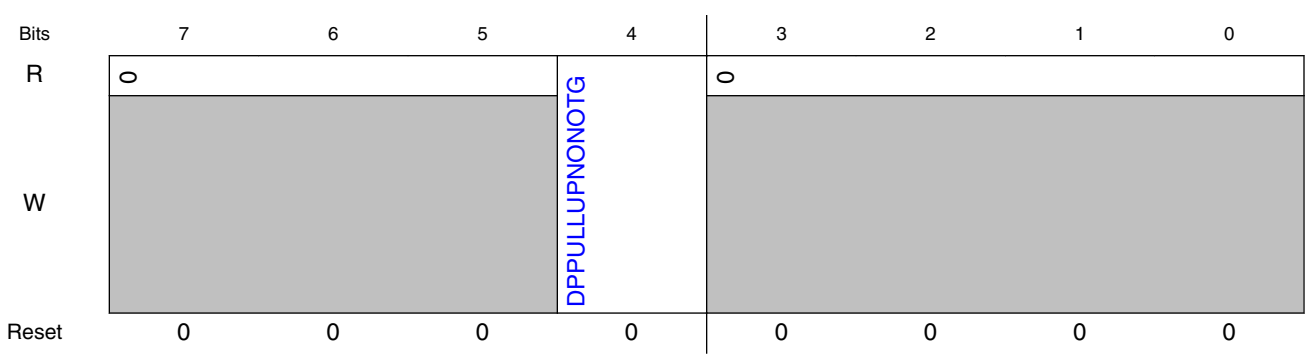
| Field | Function |
|---|---|
| 7<br>DPPU | DPPU<br>Provides observability of the D+ Pullup enable at the USB transceiver.<br>0b - D+ pullup disabled.<br>1b - D+ pullup enabled. |
| 6<br>DPPD | DPPD<br>Provides observability of the D+ Pulldown enable at the USB transceiver.<br>0b - D+ pulldown is disabled.<br>1b - D+ pulldown is enabled. |
| 5<br>— | Reserved |
| 4<br>DMPD | DMPD<br>Provides observability of the D- Pulldown enable at the USB transceiver.<br>0b - D– pulldown is disabled.<br>1b - D– pulldown is enabled. |
| 3-0<br>— | Reserved |

## 61.5.2.22 USB OTG Control register (CONTROL)

### 61.5.2.22.1 Offset

| Register | Offset |
|---|---|
| CONTROL | 108h |

### 61.5.2.22.2 Diagram



#### 61.5.2.22.3 Fields

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4<br>DPPULLUPNONOTG | DPPULLUPNONOTG<br>Provides control of the DP Pullup in USB, if USB is configured in non-OTG Device mode.<br>0b - DP Pullup in non-OTG Device mode is not enabled.<br>1b - DP Pullup in non-OTG Device mode is enabled. |
| 3-0<br>— | Reserved |

### 61.5.2.23 USB Transceiver Control register 0 (USBTRC0)

#### 61.5.2.23.1 Offset

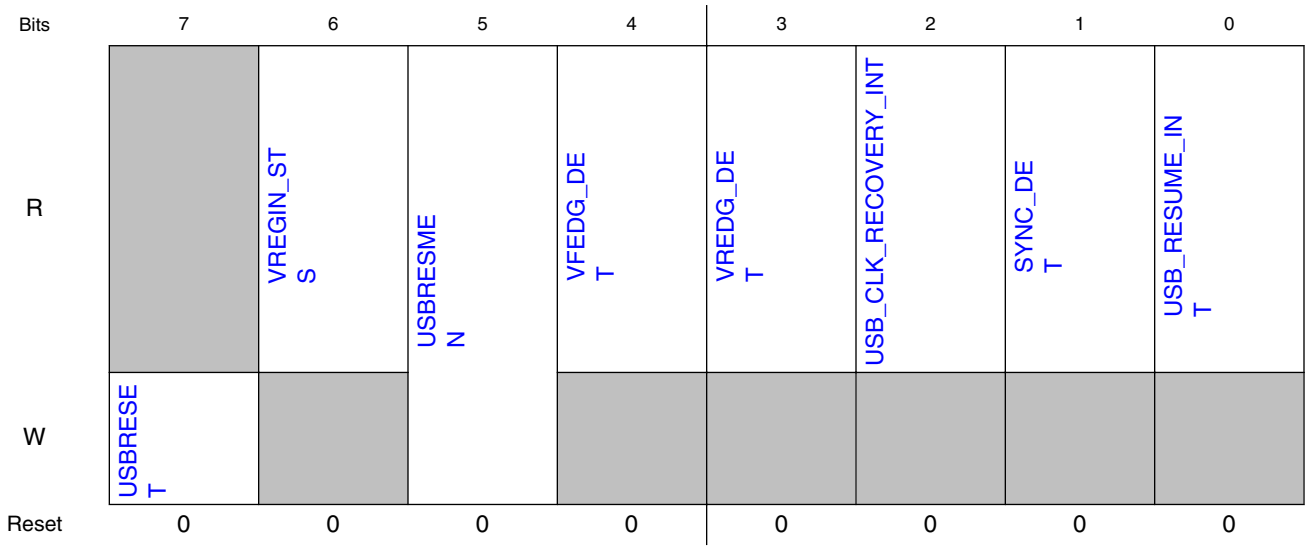
| Register | Offset |
|---|---|
| USBTRC0 | 10Ch |

#### 61.5.2.23.2 Function

Includes signals for basic operation of the on-chip USB Full Speed transceiver and for configuration of the USB data connection (that are not otherwise included in the USB Full Speed controller registers).

In bus-powered device use cases when the USB receptacle VBUS pin is connected to VREGIN, VREGIN interrupt detection using the VFEDG_DET and VREDG_DET bits may be used for VBUS detection.

## 61.5.2.23.3 Diagram




## 61.5.2.23.4 Fields

| Field | Function |
|---|---|
| 7<br>USBRESET | USB Reset<br>Generates a hard reset to USB. After USBRESET bit is set (to 1) and the reset occurs, the USBRESET bit is automatically cleared.<br>**NOTE: USBRESET bit always reads 0. After setting USBRESET bit, wait 2 USB clock cycles before accessing other USB register bit fields.**<br>0b - Normal USB module operation.<br>1b - Returns the USB module to its reset state. |
| 6<br>VREGIN_STS | VREGIN_STS<br>Indicates the VREGIN status, and is a read-only bit. |
| 5<br>USBRESMEN | Asynchronous Resume Interrupt Enable<br>When USBRESMEN bit is set to 1, after detection of resume signaling on the USB bus, it allows the USB module to send an asynchronous wakeup event to the MCU. The MCU then re-enables clocks to the USB module. This is used for low-power Suspend mode when USB module clocks are stopped or when the USB transceiver is in Suspend mode. Async wakeup only works in Device mode.<br>• The asynchronous resume interrupt differs from the synchronous resume interrupt, in that it asynchronously detects K-state using the unfiltered state of the D+ and D– pins.<br>• The asynchronous resume interrupt should only be enabled when the Transceiver is suspended.<br>0b - USB asynchronous wakeup from Suspend mode is disabled.<br>1b - USB asynchronous wakeup from Suspend mode is enabled. |
| 4<br>VFEDG_DET | VREGIN Falling Edge Interrupt Detect<br>To enable VFEDG_DET bit, use USB*x*_MISCCTRL[VFEDG_EN].<br>0b - VREGIN falling edge interrupt has not been detected.<br>1b - VREGIN falling edge interrupt has been detected. |
| 3<br>VREDG_DET | VREGIN Rising Edge Interrupt Detect<br>To enable VREDG_DET bit, use USB*x*_MISCCTRL[VREDG_EN]. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - VREGIN rising edge interrupt has not been detected.<br>1b - VREGIN rising edge interrupt has been detected. |
| 2<br>USB_CLK_RECOVERY_INT | Combined USB Clock Recovery interrupt status<br>The read-only USB_CLK_RECOVERY_INT field will be set to 1'b1 when any of USB clock recovery interrupt conditions are detected and those interrupts are unmasked.<br>• The only unmasked USB clock recovery interrupt condition results from an overflow of the frequency trim setting values, indicating that the frequency trim calculated is out of the adjustment range of the IRC48M output clock.<br>• To clear USB_CLK_RECOVERY_INT bit after it has been set, write 0xFF to register USB_CLK_RECOVER_INT_STATUS. |
| 1<br>SYNC_DET | Synchronous USB Interrupt Detect<br>0b - Synchronous interrupt has not been detected.<br>1b - Synchronous interrupt has been detected. |
| 0<br>USB_RESUME_INT | USB Asynchronous Interrupt<br>0b - No interrupt was generated.<br>1b - Interrupt was generated because of the USB asynchronous interrupt. |

## 61.5.2.24 Keep Alive mode control (KEEP_ALIVE_CTRL)

### 61.5.2.24.1 Offset

| Register | Offset |
|---|---|
| KEEP_ALIVE_CTRL | 124h |

### 61.5.2.24.2 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WAKE_INT_STS | KEEP_ALIVE_STS | 0 | WAKE_INT_EN | WAKE_REQ_EN | STOP_ACK_DLY_EN | OWN_OVERRD_EN | KEEP_ALIVE_EN |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

### 61.5.2.24.3 Fields

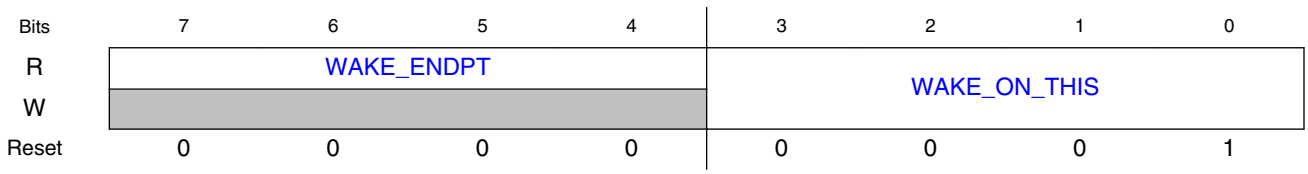
| Field | Function |
|---|---|
| 7<br>WAKE_INT_STS | Wakeup Interrupt Status<br>If KEEP_ALIVE mode is enabled and a SETUP token or software assigned token is received, then WAKE_INT_STS interrupt is set to 1 only after the token is processed and the USB state is IDLE. To clear WAKE_INT_STS bit, write 1. |
| 6<br>KEEP_ALIVE_STS | Keep Alive Status<br>Indicates if USB is in Keep Alive mode.<br>0b - USB is not in Keep Alive mode.<br>1b - USB is in Keep Alive mode. |
| 5<br>— | Reserved |
| 4<br>WAKE_INT_EN | Wakeup Interrupt Enable<br>To enable WAKE_INT_STS, set WAKE_INT_EN to 1. |
| 3<br>WAKE_REQ_EN | WAKE_REQ_EN<br>During KEEP_ALIVE mode, a bus access by the USB controller to a memory location outside the USB SRAM will cause the bus access to stall until KEEP_ALIVE mode is exited. When WAKE_REQ_EN is set to 1, a bus access by the USB controller to a memory location outside the USB SRAM will generate a wakeup, thereby allowing the bus access to complete. After the bus access finishes, the USB will reenter the KEEP_ALIVE mode if no other wakeups have been detected.<br>**NOTE:** An interrupt is not generated as a result of the wakeup request.<br>0b - USB bus wakeup request is disabled<br>1b - USB bus wakeup request is enabled |
| 2<br>STOP_ACK_DLY_EN | STOP_ACK_DLY_EN<br>• When STOP_ACK_DLY_EN is set to 1, then the module will respond to the ipg_stop *when USB core state is idle and there is no AHB transfer*.<br>• When STOP_ACK_DLY_EN is set to 0, then the module will respond to the ipg_stop *when there is no AHB transfer*.<br>0b - Enter KEEP_ALIVE mode until the USB core is idle and there is no USB AHB transfer.<br>1b - Enter KEEP_ALIVE mode immediately when there is no USB AHB transfer. |
| 1<br>OWN_OVERRD_EN | OWN bit override enable<br>During KEEP_ALIVE mode, when OWN_OVERRD_EN is set to 1, and if received token is not SETUP, then the OWN bit of current BD will be forced to 0, so the USB core will respond with NAK. |
| 0<br>KEEP_ALIVE_EN | Keep Alive mode enable<br>Global enable for USB_KEEP_ALIVE mode.<br>• When KEEP_ALIVE_EN is 1 (asserted), USB shall enter USB_KEEP_ALIVE mode after asserting ipg_stop;<br>• When KEEP_ALIVE_EN is 0 (de-asserted), everything is the same as before. |

## 61.5.2.25 Keep Alive mode wakeup control (KEEP_ALIVE_WKCTRL)

### 61.5.2.25.1 Offset

| Register | Offset |
|---|---|
| KEEP_ALIVE_WKCTRL | 128h |

### 61.5.2.25.2 Diagram




### 61.5.2.25.3 Fields

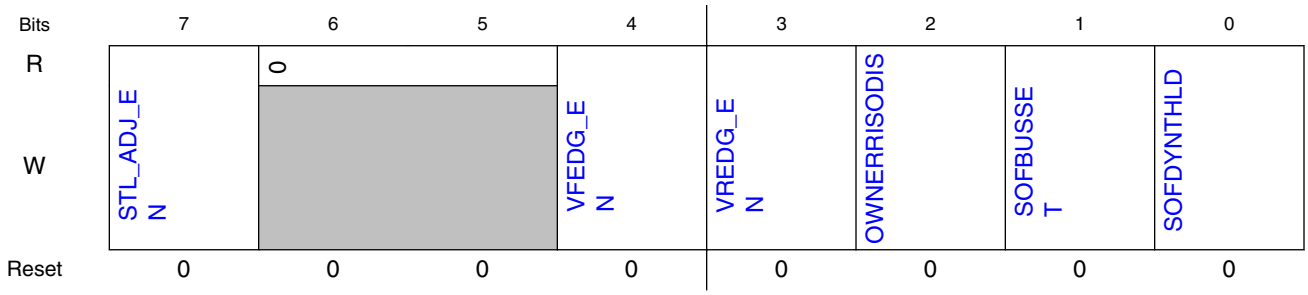
| Field | Function |
|---|---|
| 7-4<br>WAKE_ENDPT | WAKE_ENDPT<br>Indicates which endpoint causes the wakeup interrupt. Resets to 0. |
| 3-0<br>WAKE_ON_THIS | WAKE_ON_THIS<br>Software configures WAKE_ON_THIS bit to select which token can wakeup USB during KEEP_ALIVE mode. Resets to 4'b0001.<br>0001b - Wake up after receiving OUT/SETUP token packet.<br>1101b - Wake up after receiving SETUP token packet. All other values are reserved. |

## 61.5.2.26 Miscellaneous Control register (MISCCTRL)

### 61.5.2.26.1 Offset

| Register | Offset |
|---|---|
| MISCCTRL | 12Ch |

## 61.5.2.26.2 Diagram




## 61.5.2.26.3 Fields

| Field | Function |
|---|---|
| 7<br>STL_ADJ_EN | USB Peripheral mode Stall Adjust Enable<br>This field is valid only in Peripheral mode (CTL[HOSTMODEEN] = 0). By default (STL_ADJ_EN = 0), when an endpoint is stalled (ENDPTn[END_STALL] = 1), both IN and OUT directions of the endpoint are stalled. If STL_ADJ_EN = 1, then when an endpoint is stalled (ENDPTn[END_STALL] = 1), then the USBx_STALL_xx_DIS registers can be used to control which endpoint direction(s) are affected.<br>0b - If USB_ENDPTn[END_STALL] = 1, both IN and OUT directions for the associated endpoint will be stalled<br>1b - If USB_ENDPTn[END_STALL] = 1, the USB_STALL_xx_DIS registers control which directions for the associated endpoint will be stalled. |
| 6-5<br>— | Reserved |
| 4<br>VFEDG_EN | VREGIN Falling Edge Interrupt Enable<br>0b - VREGIN falling edge interrupt disabled.<br>1b - VREGIN falling edge interrupt enabled. |
| 3<br>VREDG_EN | VREGIN Rising Edge Interrupt Enable<br>0b - VREGIN rising edge interrupt disabled.<br>1b - VREGIN rising edge interrupt enabled. |
| 2<br>OWNERRISODIS | OWN Error Detect for ISO IN / ISO OUT Disable<br>OWNERRISODIS bit is only valid for Peripheral mode (CTL[HOSTMODEEN] = 0).<br>0b - OWN error detect for ISO IN / ISO OUT is not disabled.<br>1b - OWN error detect for ISO IN / ISO OUT is disabled. |
| 1<br>SOFBUSSET | SOF_TOK Interrupt Generation Mode Select<br>SOFBUSSET bit is only valid for Host mode (CTL[HOSTMODEEN] = 1).<br>0b - SOF_TOK interrupt is set according to SOF threshold value.<br>1b - SOF_TOK interrupt is set when SOF counter reaches 0. |
| 0<br>SOFDYNTHLD | Dynamic SOF Threshold Compare mode<br>This field is only valid for Host mode (CTL[HOSTMODEEN] = 1).<br>0b - SOF_TOK interrupt is set when byte times SOF threshold is reached.<br>1b - SOF_TOK interrupt is set when 8 byte times SOF threshold is reached or overstepped. |

## 61.5.2.27 Peripheral mode stall disable for endpoints 7 to 0 in IN direction (STALL_IL_DIS)

### 61.5.2.27.1 Offset

| Register | Offset |
|---|---|
| STALL_IL_DIS | 130h |

### 61.5.2.27.2 Function

This register is valid only in Peripheral mode(CTL[HOSTMODEEN] = 0) and when the stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN] = 1).

When an endpoint is stalled (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the IN direction of the endpoints 7 to 0.

### 61.5.2.27.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | STALL_I_DIS7 | STALL_I_DIS6 | STALL_I_DIS5 | STALL_I_DIS4 | STALL_I_DIS3 | STALL_I_DIS2 | STALL_I_DIS1 | STALL_I_DIS0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.27.4 Fields

| Field | Function |
|---|---|
| 7<br>STALL_I_DIS7 | STALL_I_DIS7<br>Disable endpoint 7 IN direction.<br>0b - Endpoint 7 IN direction stall is enabled.<br>1b - Endpoint 7 IN direction stall is disabled. |
| 6<br>STALL_I_DIS6 | STALL_I_DIS6<br>Disable endpoint 6 IN direction.<br>0b - Endpoint 6 IN direction stall is enabled.<br>1b - Endpoint 6 IN direction stall is disabled. |
| 5<br>STALL_I_DIS5 | STALL_I_DIS5<br>Disable endpoint 5 IN direction.<br>0b - Endpoint 5 IN direction stall is enabled.<br>1b - Endpoint 5 IN direction stall is disabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 4<br>STALL_I_DIS4 | STALL_I_DIS4<br>Disable endpoint 4 IN direction.<br>0b - Endpoint 4 IN direction stall is enabled.<br>1b - Endpoint 4 IN direction stall is disabled. |
| 3<br>STALL_I_DIS3 | STALL_I_DIS3<br>Disable endpoint 3 IN direction.<br>0b - Endpoint 3 IN direction stall is enabled.<br>1b - Endpoint 3 IN direction stall is disabled. |
| 2<br>STALL_I_DIS2 | STALL_I_DIS2<br>Disable endpoint 2 IN direction.<br>0b - Endpoint 2 IN direction stall is enabled.<br>1b - Endpoint 2 IN direction stall is disabled. |
| 1<br>STALL_I_DIS1 | STALL_I_DIS1<br>Disable endpoint 1 IN direction.<br>0b - Endpoint 1 IN direction stall is enabled.<br>1b - Endpoint 1 IN direction stall is disabled. |
| 0<br>STALL_I_DIS0 | STALL_I_DIS0<br>Disable endpoint 0 IN direction.<br>0b - Endpoint 0 IN direction stall is enabled.<br>1b - Endpoint 0 IN direction stall is disabled. |

## 61.5.2.28 Peripheral mode stall disable for endpoints 15 to 8 in IN direction (STALL_IH_DIS)

### 61.5.2.28.1 Offset

| Register | Offset |
|---|---|
| STALL_IH_DIS | 134h |

### 61.5.2.28.2 Function

This register is valid only in Peripheral mode(CTL[HOSTMODEEN] = 0) and when the stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN] = 1).

When an endpoint is stalled (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the IN direction of the endpoints 15 to 8.

## 61.5.2.28.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | STALL_I_DIS15 | STALL_I_DIS14 | STALL_I_DIS13 | STALL_I_DIS12 | STALL_I_DIS11 | STALL_I_DIS10 | STALL_I_DIS9 | STALL_I_DIS8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 61.5.2.28.4 Fields

| Field | Function |
|---|---|
| 7<br>STALL_I_DIS15 | STALL_I_DIS15<br>Disable endpoint 15 IN direction.<br>0b - Endpoint 15 IN direction stall is enabled.<br>1b - Endpoint 15 IN direction stall is disabled. |
| 6<br>STALL_I_DIS14 | STALL_I_DIS14<br>Disable endpoint 14 IN direction.<br>0b - Endpoint 14 IN direction stall is enabled.<br>1b - Endpoint 14 IN direction stall is disabled. |
| 5<br>STALL_I_DIS13 | STALL_I_DIS13<br>Disable endpoint 13 IN direction.<br>0b - Endpoint 13 IN direction stall is enabled.<br>1b - Endpoint 13 IN direction stall is disabled. |
| 4<br>STALL_I_DIS12 | STALL_I_DIS12<br>Disable endpoint 12 IN direction.<br>0b - Endpoint 12 IN direction stall is enabled.<br>1b - Endpoint 12 IN direction stall is disabled. |
| 3<br>STALL_I_DIS11 | STALL_I_DIS11<br>Disable endpoint 11 IN direction.<br>0b - Endpoint 11 IN direction stall is enabled.<br>1b - Endpoint 11 IN direction stall is disabled. |
| 2<br>STALL_I_DIS10 | STALL_I_DIS10<br>Disable endpoint 10 IN direction.<br>0b - Endpoint 10 IN direction stall is enabled.<br>1b - Endpoint 10 IN direction stall is disabled. |
| 1<br>STALL_I_DIS9 | STALL_I_DIS9<br>Disable endpoint 9 IN direction.<br>0b - Endpoint 9 IN direction stall is enabled.<br>1b - Endpoint 9 IN direction stall is disabled. |
| 0<br>STALL_I_DIS8 | STALL_I_DIS8<br>Disable endpoint 8 IN direction.<br>0b - Endpoint 8 IN direction stall is enabled.<br>1b - Endpoint 8 IN direction stall is disabled. |

## 61.5.2.29 Peripheral mode stall disable for endpoints 7 to 0 in OUT direction (STALL_OL_DIS)

### 61.5.2.29.1 Offset

| Register | Offset |
|---|---|
| STALL_OL_DIS | 138h |

### 61.5.2.29.2 Function

This register is valid only in Peripheral mode(CTL[HOSTMODEEN] = 0) and when the stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN] = 1).

When an endpoint is stalled (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 7 to 0.

### 61.5.2.29.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | STALL_O_DIS7 | STALL_O_DIS6 | STALL_O_DIS5 | STALL_O_DIS4 | STALL_O_DIS3 | STALL_O_DIS2 | STALL_O_DIS1 | STALL_O_DIS0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.29.4 Fields

| Field | Function |
|---|---|
| 7<br>STALL_O_DIS7 | STALL_O_DIS7<br>Disable endpoint 7 OUT direction.<br>0b - Endpoint 7 OUT direction stall is enabled.<br>1b - Endpoint 7 OUT direction stall is disabled. |
| 6<br>STALL_O_DIS6 | STALL_O_DIS6<br>Disable endpoint 6 OUT direction.<br>0b - Endpoint 6 OUT direction stall is enabled.<br>1b - Endpoint 6 OUT direction stall is disabled. |
| 5 | STALL_O_DIS5 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| STALL_O_DIS5 | Disable endpoint 5 OUT direction.<br>0b - Endpoint 5 OUT direction stall is enabled.<br>1b - Endpoint 5 OUT direction stall is disabled. |
| 4<br>STALL_O_DIS4 | STALL_O_DIS4<br>Disable endpoint 4 OUT direction.<br>0b - Endpoint 4 OUT direction stall is enabled.<br>1b - Endpoint 4 OUT direction stall is disabled. |
| 3<br>STALL_O_DIS3 | STALL_O_DIS3<br>Disable endpoint 3 OUT direction.<br>0b - Endpoint 3 OUT direction stall is enabled.<br>1b - Endpoint 3 OUT direction stall is disabled. |
| 2<br>STALL_O_DIS2 | STALL_O_DIS2<br>Disable endpoint 2 OUT direction.<br>0b - Endpoint 2 OUT direction stall is enabled.<br>1b - Endpoint 2 OUT direction stall is disabled. |
| 1<br>STALL_O_DIS1 | STALL_O_DIS1<br>Disable endpoint 1 OUT direction.<br>0b - Endpoint 1 OUT direction stall is enabled.<br>1b - Endpoint 1 OUT direction stall is disabled. |
| 0<br>STALL_O_DIS0 | STALL_O_DIS0<br>Disable endpoint 0 OUT direction.<br>0b - Endpoint 0 OUT direction stall is enabled.<br>1b - Endpoint 0 OUT direction stall is disabled. |

## 61.5.2.30 Peripheral mode stall disable for endpoints 15 to 8 in OUT direction (STALL_OH_DIS)

### 61.5.2.30.1 Offset

| Register | Offset |
|---|---|
| STALL_OH_DIS | 13Ch |

### 61.5.2.30.2 Function

This register is valid only in Peripheral mode(CTL[HOSTMODEEN] = 0) and when stall adjust enable bit is set (MISCCTRL[STL_ADJ_EN] = 1).

When an endpoint is stalled (ENDPTn[END_STALL] = 1), the fields in this register enable or disable stalling of the OUT direction for the endpoints 15 to 8.

## 61.5.2.30.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | STALL_O_DIS15 | STALL_O_DIS14 | STALL_O_DIS13 | STALL_O_DIS12 | STALL_O_DIS11 | STALL_O_DIS10 | STALL_O_DIS9 | STALL_O_DIS8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 61.5.2.30.4 Fields

| Field | Function |
|---|---|
| 7<br>STALL_O_DIS15 | STALL_O_DIS15<br>Disable endpoint 15 OUT direction.<br>0b - Endpoint 15 OUT direction stall is enabled.<br>1b - Endpoint 15 OUT direction stall is disabled. |
| 6<br>STALL_O_DIS14 | STALL_O_DIS14<br>Disable endpoint 14 OUT direction.<br>0b - Endpoint 14 OUT direction stall is enabled.<br>1b - Endpoint 14 OUT direction stall is disabled. |
| 5<br>STALL_O_DIS13 | STALL_O_DIS13<br>Disable endpoint 13 OUT direction.<br>0b - Endpoint 13 OUT direction stall is enabled.<br>1b - Endpoint 13 OUT direction stall is disabled. |
| 4<br>STALL_O_DIS12 | STALL_O_DIS12<br>Disable endpoint 12 OUT direction.<br>0b - Endpoint 12 OUT direction stall is enabled.<br>1b - Endpoint 12 OUT direction stall is disabled. |
| 3<br>STALL_O_DIS11 | STALL_O_DIS11<br>Disable endpoint 11 OUT direction.<br>0b - Endpoint 11 OUT direction stall is enabled.<br>1b - Endpoint 11 OUT direction stall is disabled. |
| 2<br>STALL_O_DIS10 | STALL_O_DIS10<br>Disable endpoint 10 OUT direction.<br>0b - Endpoint 10 OUT direction stall is enabled.<br>1b - Endpoint 10 OUT direction stall is disabled. |
| 1<br>STALL_O_DIS9 | STALL_O_DIS9<br>Disable endpoint 9 OUT direction.<br>0b - Endpoint 9 OUT direction stall is enabled.<br>1b - Endpoint 9 OUT direction stall is disabled. |
| 0<br>STALL_O_DIS8 | STALL_O_DIS8<br>Disable endpoint 8 OUT direction.<br>0b - Endpoint 8 OUT direction stall is enabled.<br>1b - Endpoint 8 OUT direction stall is disabled. |

## 61.5.2.31 USB Clock recovery control (CLK_RECOVER_CTRL)

### 61.5.2.31.1 Offset

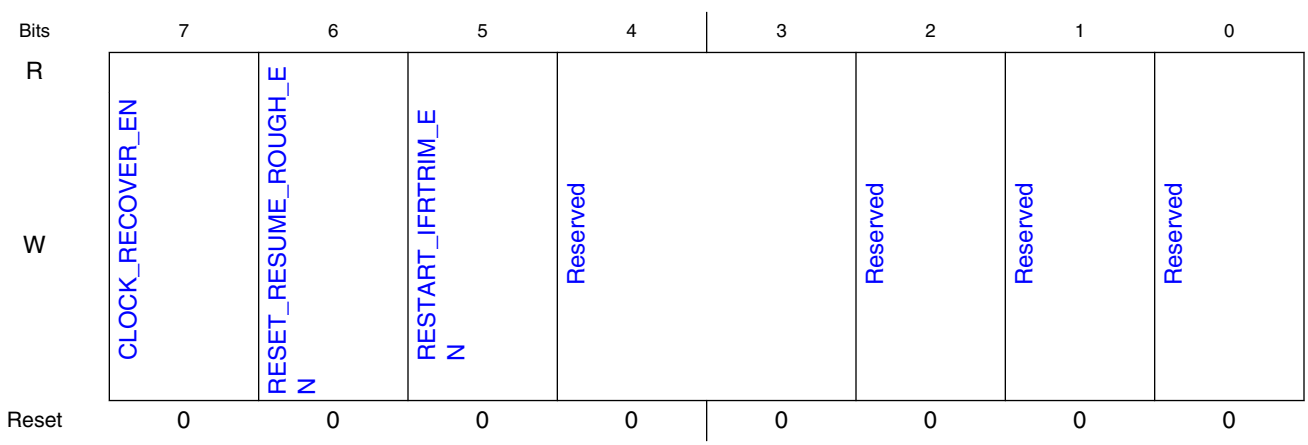
| Register | Offset |
|---|---|
| CLK_RECOVER_CTRL | 140h |

### 61.5.2.31.2 Function

Signals in CLK_RECOVER_CTRL register control the crystal-less USB clock mode in which the internal IRC48M oscillator is tuned to match the clock extracted from the incoming USB data stream.

**The IRC48Minternal oscillator module must be enabled in register USB_CLK_RECOVER_IRC_EN for this mode.**

### 61.5.2.31.3 Diagram




### 61.5.2.31.4 Fields

| Field | Function |
|---|---|
| 7<br>CLOCK_RECOVER_EN | Crystal-less USB enable<br>CLOCK_RECOVER_EN bit must be enabled if user wants to use the crystal-less USB mode for the Full Speed USB controller and transceiver.<br>**NOTE: CLOCK_RECOVER_EN bit should not be set for USB Host mode or OTG.**<br>0b - Disable clock recovery block (default) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Enable clock recovery block |
| 6<br>RESET_RESUME_ROUGH_EN | Reset/resume to rough phase enable<br>The clock recovery block tracks the IRC48M to get an accurate 48 MHz clock. It has 2 phases after user enables clock_recover_en bit, rough phase and tracking phase. The step to fine tune the IRC48M by adjusting the trim fine value is different during these 2 phases. The step-in rough phase is larger than that in tracking phase. Switch back to the rough stage whenever a USB bus reset or bus resume occurs.<br>0b - Always works in tracking phase after the first time rough phase, to track transition (default).<br>1b - Go back to rough stage whenever a bus reset or bus resume occurs. |
| 5<br>RESTART_IFRTRIM_EN | Restart from IFR trim value<br>IRC48M has a default trim fine value whose default value is factory trimmed (the IFR trim value). The clock recover block tracks the accuracy of the clock 48 MHz, and keeps updating the trim fine value accordingly.<br>0b - Trim fine adjustment always works based on the previous updated trim fine value (default).<br>1b - Trim fine restarts from the IFR trim value, whenever bus_reset/bus_resume is detected or module enable is desasserted. |
| 4-3<br>— | Reserved |
| 2<br>— | Reserved<br>Customers should not change this bit from its default state. |
| 1<br>— | Reserved<br>Customers should not change this bit from its default state. |
| 0<br>— | Reserved<br>Customers should not change this bit from its default state. |

## 61.5.2.32 IRC48MFIRC oscillator enable register (CLK_RECOVER_IRC_EN)

### 61.5.2.32.1 Offset

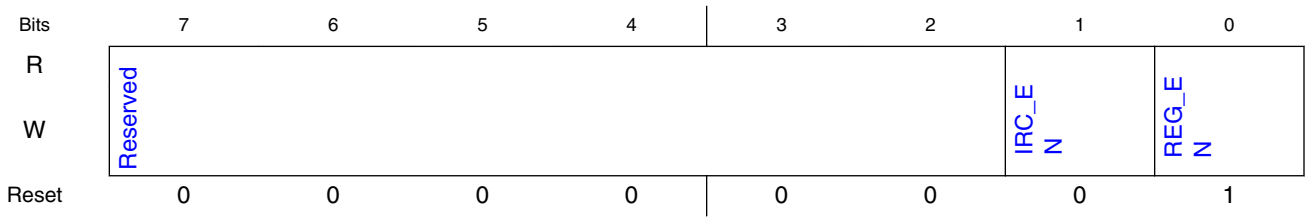
| Register | Offset |
|---|---|
| CLK_RECOVER_IRC_EN | 144h |

### 61.5.2.32.2 Function

Controls basic operation of the on-chip IRC48M module used to produce nominal 48 MHz clocks for USB crystal-less operation and other functions.

See additional information about the IRC48M operation in the Clock Distribution chapter.

### 61.5.2.32.3 Diagram




### 61.5.2.32.4 Fields

| Field | Function |
|---|---|
| 7-2<br>— | Reserved |
| 1<br>IRC_EN | IRC_EN<br>IRC48M enable bit is used to enable the on-chip IRC48M module to generate clocks for crystal-less USB, and it can be used for FS USB Device mode operation. IRC_EN bit must be set before using the crystal-less USB clock configuration.<br>0b - Disable the IRC48M module (default)<br>1b - Enable the IRC48M module |
| 0<br>REG_EN | Regulator enable<br>IRC48M regulator enable bit is used to enable the local analog regulator for IRC48M module. REG_EN bit must be set if user wants to use the crystal-less USB clock configuration.<br>0b - IRC48M local regulator is disabled<br>1b - IRC48M local regulator is enabled (default) |

## 61.5.2.33 Clock recovery combined interrupt enable (CLK_RECOVER_INT_EN)

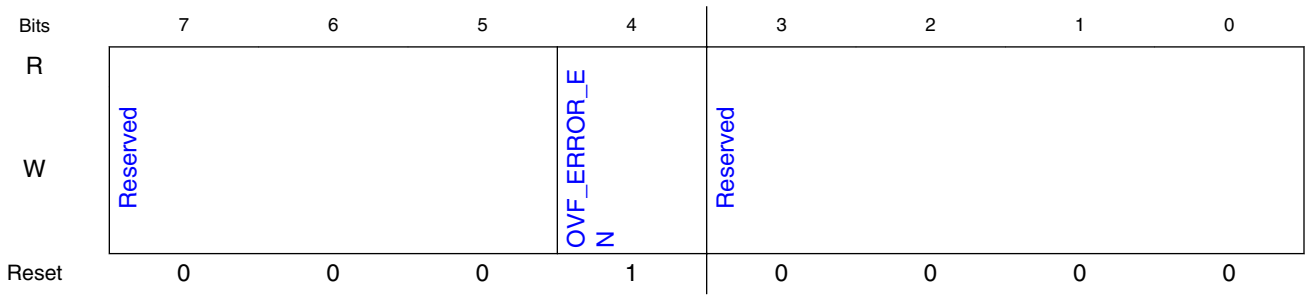
### 61.5.2.33.1 Offset

| Register | Offset |
|---|---|
| CLK_RECOVER_INT_EN | 154h |

### 61.5.2.33.2 Function

Enables or masks the individual interrupt flags which are logically OR'ed together, to produce the combined interrupt indication at USB_USBTRC0[USB_CLK_RECOVERY_INT] bit, if the indicated conditions have been detected in the USB clock recovery algorithm operation.

### 61.5.2.33.3 Diagram




### 61.5.2.33.4 Fields

| Field | Function |
|---|---|
| 7-5<br>— | Reserved<br>Should always be written as 0. |
| 4<br>OVF_ERROR_EN | OVF_ERROR_EN<br>Determines whether the OVF_ERROR condition signal is used in generation of USB_CLK_RECOVERY_INT.<br>0b - The interrupt will be masked<br>1b - The interrupt will be enabled (default) |
| 3-0<br>— | Reserved<br>Should always be written as 0. |

## 61.5.2.34 Clock recovery separated interrupt status (CLK_RECOVER_INT_STATUS)

### 61.5.2.34.1 Offset

| Register | Offset |
|---|---|
| CLK_RECOVER_INT_STATUS | 15Ch |

### 61.5.2.34.2 Function

A Write operation with a 1'b1 on any combination of individual bits will clear those bits.

### 61.5.2.34.3 Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | OVF_ERRO R | Reserved | | | |
| W | W1C | | | W1C | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 61.5.2.34.4 Fields

| Field | Function |
|---|---|
| 7-5<br>— | Reserved<br>Should always be written as 0. |
| 4<br>OVF_ERROR | OVF_ERROR<br>Indicates that the USB clock recovery algorithm has detected that the frequency trim adjustment needed for the IRC48M output clock is outside the available TRIM_FINE adjustment range for the IRC48M module.<br>0b - No interrupt is reported<br>1b - Unmasked interrupt has been generated |
| 3-0<br>— | Reserved<br>Should always be written as 0. |

## 61.6 Keep Alive mode

Keep Alive mode is for chips with *device-only* USB implementations.

**NOTE**

The address range 0x4010_0000 to 0x4010_07FF consists of 2 KB of on-chip USB RAM in Run mode. Due to a design limitation, only the bottom half of the USB RAM (0x4010_0000 to 0x4010_03FF) can be accessed by the USB core in Keep Alive Mode.

When entering Keep Alive Mode, all USB buffers (and BDT) *that will be accessed during Keep Alive mode* will be placed in the bottom half of USB RAM.

Keep Alive mode keeps USB active in STOP/VLPS low-power modes so there is no need to re-start USB after exiting low power modes. When connected to the host and when there is no data to transfer, to save power, the software can power down the system bus and the CPU, keeping only the USB power on. In Keep Alive mode, only the USB 48 MHz functional clock is required by the USB controller; the rest of the device enters Stop mode. When the SETUP token or a software-assigned token is received, the USB core wakes up and responds NAK to all other packets.

Because the USB core must access the BDT in Keep Alive mode, the bottom half of 2KB on-chip USB RAM (located from 0x4010_0000 to 0x4010_03FF) is used for the BDT and small data packets during Keep Alive mode. Software puts the BDT into this RAM in Keep Alive mode and changes all OWN bits to 0 (except the SETUP endpoint). This can be done by setting USB_KEEP_ALIVE_CTRL[OWN_OVERRD_EN].

**NOTE**

For more about entering/exiting Keep Alive mode and device mode IRC48M operations, see the chip-specific section in the front of this chapter.

# Chapter 62
# USB Voltage Regulator (USBVREG)

## 62.1 Chip-specific USBVREG information

### 62.1.1 USB Voltage Regulator Configuration

This section summarizes how the module has been configured in the chip.

**NOTE**

When USB is not used in the application, it is recommended to connect VOUT33 and VREGIN together and tie to ground through a 10kΩ resistor. Do not tie directly to ground, as this causes a latch-up risk.

## 62.2 Introduction

This section provides an introduction to the USB_VREG module.The USB voltage regulator module (USB_VREG) provides USB regulator control registers.

### 62.2.1 Features

The USB_VREG includes the following features:

- USB regulator control

# 62.3 Memory Map and Registers

## 62.3.1 USBVREG register descriptions

### 62.3.1.1 USBVREG Memory map

USBVREG base address: 4002_7000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | USB VREG Control Register (CTRL) | 32 | RW | 8000_0000h |
| 4h | USB VREG Configuration Control Register (CFGCTRL) | 32 | RW | 0000_0000h |

### 62.3.1.2 USB VREG Control Register (CTRL)

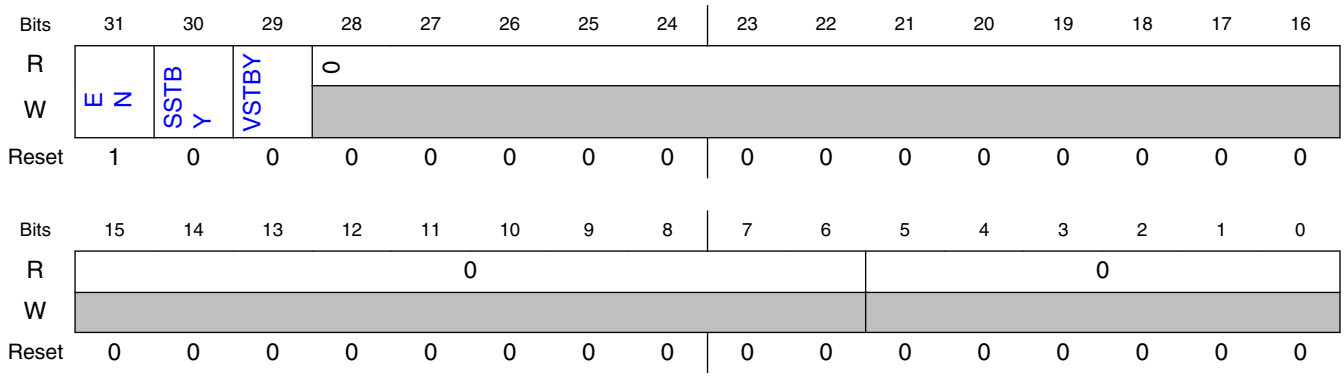
#### 62.3.1.2.1 Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

#### 62.3.1.2.2 Function

This register controls USB regulator.

### 62.3.1.2.3 Diagram




### 62.3.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>EN | USB Voltage Regulator Enable<br>This field controls whether the USB voltage regulator is enabled.<br>0b - USB voltage regulator is disabled.<br>1b - USB voltage regulator is enabled. |
| 30<br>SSTBY | USB Voltage Regulator in Standby Mode during Stop, VLPS, LLS and VLLS Modes<br>This field controls whether the USB voltage regulator is placed in standby mode during Stop,VLPS,LLS and VLLS modes.<br>0b - USB voltage regulator is not in standby during Stop,VLPS,LLS and VLLS modes.<br>1b - USB voltage regulator is in standby during Stop,VLPS,LLS and VLLS modes. |
| 29<br>VSTBY | USB Voltage Regulator in Standby Mode during VLPR and VLPW modes<br>This field controls whether the USB voltage regulator is placed in standby mode during VLPR and VLPW modes.<br>0b - USB voltage regulator is not in standby during VLPR and VLPW modes.<br>1b - USB voltage regulator in standby during VLPR and VLPW modes. |
| 28-6<br>— | Reserved |
| 5-0<br>— | Reserved |

## 62.3.1.3 USB VREG Configuration Control Register (CFGCTRL)

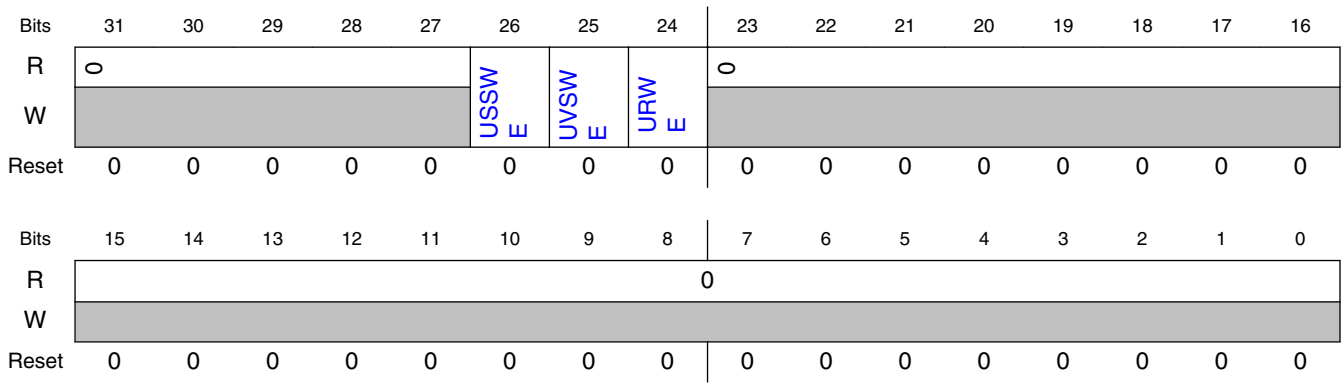
### 62.3.1.3.1 Offset

| Register | Offset |
|---|---|
| CFGCTRL | 4h |

### 62.3.1.3.2 Function

This register controls the USB regulator configurations.

### 62.3.1.3.3 Diagram




### 62.3.1.3.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>USSWE | USB Voltage Rregulator Stop Standby Write Enable<br>Writing 1 to this field allows the CTRL[SSTBY] field to be written. This register field clears after a write to CTRL[SSTBY].<br>0b - CTRL[SSTBY] field cannot be written.<br>1b - CTRL[SSTBY] can be written. |
| 25<br>UVSWE | USB Voltage Regulator VLP Standby Write Enable<br>Writing 1 to this field allows the CTRL[VSTBY] field to be written. This register field clears after a write to CTRL[VSTBY].<br>0b - CTRL[VSTBY] cannot be written.<br>1b - CTRL[VSTBY] can be written. |
| 24<br>URWE | USB Voltage Regulator Enable Write Enable<br>Writing 1 to this field allows the CTRL[EN] to be written. This register field clears after a write to CTRL[EN].<br>0b - CTRL[EN] can not be written.<br>1b - CTRL[EN] can be written. |
| 23-0<br>— | Reserved |

# Chapter 63
# Ultra Secured Digital Host Controller (uSDHC)

## 63.1 Overview

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 63-1.

The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

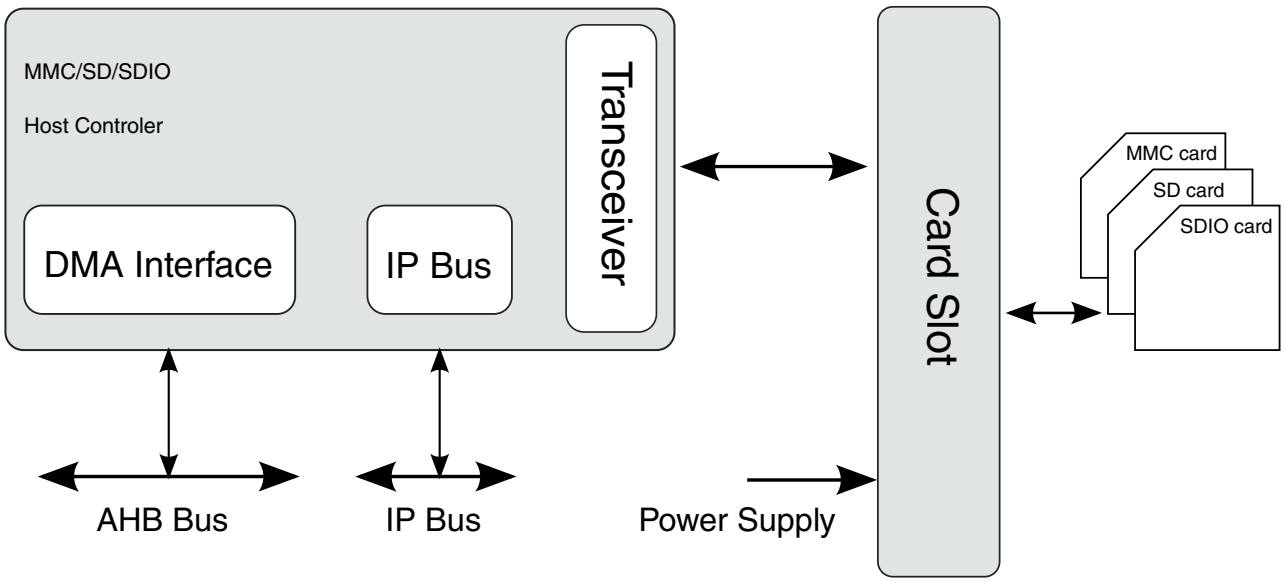



**Figure 63-1. System Connection of the uSDHC**

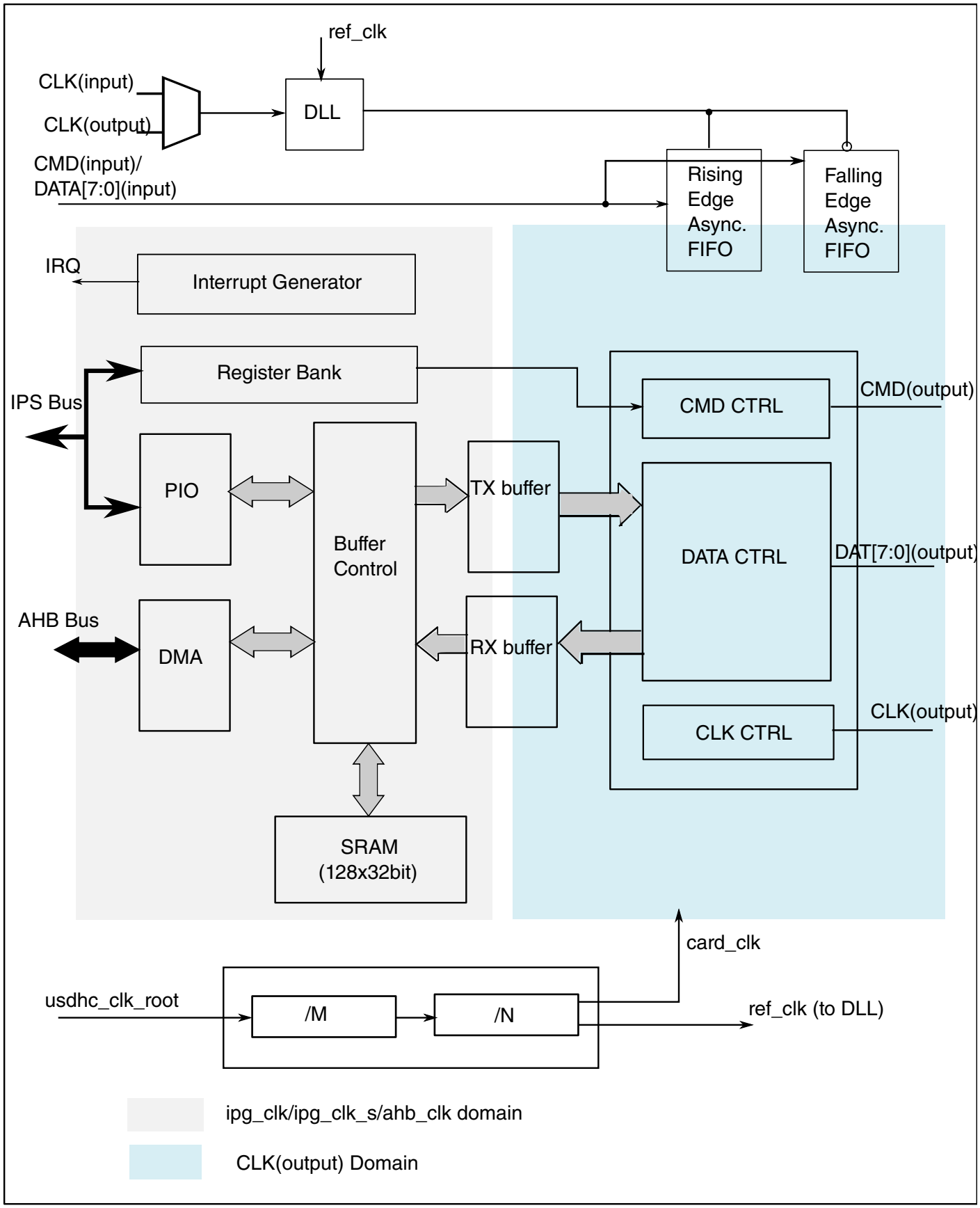



**Figure 63-2. ultra Secure Digital Host Controller Block Diagram**

### 63.1.1 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 2.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 52 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes
  - Up to 200 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR(Single Data Rate) mode
  - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR(Dual Data Rate) mode
  - Up to 200 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR(Single Data Rate) mode
  - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR(Dual Data Rate) mode
  - Up to 1600 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR(Single Data Rate) mode
  - Up to 832 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR(Dual Data Rate) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal DMA capabilities
- Support voltage selection by configuring vendor specific register bit
- Supports Advanced DMA to perform linked memory access

Note: IP can support above listed speed mode and max data throughput. For SOC, speed mode and data throughput should be Chip-specific.Pls check corresponding SOC description.

### 63.1.2 Modes and Operations

#### 63.1.2.1 Data transfer Modes

The uSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 26 MHz)
- MMC high speed mode (up to 52 MHz)
- MMC HS200 mode(up to 200 Mhz)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)

Note: IP can support above listed speed mode and max clock frequency. For SOC, speed mode and max clock frequency should be Chip-specific.Pls check corresponding SOC description.

## 63.2 External Signals

The following table describes the external signals of uSDHC:

**Table 63-1. uSDHC External Signals**

| Signal | Description | Direction |
|---|---|---|
| CLK | Clock for MMC/SD/SDIO card | O |
| CMD | CMD line connect to card | I/O |
| DATA7 | DAT7 line in 8-bit mode<br>Not used in other modes | I/O |
| DATA6 | DAT6 line in 8-bit mode | I/O |

*Table continues on the next page...*

**Table 63-1. uSDHC External Signals (continued)**

| Signal | Description | Direction |
|---|---|---|
| | Not used in other modes | |
| DATA5 | DAT5 line in 8-bit mode<br>Not used in other modes | I/O |
| DATA4 | DAT4 line in 8-bit mode<br>Not used in other modes | I/O |
| DATA3 | DAT3 line in 4/8-bit mode or configured as card detection pin<br>May be configured as card detection pin in 1-bit mode | I/O |
| DATA2 | DAT2 line or Read Wait in 4-bit mode<br>Read Wait in 1-bit mode | I/O |
| DATA1 | DAT1 line in 4/8-bit mode<br>Also used to detect interrupt in 1/4-bit mode | I/O |
| DATA0 | DAT0 line in all modes<br>Also used to detect busy state | I/O |
| CD_B | Card detection pin<br>If not used(for the embedded memory),tie low to indicate there is a card attached. | I |
| WP | Card write protect detect<br>If not used(for the embedded memory), tie low to indicate it's not write protected. | I |
| LCTL | LED control used to drive an external LED Active high<br>Fully controlled by the driver<br>Optional output | O |
| RESET_B | Card hardware reset signal, active LOW | O |
| VSELECT | IO power voltage selection signal | O |

## 63.2.1 Signals Overview

The uSDHC has 14 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC and the card.
- The CD and WP are card detection and write protection signals directly routed from the socket. These two signals are active low (0). A low on CD# means that a card is inserted, and a high on WP means that the write protect switch is active.
- LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.

- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier.

CD, WP, LCTL, RST and VSELECT are all optional for system implementation. If the uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.If the uSDHC does not support HS400 mode, STROBE can also be optional and tied to low.

## 63.3 Clocks

The table found here describes the clock sources for uSDHC.

**Table 63-2. uSDHC Clocks**

| Clock name | Clock Root | Description |
|---|---|---|
| hclk | ahb_clk_root | AHB bus clock |
| ipg_clk | ipg_clk_root | Peripheral clock |
| ipg_clk_perclk | usdhc_clk_root | Base clock |
| ipg_clk_s | ipg_clk_root | Peripheral access clock for register accesses |

## 63.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

### 63.4.1 Data Buffer

The uSDHC uses one configurable data buffer to transfer data between the system bus (IP Bus or AHB Bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).

The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable and can be from 1 to 128 words.The burst lengths for read and write are also configurable and can be from 1 to 31 words.

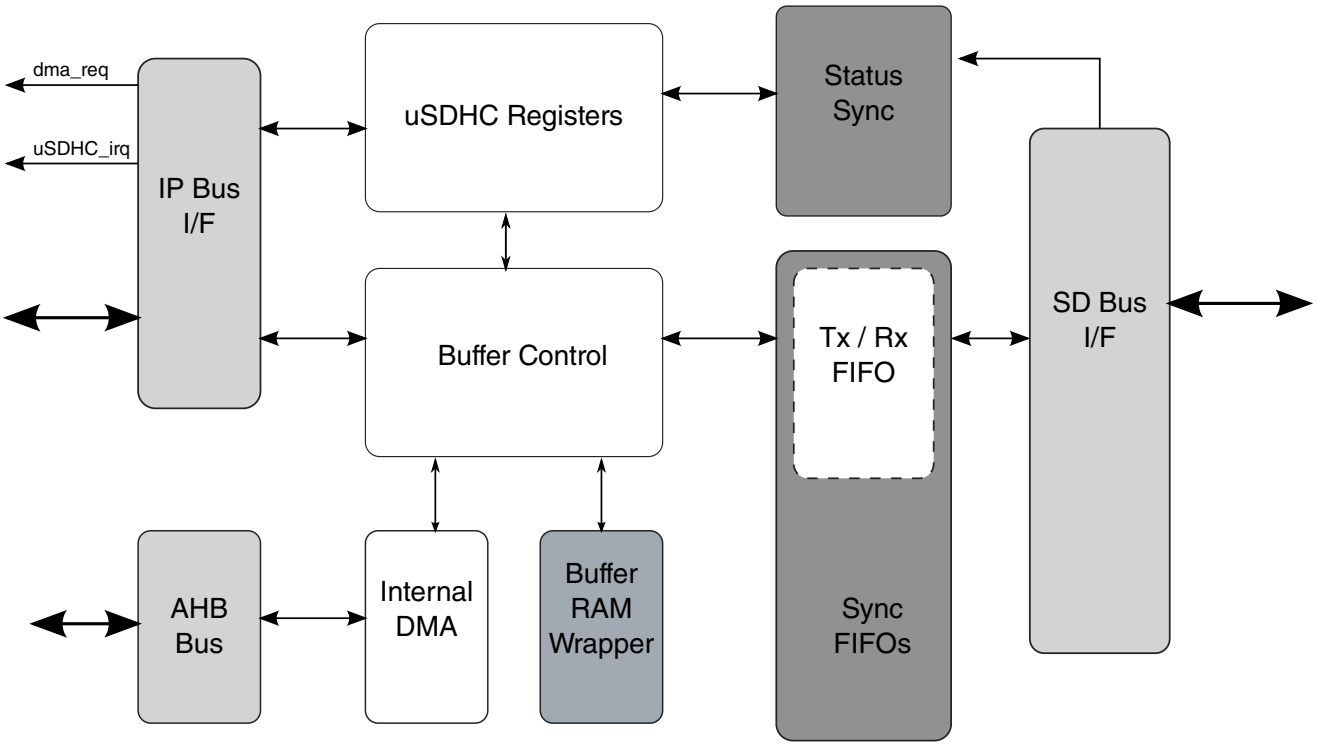



**Figure 63-3. uSDHC Buffer Scheme**

There are 2 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark Level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored in the buffer.

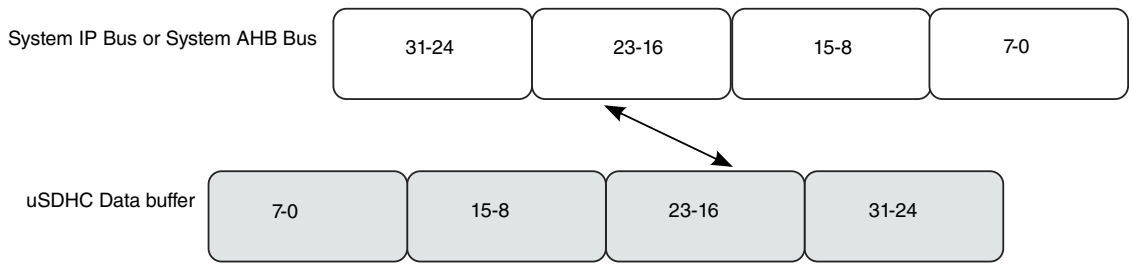



**Figure 63-4. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode**

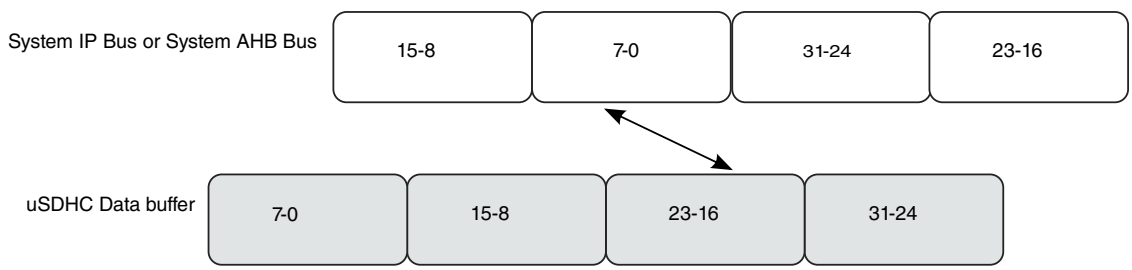



**Figure 63-5. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode**

### 63.4.1.1 Write Operation Sequence

There are 2 ways to write data into the buffer when the user transfers data to the card:

- Processor core polling through the BWR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, (the DMAEN bit in the Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the uSDHC sets the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

The uSDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the uSDHC will stop the CLK to avoid the data buffer under-run situation.

#### 63.4.1.2 Read Operation Sequence

There are 2 ways to read data from the buffer when the user transfers data to the card:

- Processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data. At the same time, the uSDHC sets the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

For any write transfer mode, the uSDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the uSDHC will stop the CLK to avoid the data buffer over-run situation.

#### 63.4.1.3 Data Buffer and Block Size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly.

For both read and write, the watermark level can be from 1 to 128 words. For both read and write the burst length can be from 1 to 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned) stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block the ending byte will be abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

#### 63.4.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the dividing of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size up to 64 bytes. Although the uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided (see example below).

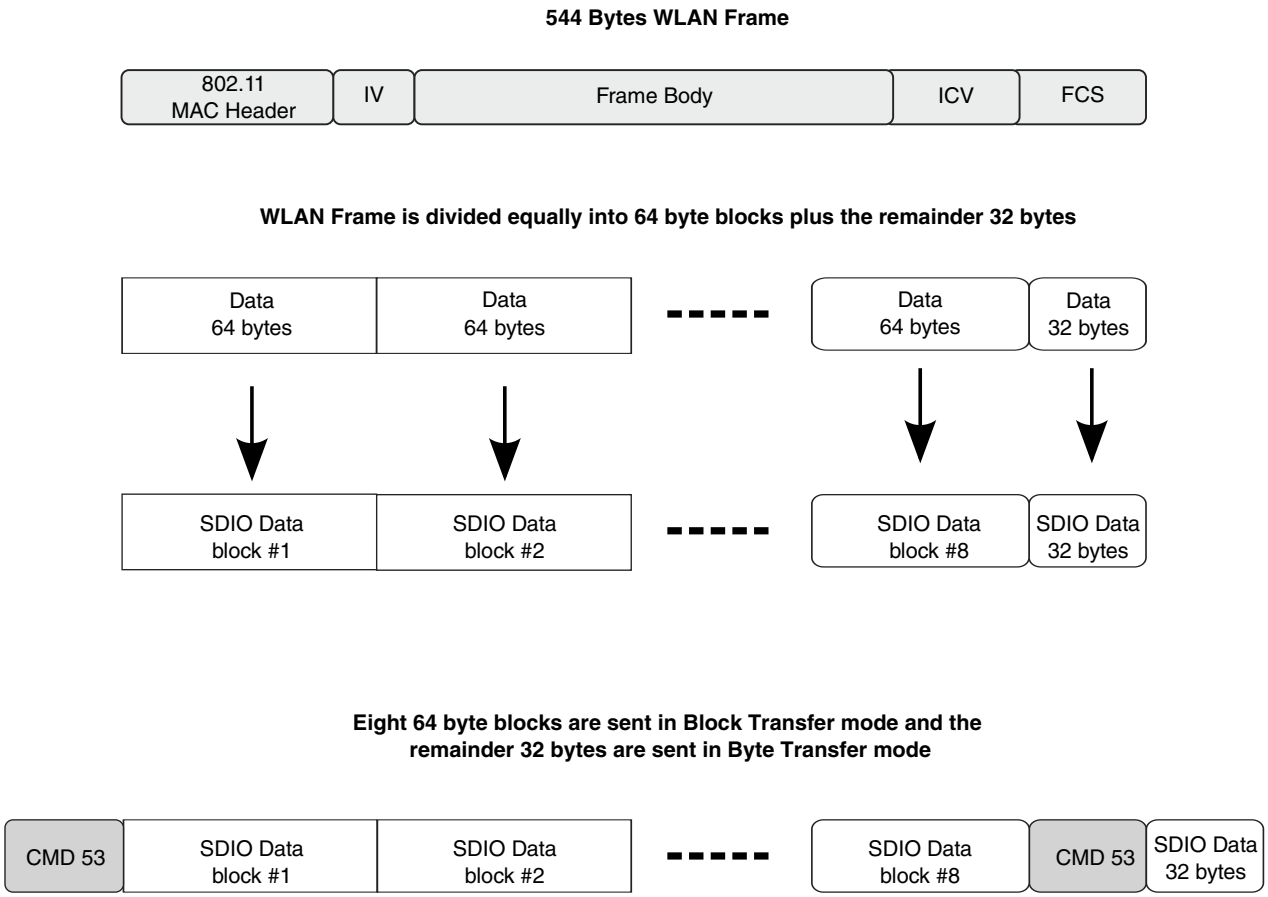



**Figure 63-6. Example for Dividing Large Data Transfers**

## 63.4.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register.

See the figure below for an illustration of the DMA AHB interface block.

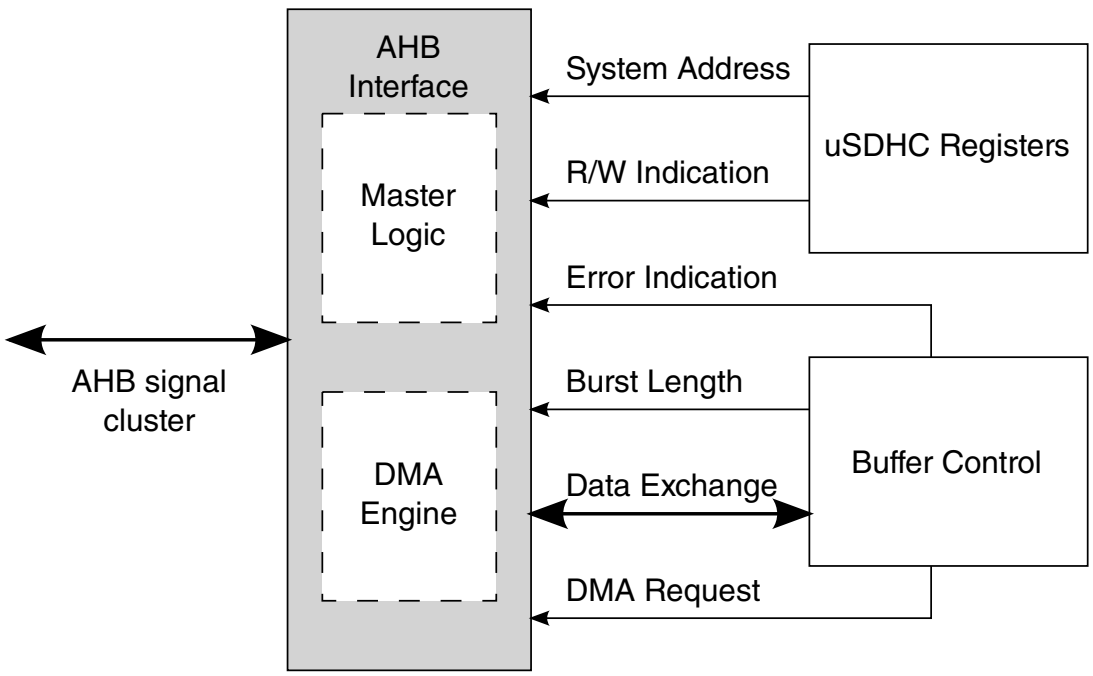



**Figure 63-7. DMA AHB Interface Block**

### 63.4.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request if the data buffer space(for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The uSDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuffed byte.

#### 63.4.2.2 DMA Burst Length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block.

See the example in Internal DMA Request. After 6 words are read, the burst length will be 2 words, then the next burst length will be 6 words. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

#### 63.4.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state.At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register will be generated to host CPU to report a bus error condition.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block. DMA operation will resume when the interrupt is serviced by software.

#### 63.4.2.4 ADMA Engine

In the SD Host Controller Standard, a new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver.

The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

#### 63.4.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer

See the figure below for the format of the descriptor table for ADMA1.

| Address/ Page Field | | Address/ Page Field | | Attribute Field | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 31 | 12 | 11 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address or Data Length | | 000000 | | Act 2 | Act 1 | 0 | Int | End | Valid |

| Act 2 | Act1 | Symbol | Comment | 31- 28 | 27- 12 |
|---|---|---|---|---|---|
| 0 | 0 | Nop | No Operation | Don't Care | |
| 0 | 1 | Set | Set Data Length | 0000 | Data Length |
| 1 | 0 | Tran | Transfer Data | Data Address | |
| 1 | 1 | Link | Link Descriptor | Descriptor Address | |

| | |
|---|---|
| Valid | Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA. |
| End | End = 1 indicates current descriptor is the ending one. |
| Int | Int = 1 generates DMA Interrupt when this descriptor is processed. |

**Figure 63-8. Format of the ADMA1 Descriptor Table**

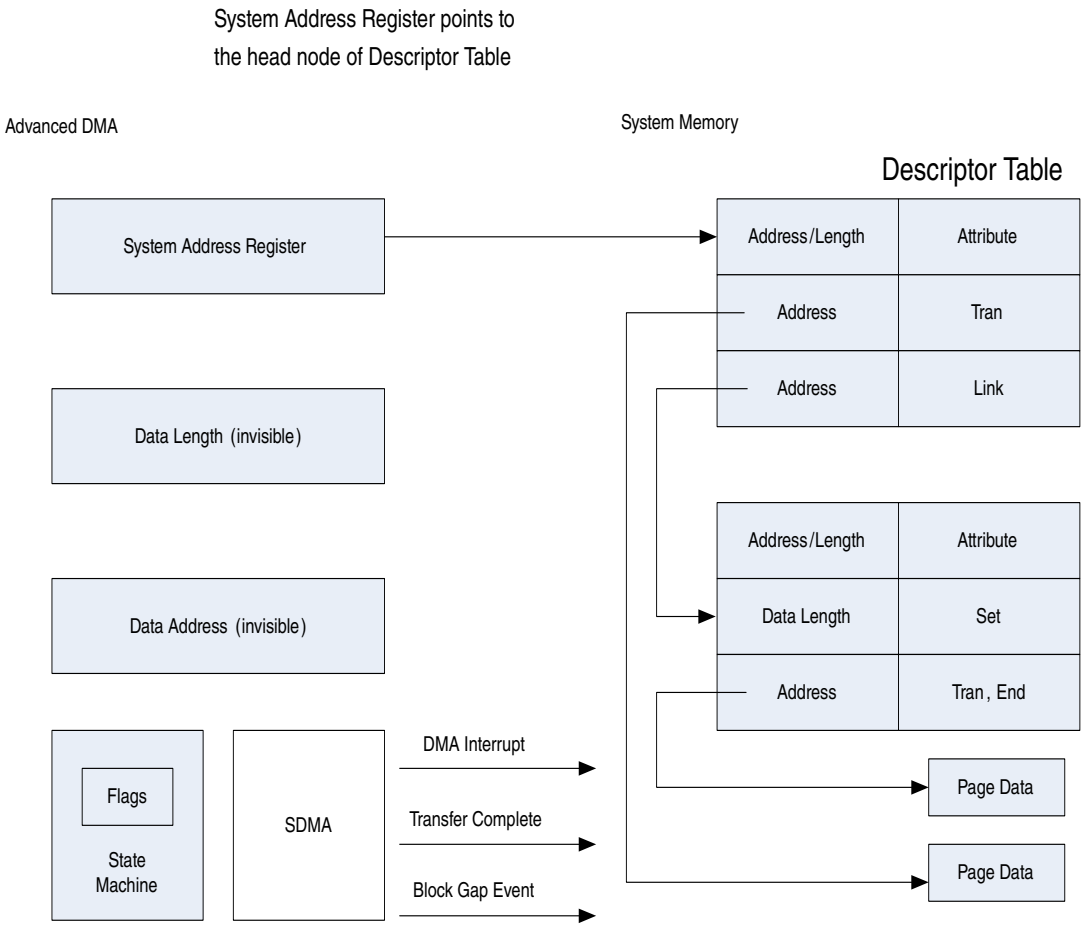



**Figure 63-9. Concept and Access Method of ADMA1 Descriptor Table**

The figure below explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned(lower 2-bit is always set to 0). Data length is in byte unit.

| Address Field | | Length | | Reserved | | Attribute Field | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 32 | 31 | 16 | 15 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 32-bit Address | | 16-bit length | | 0000000000 | | Act 2 | Act 1 | 0 | Int | End | Valid |

| Act 2 | Act1 | Symbol | Comment | Operation |
|---|---|---|---|---|
| 0 | 0 | Nop | No Operation | Don't Care |
| 0 | 1 | Rsv | Reserved | Same as Nop. Read this line and go to next one |
| 1 | 0 | Tran | Transfer Data | Transfer data with address and length set in this descriptor line |
| 1 | 1 | Link | Link Descriptor | Link to another descriptor |

| | |
|---|---|
| Valid | Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA. |
| End | End = 1 indicates current descriptor is the ending one. |
| Int | Int = 1 generates DMA Interrupt when this descriptor is processed. |

**Figure 63-10. Format of the ADMA2 Descriptor Table**

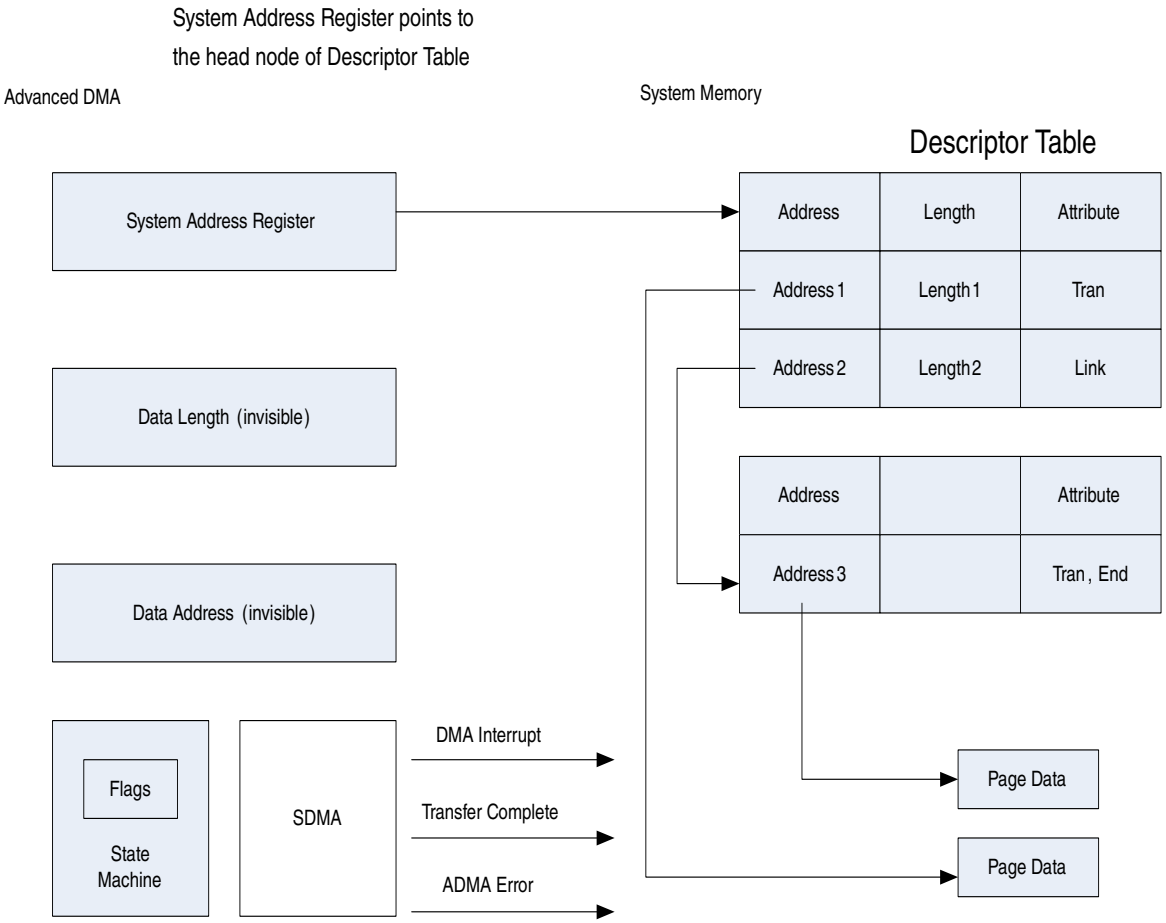



**Figure 63-11. Concept and Access Method of ADMA2 Descriptor Table**

### 63.4.2.4.2 ADMA Interrupt

If the interrupt flag descriptor is set, ADMA will generate an interrupt according to various types of descriptors:

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

#### 63.4.2.4.3 ADMA Error

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error will be generated when it fails to detect a 'Valid' flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

When BLKCNTEN bit is not set, then whole data length set in descriptor should be a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors will occur.

### 63.4.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank.

See the figure below for the block diagram.

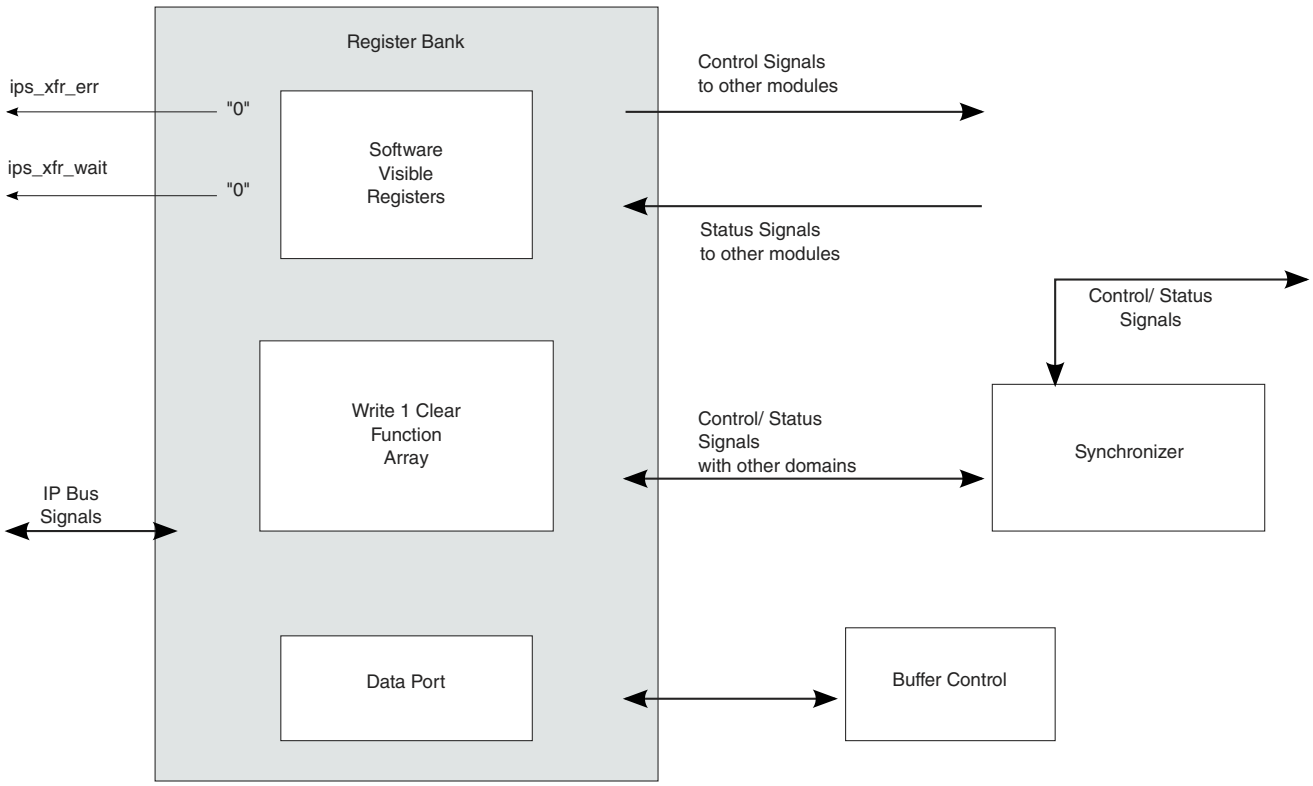



**Figure 63-12. Register Bank Diagram**

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

### 63.4.3.1 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub modules:

1. SD control misc.

2. Command control.
3. Data control.
4. Clock control

### 63.4.3.2 SD control misc

In the SD control misc unit, the card detect(include the CD_B and DATA3 used as Card Detection), write protection and card interrupt are implemented.

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The module also detects the WP (Write Protect) line. If WP is active, writes to the register bank will be ignored.

This module also drives the LCTL output signal when the LCTL bit is set by the driver.

### 63.4.3.3 SD Clock control

If the internal data buffer is near full(for read) or near empty(for write), the SD clock must be gated off to avoid buffer over/under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer has space(for read) or has data(for write), the clock gate of this module will open and the SD clock will be active again.

### 63.4.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC Shift Register.

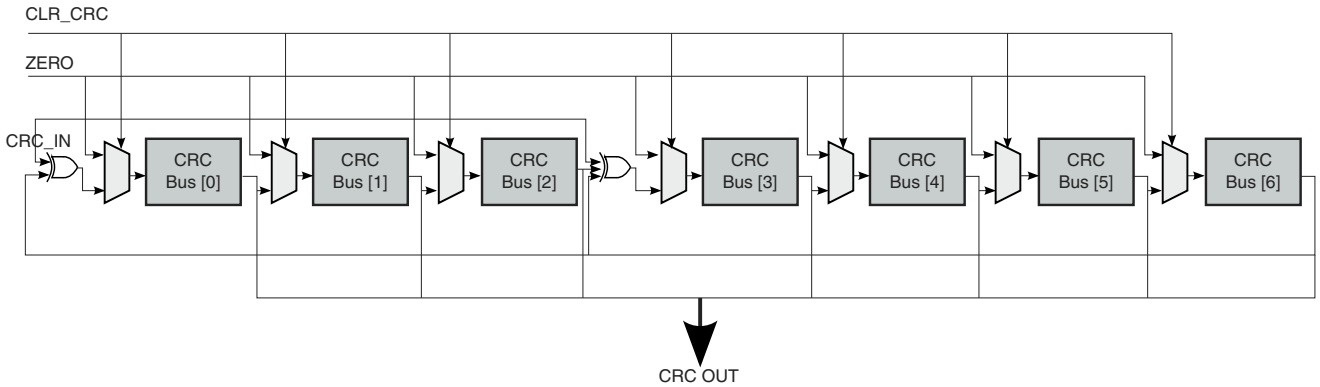



**Figure 63-13. Command CRC Shift Register**

The CRC polynomials for the CMD are as follows:

```
Generator polynomial: G(x) = x^7 + x^3 + 1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]
```

### 63.4.3.5 Data control

The Data Agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the Read Wait state by the request from the Transceiver.

The CRC polynomials for the DATA are as follows:

```
Generator polynomial: G(x) = x^16 + x^12 + x^5 +1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[15:0] = Remainder [(M(x) * x^16) / G(x)]
```

## 63.4.4 Clock & Reset Manager

This module controls all the reset signals within the uSDHC.

There are four kinds of reset signals within uSDHC:

1. Hardware reset.
2. Software reset for all logic.
3. Software reset for the data logic.
4. Software reset for the command logic.

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

## 63.4.5 Clock Generator

The Clock Generator generates the card CLK by peripheral source clock in two stages.

Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

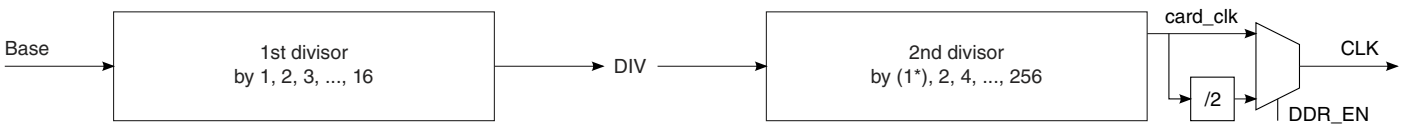



**Figure 63-14. Two Stages of the Clock Divider**

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual internal working clock (card_clk). This clock is the driving clock for all sub modules of the SD Protocol Unit, and the sync FIFOs (see Figure 63-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the card_clk is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of Base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

Please note, in SDR mode and DDR mode, the CLK are different.

- In SDR mode, CLK is equal to the internal working clock(card_clk).

- In DDR mode, CLK is equal to the card_clk/2.

## 63.4.6 SDIO Card Interrupt

Information on Interrupts in 1-bit Mode, Interrupts in 4-bit Mode, and Card Interrupt Handling are detailed in the sections below.

### 63.4.6.1 Interrupts in 1-bit Mode

In this case the DATA1 pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

### 63.4.6.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The uSDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line will be held low for one clock cycle with the last clock cycle pulling DATA1 high. On completion of the Interrupt Period, the card releases the DATA1 line into the high Z state. The uSDHC samples the DATA1 during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

### 63.4.6.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the uSDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. In 1-bit mode, the uSDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the

SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1, and the uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

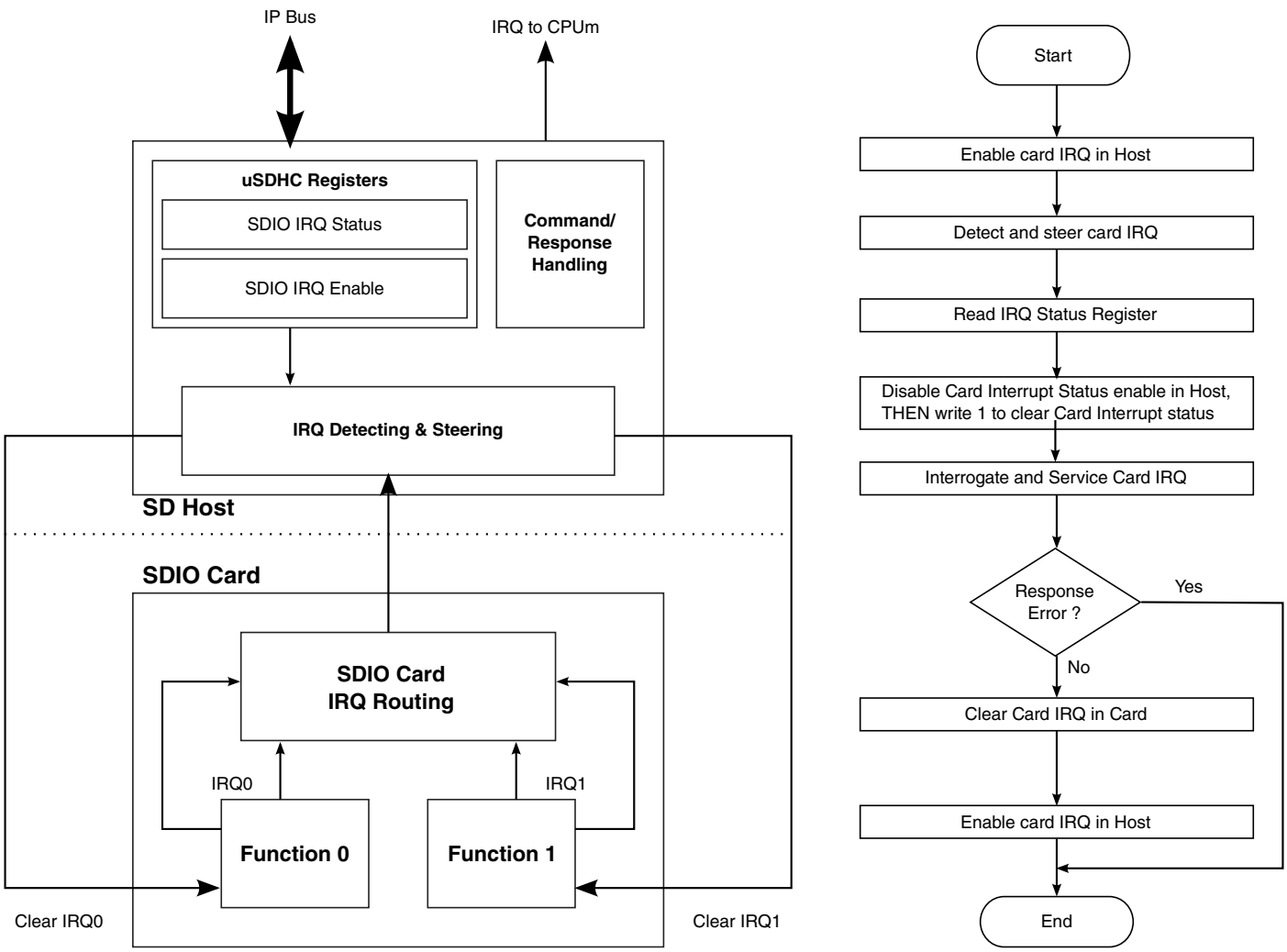



**Figure 63-15. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure**

### 63.4.7 Card Insertion and Removal Detection

The uSDHC uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 will be pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, the uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, the uSDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

### 63.4.8 Power Management and Wake Up Events

When there is no operation between the uSDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the uSDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The uSDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register, the clocks are gated in the low position to the uSDHC. For maximum power saving, the user can disable all the clocks to the uSDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

**NOTE**

To make the interrupt a wakeup event, when all the clocks to the uSDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be

set. Refer to Protocol Control (PROT_CTRL) for more information on the uSDHC Protocol Control register.

#### 63.4.8.1 Setting Wake Up Events

For the uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters sleep mode.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

### 63.4.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature which requires hardware support. There are two types of fast boot mode, boot operation, and alternative boot operation in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

**NOTE**

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

#### 63.4.9.1 Boot operation

**NOTE**

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

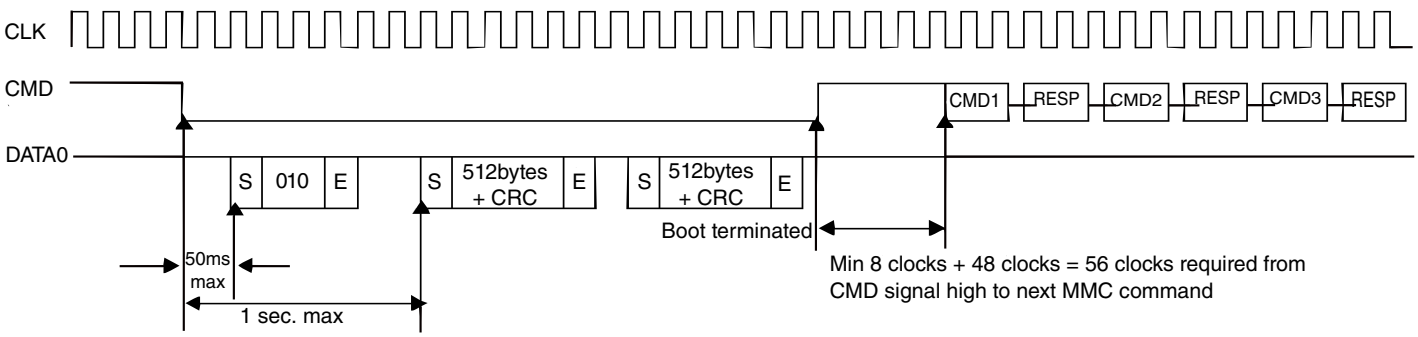



**Figure 63-16. MultiMediaCard state diagram (normal boot mode)**

### 63.4.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

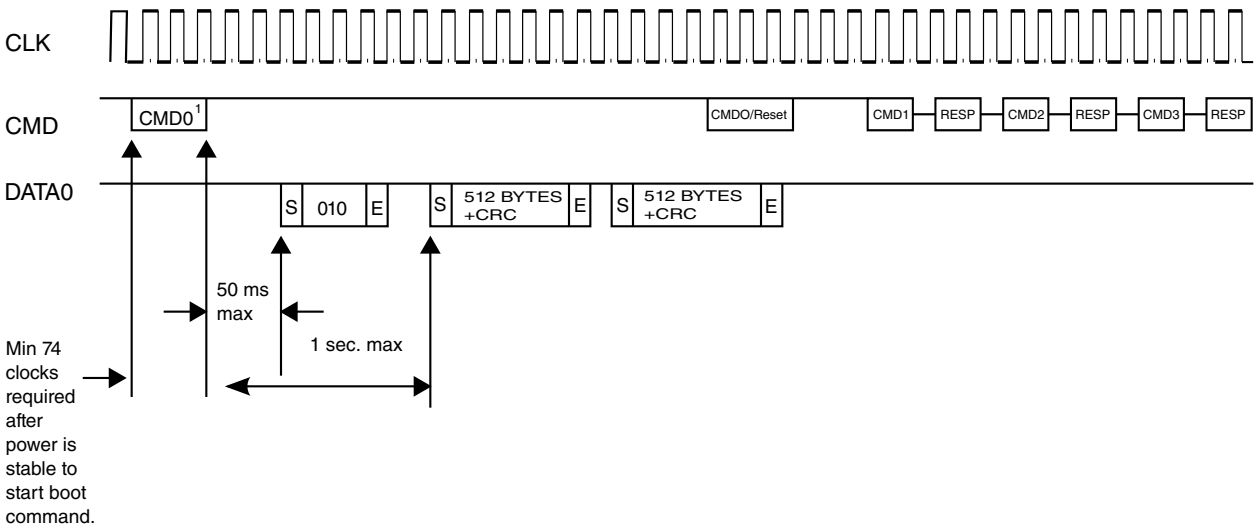
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, theslave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.




NOTE 1. CMD0 with argument 0xFFFFFFFA

**Figure 63-17. MultiMediaCard state diagram (alternative boot mode)**

## 63.5 Initialization/Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

### 63.5.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver will deal with "fake" errors like this with caution.

### 63.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

#### 63.5.2.1 Card Detect

See the figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the uSDHC.

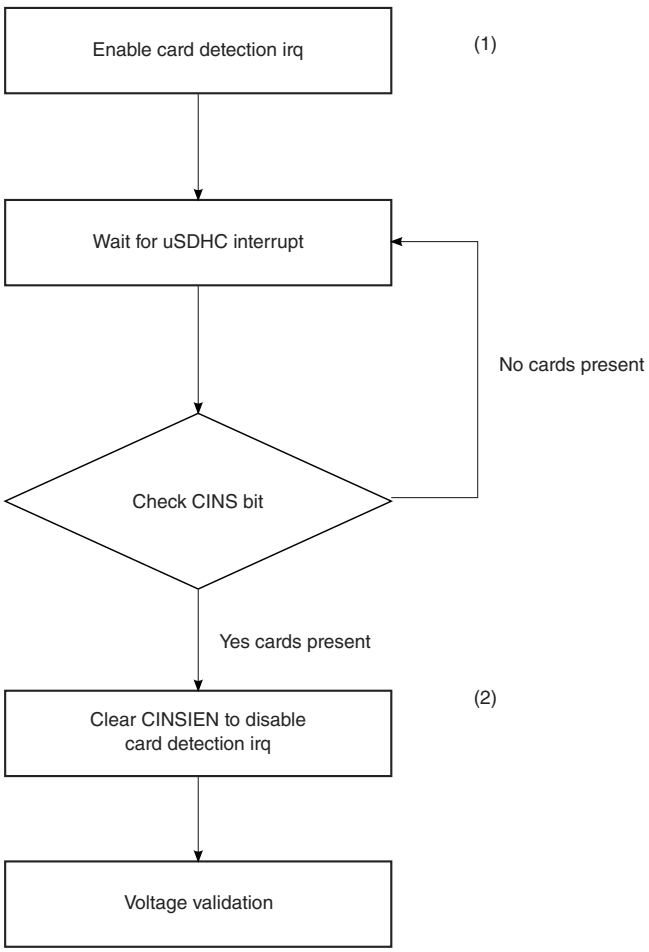



**Figure 63-18. Flow Diagram for Card Detection**

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

### 63.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)

- Software reset (Host Only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both the uSDHC and the card.

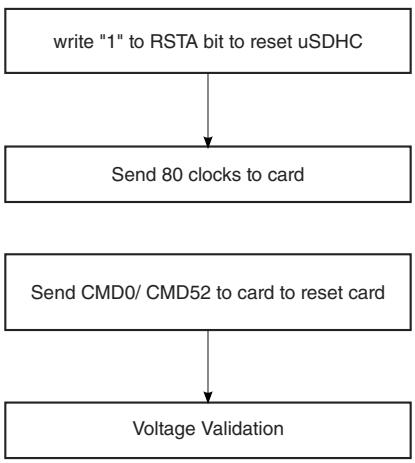



**Figure 63-19. Flow Chart for Reset of the uSDHC and SD I/O Card**

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

### 63.5.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range.

Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card  if (card is already labelled as SDCombo)
{ //
change label
```

```
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
} // of else
}
```

### 63.5.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different. For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions.

The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards

(the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC) {
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

## 63.5.3 Card Access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

### 63.5.3.1 Block Write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

#### 63.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DATA line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DATA line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DATA to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:

- For SD/MMC cards, use SET_BLOCKLEN (CMD16)
- For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

#### 63.5.3.1.2 DDR Write

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

#### 63.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the Stop At Block Gap Request(SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the uSDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The uSDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to

send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

### 63.5.3.2 Block Read

Information about Normal Read, DDR Read, Read with Pause, and DLL (Delay Line) in Read Path are detailed in the sections below.

#### 63.5.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other methods (CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

#### 63.5.3.2.2 DDR Read

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

#### 63.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks.

Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the uSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

5. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the uSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the uSDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the uSDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

#### 63.5.3.2.4 DLL (Delay Line) in Read Path

The DLL(Delay Line) is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT).

The reasons why the DLL is needed for uSDHC are 1.) the path of read data traveling from card to host varies. 2.) in SD/MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL will automatically adjust the number of delay cells according to target value set by user and PVT changes. Be aware that target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns, setting target value of 16 means 5ns =(16/32)*10ns. Software can disable automatic update by setting dll_gate_update bit.

Since the user may change the frequency of the card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. The following is the correct flow, which should be ignored if DLL_CTRL_ENABLE is not set.

Step 1: Set DLL_CTRL_RESET bit

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: clear DLL_CTRL_RESET bit

Step 5: Wait until both DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: set DLL_CTRL_SLV_FORCE_UPD

Step 7: clear DLL_CTRL_SLV_FORCE_UPD

**NOTE**

Software should make sure the DLL_CTRL_SLV_FORCE_UPD is lasted for at least one card_clk. So software may need to add some delay between Step6 and Step7.

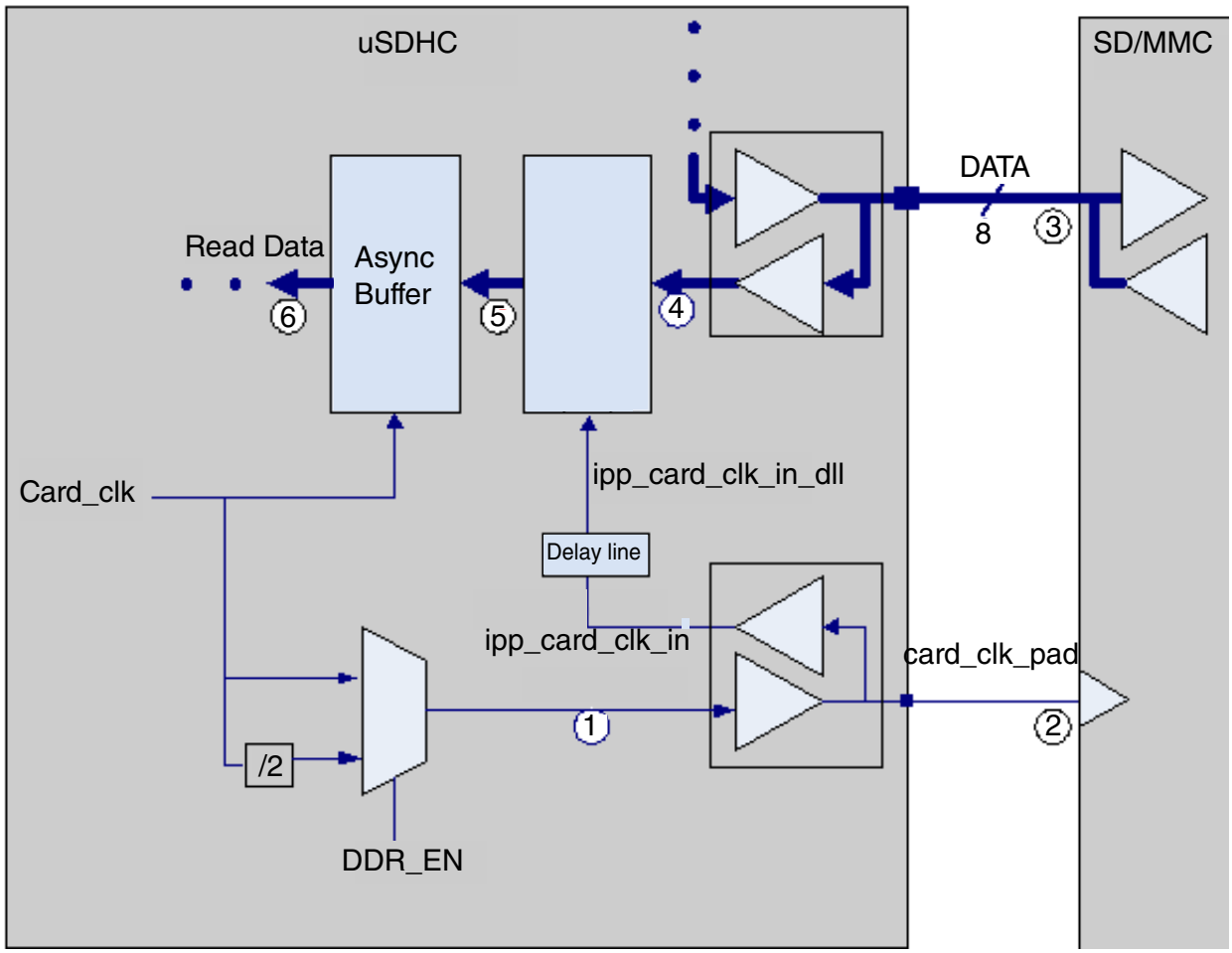



**Figure 63-20. DLL(Delay Line) in Read Path**

### 63.5.3.3 Suspend Resume

The uSDHC supports the Suspend Resume operations of SDIO cards, although slightly differently than the suggested implementation of Suspend in the SDIO card specification.

#### 63.5.3.3.1 Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it doesn't de-assert Read Wait for read pause. To solve this problem, the Driver shall not mark the Suspend command as a "Suspend", (i.e. setting the CMDTYP bits to 01). Instead, the Driver shall send this command as if it were a normal command, and only

when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform the uSDHC that the current transfer is suspended. As shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

#### 63.5.3.3.2 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

### 63.5.3.4 ADMA Usage

To use the ADMA in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command.

The steps to prepare the correct descriptor chain are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.

4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

#### 63.5.3.5 Transfer Error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 Errors are detailed in the sections below.

##### 63.5.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one.

For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the uSDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

##### 63.5.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs.

The start address can be calculated by either:

1. Reading the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the

next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Reading the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

#### 63.5.3.5.3 Transfer ADMA Error

There are 3 kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set.

For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

#### 63.5.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the uSDHC automatically sends a CMD12 to the card to stop the transfer.

When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

#### 63.5.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DATA1 line during some special period. The uSDHC only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the uSDHC, and the Host System is informed by the uSDHC asserting the uSDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written. Refer to Card Interrupt Handling for the card interrupt handling flow.

### 63.5.4 Switch Function

A switch command shall be issued by the Host Driver to enable new features added to the SD/MMC spec. SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed mode are also defined. To enable these features, a switch command shall be issued by the Host Driver.

For SDIO cards, the high speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards , the high speed mode/HS200 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit ,DDR4-bit and DDR8-bit width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

### 63.5.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

### 63.5.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/ SDR104/DDR50

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz for high speed mode, 100Mhz for SDR50, 200Mhz for SDR104, 50Mhz for
DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
```

```
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

### 63.5.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

### 63.5.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-
bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

## 63.5.5 ADMA Operation

Code for ADMA1 Operation and ADMA2 Operation can be found here.

### 63.5.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

### 63.5.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

## 63.5.6 Fast Boot Operation

### 63.5.6.1 Normal fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If the data will be sent through DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept at the default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.
7. DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt will be triggered, and software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
10. Reset the host and then can begin the normal process.

### 63.5.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
7. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host will send out the interrupt and software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process..
8. If there is no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

### 63.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow is described below:

1. Software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1).Here VALUE1 is the value of the block count that needs to transfer the first time, so that that the hostwill stop at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also configure bits[3-0] to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. Software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enbable ADMA2 by configuring Protocol Control Register bits [9-8].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (ie, in IRAM, at least 6 word). The first pair descriptor define the start address (ie, IRAM)and data length(ie,512byte*VALUE1) of fisrt part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word(record as VALUE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software needs to configure Transfer Type Register to start the boot process . CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin, the first VALUE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.

10. When TC, BGE bit is 1, . SW can analyzes the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of boot code(VALUE3 the remain boot code block). Remeber set the last descriptor with END.
11. Software needs to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the *(BLK_CNT - (VALUE1+1+VALUE3))*, that host will stop at block gap when *the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9*. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequence. Please note, Software doesn't need to configure the *BLK_CNT* again, because it's counted down automatically by the uSDHC controller.
12. Software needs to clear TC and BGE bit. And software needs to clear SABGREQ(bit 16 in Protocol control register), and set CREQ(bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

Note:

1. When ADMA boot flow is started, for uSDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

## 63.6 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards can be found here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DATA.
4. addressed (point-to-point) data transfer commands (adtc).

Response: a response is a token which is sent from the card to the host as an answner to a previously received command. A response is transferred serially on the CMD line.

**Table 63-3. Commands for MMC/SD/SDIO Cards**

| CMD INDEX | Type | Argument | Response type | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD0 | bc | [31:0] stuff bits | - | GO_IDLE_STATE | Resets all MMC and SD memory cards to idle state. |
| CMD1 | bcr | [31:0] OCR without busy | R3 | SEND_OP_COND | Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line. |
| CMD2 | bcr | [31:0] stuff bits | R2 | ALL_SEND_CID | Asks all cards to send their CID numbers on the CMD line. |
| CMD3[1] | ac | [31:6] RCA<br>[15:0] stuff bits | R1<br>R6 (SDIO) | SET/ SEND_RELATIVE_ADDR | Assigns relative address to the card. |
| CMD4 | bc | [31:0] DSR<br>[15:0] stuff bits | - | SET_DSR | Programs the DSR of all cards. |
| CMD5 | bc | [31:0] OCR without busy | R4 | IO_SEND_OP_COND | Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line. |
| CMD6[2] | adtc | [31] Mode<br>0: Check function<br>1: Switch function<br>[30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF)<br>[7:4] Function group1 for command system<br>[3:0] Function group2 for access mode | R1 | SWITCH_FUNC | Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details. |
| CMD6[3] | ac | [31:26] Set to 0<br>[25:24] Access<br>[23:16] Index<br>[15:8] Value<br>[7:3] Set to 0<br>[2:0] Cmd Set | R1b | SWITCH | Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details. |
| CMD7 | ac | [31:6] RCA<br>[15:0] stuff bits | R1b | SELECT/ DESELECT_CARD | Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all. |

*Table continues on the next page...*

## Table 63-3. Commands for MMC/SD/SDIO Cards (continued)

| CMD INDEX | Type | Argument | Response type | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD8 | adtc | [31:0] stuff bits | R1 | SEND_EXT_CSD | The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes. |
| CMD9 | ac | [31:6] RCA<br>[15:0] stuff bits | R2 | SEND_CSD | Addressed card sends its card-specific data (CSD) on the CMD line. |
| CMD10 | ac | [31:6] RCA<br>[15:0] stuff bits | R2 | SEND_CID | Addressed card sends its card-identification (CID) on the CMD line. |
| CMD11 | adtc | [31:0] data address | R1 | READ_DAT_UNTIL_STOP | Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows. |
| CMD12 | ac | [31:0] stuff bits | R1b | STOP_TRANSMISSION | Forces the card to stop transmission. |
| CMD13 | ac | [31:6] RCA<br>[15:0] stuff bits | R1 | SEND_STATUS | Addressed card sends its status register. |
| CMD14 | Reserved | | | | |
| CMD15 | ac | [31:6] RCA<br>[15:0] stuff bits | - | GO_INACTIVE_STATE | Sets the card to inactive state in order to protect the card stack against communication breakdowns. |
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCKLEN | Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD. |
| CMD17 | adtc | [31:0] data address | R1 | READ_SINGLE_BLOCK | Reads a block of the size selected by the SET_BLOCKLEN command. |
| CMD18 | adtc | [31:0] data address | R1 | READ_MULTIPLE_BLOCK | Continuously transfers data blocks from card to host until interrupted by a stop command. |
| CMD19 | Reserved | | | | |
| CMD20 | adtc | [31:0] data address | R1 | WRITE_DAT_UNTIL_STOP | Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows. |
| CMD21 | adtc | [31:0] stuff bits | R1 | SEND_TUNING_BLOCK | 128 clocks of tuning pattern (64 byte in 4 bit mode or 128 byte in 8 bit mode) is sent for HS200 optimal sampling point detection. |
| CMD22-23 | Reserved | | | | |
| CMD24 | adtc | [31:0] data address | R1 | WRITE_BLOCK | Writes a block of the size selected by the SET_BLOCKLEN command. |
| CMD25 | adtc | [31:0] data address | R1 | WRITE_MULTIPLE_BLOCK | Continuously writes blocks of data until a STOP_TRANSMISSION follows. |
| CMD26 | adtc | [31:0] stuff bits | R1 | PROGRAM_CID | Programming of the card identification register. This command shall be issued only |

*Table continues on the next page...*

## Table 63-3. Commands for MMC/SD/SDIO Cards (continued)

| CMD INDEX | Type | Argument | Response type | Abbreviation | Description |
|---|---|---|---|---|---|
| | | | | | once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer. |
| CMD27 | adtc | [31:0] stuff bits | R1 | PROGRAM_CSD | Programming of the programmable bits of the CSD. |
| CMD28 | ac | [31:0] data address | R1b | SET_WRITE_PROT | If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE). |
| CMD29 | ac | [31:0] data address | R1b | CLR_WRITE_PROT | If the card provides write protection features, this command clears the write protection bit of the addressed group. |
| CMD30 | adtc | [31:0] write protect data address | R1 | SEND_WRITE_PROT | If the card provides write protection features, this command asks the card to send the status of the write protection bits. |
| CMD31 | Reserved | | | | |
| CMD32 | ac | [31:0] data address | R1 | TAG_SECTOR_START | Sets the address of the first sector of the erase group. |
| CMD33 | ac | [31:0] data address | R1 | TAG_SECTOR_END | Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase. |
| CMD34 | ac | [31:0] data address | R1 | UNTAG_SECTOR | Removes one previously selected sector from the erase selection. |
| CMD35 | ac | [31:0] data address | R1 | TAG_ERASE_GROUP_START | Sets the address of the first erase group within a range to be selected for erase. |
| CMD36 | ac | [31:0] data address | R1 | TAG_ERASE_GROUP_END | Sets the address of the last erase group within a continuous range to be selected for erase. |
| CMD37 | ac | [31:0] data address | R1 | UNTAG_ERASE_GROUP | Removes one previously selected erase group from the erase selection. |
| CMD38 | ac | [31:0] stuff bits | R1b | ERASE | Erase all previously selected sectors. |
| CMD39 | ac | [31:0] RCA<br>[15] register write flag<br>[14:8] register address<br>[7:0] register data | R4 | FAST_IO | Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This |

*Table continues on the next page...*

## Table 63-3. Commands for MMC/SD/SDIO Cards (continued)

| CMD INDEX | Type | Argument | Response type | Abbreviation | Description |
|---|---|---|---|---|---|
| | | | | | command accesses application dependent registers which are not defined in the MMC standard. |
| CMD40 | bcr | [31:0] stuff bits | R5 | GO_IRQ_STATE | Sets the system into interrupt mode. |
| CMD41 | Reserved | | | | |
| CDM42 | adtc | [31:0] stuff bits | R1b | LOCK_UNLOCK | Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command. |
| CMD43~51 | Reserved | | | | |
| CMD52 | ac | [31:0] stuff bits | R5 | IO_RW_DIRECT | Access a single register within the total 128k of register space in any I/O function. |
| CMD53 | ac | [31:0] stuff bits | R5 | IO_RW_EXTENDED | Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers. |
| CMD54 | Reserved | | | | |
| CMD55 | ac | [31:16] RCA<br>[15:0] stuff bits | R1 | APP_CMD | Indicates to the card that the next command is an application specific command rather that a standard command. |
| CMD56 | adtc | [31:1] stuff bits<br>[0]: RD/WR | R1b | GEN_CMD | Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command. |
| CMD57-59 | Reserved | | | | |
| CMD60 | adtc | [31] WR<br>[30:24] stuff bits<br>[23:16] address<br>[15:8] stuff bits<br>[7:0] byte count | R1b | RW_MULTIPLE_REGISTER | These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers. |
| CMD61 | adtc | [31] WR<br>[30:16] stuff bits<br>[15:0] data unit count | R1b | RW_MULTIPLE_BLOCK | The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer. |
| CMD62-63 | Reserved | | | | |
| ACMD6[4] | ac | [31:2] stuff bits<br>[1:0] bus width | R1 | SET_BUS_WIDTH | Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register. |

*Table continues on the next page...*

**Table 63-3. Commands for MMC/SD/SDIO Cards (continued)**

| CMD INDEX | Type | Argument | Response type | Abbreviation | Description |
|---|---|---|---|---|---|
| ACMD13[4] | adtc | [31:0] stuff bits | R1 | SD_STATUS | Send the SD Memory Card status. |
| ACMD22[4] | adtc | [31:0] stuff bits | R1 | SEND_NUM_WR_SECTORS | Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block. |
| ACMD23[4] | ac | [31:23] stuff bits<br>[22:0] Number of blocks | R1 | SET_WR_BLK_ERASE_COUNT | Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block). |
| ACMD41[4] | bcr | [31:0] OCR | R3 | SD_APP_OP_COND | Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line. |
| ACMD42[4] | ac | [31:1] stuff bits<br>[0] set_cd | R1 | SET_CLR_CARD_DETECT | Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/DATA3 of the card. |
| ACMD51[4] | adtc | [31:0] stuff bits | R1 | SEND_SCR | Reads the SD Configuration Register (SCR). |

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown below.

**Table 63-4. EXT_CSD Access Modes**

| Bits | Access Name | Operation |
|---|---|---|
| 00 | Command Set | The command set is changed according to the Cmd Set field of the argument |
| 01 | Set Bits | The bits in the pointed byte are set, according to the 1 bits in the Value field. |
| 10 | Clear Bits | The bits in the pointed byte are cleared, according to the 1 bits in the Value field. |
| 11 | Write Byte | The Value field is written into the pointed byte. |

# 63.7 Software Restrictions

### 63.7.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared.

### 63.7.2 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.

For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

### 63.7.3 Suspend Operation

In order to suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such 'suspend' command, uSDHC will regard the current transfer is aborted and change BLKCNT register to its original value, instead of keeping the remained number of blocks.

### 63.7.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

### 63.7.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

### 63.7.6 Data Port Access

Data Port does not support parallel access. For example, during an internal DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or internal DMA. Otherwise the data would be corrupted inside the uSDHC buffer.

### 63.7.7 Change Clock Frequency

uSDHC does not automatically gate off the card clock when the Host Driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing clock divisor value(SDCLKFS or DVS in System Control Register) or setting RSTA bit.

Also before changing the clock divisor value, Host Driver should make sure the SDSTB bit is high.

### 63.7.8 Multi-block Read

For pre-defined multi-block read operation, i.e., the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode.

In this case, the card may not respond to this extra abort command and uSDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

## 63.8 uSDHC Memory Map/Register Definition

## 63.8.1 uSDHC register descriptions

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map for the uSDHC. All these registers only support 32-bit accesses.

**NOTE**

The uSDHC registers are 32-bit wide and only support 32-bit access.

### 63.8.1.1 uSDHC Memory map

USDHC0 base address: 4003_E000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | DMA System Address (DS_ADDR) | 32 | RW | 0000_0000h |
| 4h | Block Attributes (BLK_ATT) | 32 | RW | 0000_0000h |
| 8h | Command Argument (CMD_ARG) | 32 | RW | 0000_0000h |
| Ch | Command Transfer Type (CMD_XFR_TYP) | 32 | RW | 0000_0000h |
| 10h | Command Response0 (CMD_RSP0) | 32 | RO | 0000_0000h |
| 14h | Command Response1 (CMD_RSP1) | 32 | RO | 0000_0000h |
| 18h | Command Response2 (CMD_RSP2) | 32 | RO | 0000_0000h |
| 1Ch | Command Response3 (CMD_RSP3) | 32 | RO | 0000_0000h |
| 20h | Data Buffer Access Port (DATA_BUFF_ACC_PORT) | 32 | RW | 0000_0000h |
| 24h | Present State (PRES_STATE) | 32 | RO | See description. |
| 28h | Protocol Control (PROT_CTRL) | 32 | RW | 0880_0020h |
| 2Ch | System Control (SYS_CTRL) | 32 | RW | 0080_800Fh |
| 30h | Interrupt Status (INT_STATUS) | 32 | W1C | 0000_0000h |
| 34h | Interrupt Status Enable (INT_STATUS_EN) | 32 | RW | 0000_0000h |
| 38h | Interrupt Signal Enable (INT_SIGNAL_EN) | 32 | RW | 0000_0000h |
| 3Ch | Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS) | 32 | RO | 0000_0000h |
| 40h | Host Controller Capabilities (HOST_CTRL_CAP) | 32 | RO | 07F3_0000h |
| 44h | Watermark Level (WTMK_LVL) | 32 | RW | 0810_0810h |
| 48h | Mixer Control (MIX_CTRL) | 32 | RW | 8000_0000h |
| 50h | Force Event (FORCE_EVENT) | 32 | WORZ | 0000_0000h |
| 54h | ADMA Error Status Register (ADMA_ERR_STATUS) | 32 | RO | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 58h | ADMA System Address (ADMA_SYS_ADDR) | 32 | RW | 0000_0000h |
| C0h | Vendor Specific Register (VEND_SPEC) | 32 | RW | 2000_7809h |
| C4h | MMC Boot Register (MMC_BOOT) | 32 | RW | 0000_0000h |
| C8h | Vendor Specific 2 Register (VEND_SPEC2) | 32 | RW | 0000_1006h |

## 63.8.1.2 DMA System Address (DS_ADDR)

### 63.8.1.2.1 Offset

| Register | Offset |
|---|---|
| DS_ADDR | 0h |

### 63.8.1.2.2 Function

This register contains the physical system memory address used for DMA transfers.

### 63.8.1.2.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DS_ADDR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DS_ADDR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 63.8.1.2.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DS_ADDR | DS_ADDR<br>**DMA System Address / Argument 2** |

| Field | Function |
|---|---|
| | When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23.<br>1. **SDMA System Address**<br>Since the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When the uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register.<br>This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until the DLA bit in the Present State register is cleared, before writing to this register.<br>The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC will automatically change SEQ burst type to NSEQ.<br>Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions.<br>2. **Argument 2**<br>This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23.<br>If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case. |

## 63.8.1.3 Block Attributes (BLK_ATT)

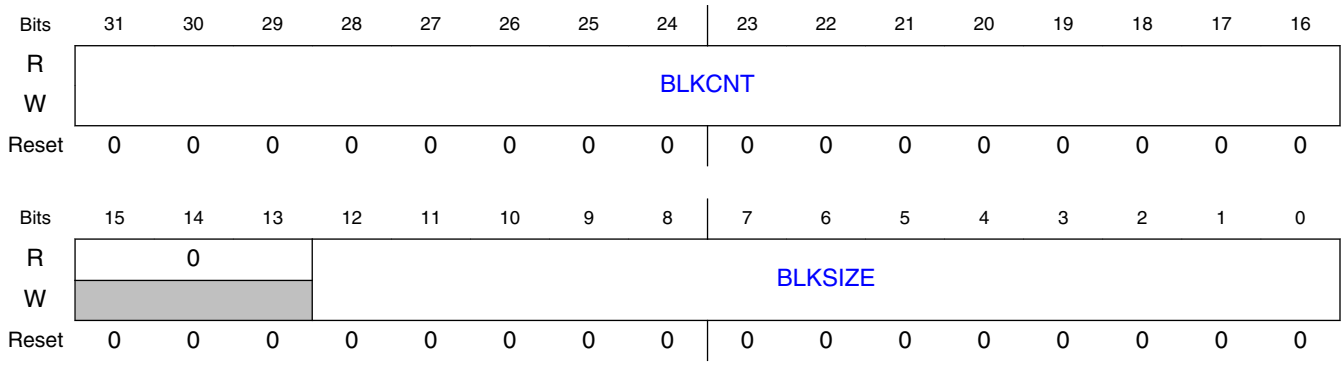
### 63.8.1.3.1 Offset

| Register | Offset |
|---|---|
| BLK_ATT | 4h |

### 63.8.1.3.2 Function

This register is used to configure the number of data blocks and the number of bytes in each block.

### 63.8.1.3.3 Diagram




### 63.8.1.3.4 Fields

| Field | Function |
|---|---|
| 31-16<br>BLKCNT | Block Count<br>Blocks Count For Current Transfer:<br>This register is enabled when the Block Count Enable bit in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The uSDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred.<br>This register should be accessed only when no transaction is executing (i.e. after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored.<br>When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, uSDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count.<br>When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count.<br>**NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL bit is indicating a single block transfer, the read value of BLKCNT is always 1.<br>0000000000000000b - Stop Count<br>0000000000000001b - 1 block<br>0000000000000010b - 2 blocks<br>1111111111111111b - 65535 blocks |
| 15-13<br>— | Reserved. |
| 12-0<br>BLKSIZE | Block Size<br>Transfer Block Size: |

| Field | Function |
|---|---|
| | This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored.<br>0000000000000b - No data transfer<br>0000000000001b - 1 Byte<br>0000000000010b - 2 Bytes<br>0000000000011b - 3 Bytes<br>0000000000100b - 4 Bytes<br>0000111111111b - 511 Bytes<br>0001000000000b - 512 Bytes<br>0100000000000b - 2048 Bytes<br>1000000000000b - 4096 Bytes |

## 63.8.1.4 Command Argument (CMD_ARG)

### 63.8.1.4.1 Offset

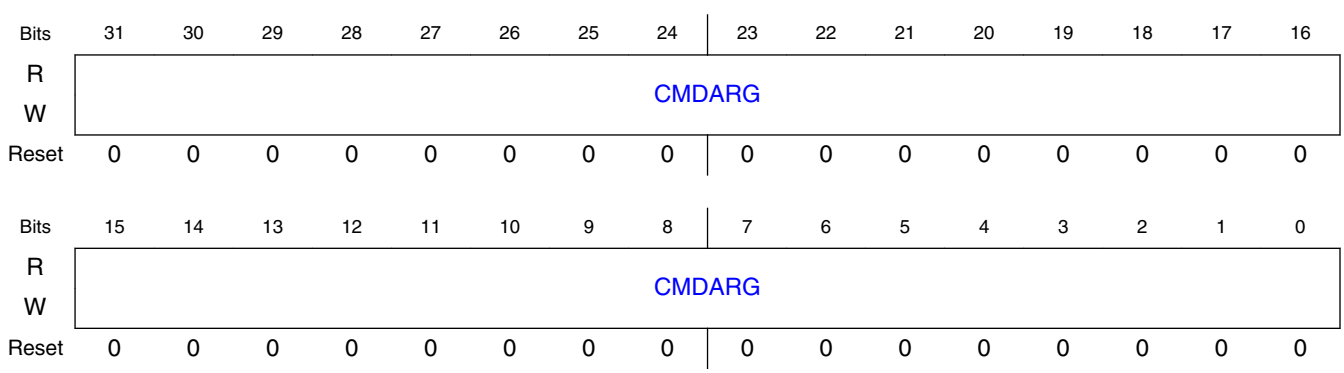
| Register | Offset |
|---|---|
| CMD_ARG | 8h |

### 63.8.1.4.2 Function

This register contains the SD / MMC Command Argument.

.

### 63.8.1.4.3 Diagram



#### 63.8.1.4.4 Fields

| Field | Function |
|---|---|
| 31-0<br>CMDARG | Command Argument<br>The SD / MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set. |

### 63.8.1.5 Command Transfer Type (CMD_XFR_TYP)

#### 63.8.1.5.1 Offset

| Register | Offset |
|---|---|
| CMD_XFR_TYP | Ch |

#### 63.8.1.5.2 Function

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the uSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer invovled, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1'), otherwise uSDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, i.e., response time-out, uSDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to

re-try the transfer. It is also possible that for some reason the card responds the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

**Table 63-5. Transfer Type Register Setting for Various Transfer Types**

| Multi/Single Block Select | Block Count Enable | Block Count | Function |
|---|---|---|---|
| 0 | Don't Care | Don't Care | Single Transfer |
| 1 | 0 | Don't Care | Infinite Transfer |
| 1 | 1 | Positive Number | Multiple Transfer |
| 1 | 1 | Zero | No Data Transfer |

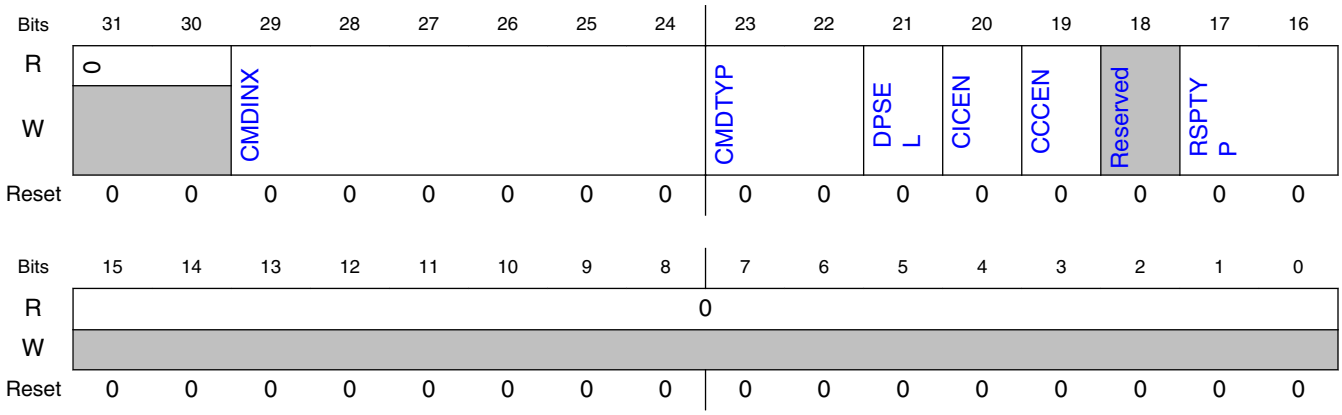
The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits as well as the name of the response type.

**Table 63-6. Relationship Between Parameters and the Name of the Response Type**

| Response Type | Index Check Enable | CRC Check Enable | Name of Response Type |
|---|---|---|---|
| 00 | 0 | 0 | No Response |
| 01 | 0 | 1 | R2 |
| 10 | 0 | 0 | R3,R4 |
| 10 | 1 | 1 | R1,R5,R6 |
| 11 | 1 | 1 | R1b,R5b |

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the uSDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

### 63.8.1.5.3 Diagram




### 63.8.1.5.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | -<br>Reserved |
| 29-24<br>CMDINX | Command Index<br>These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification. |
| 23-22<br>CMDTYP | Command Type<br>There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands.<br>• Suspend Command: If the Suspend command succeeds, the uSDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DATA line. Since the uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the Host Driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the uSDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, the uSDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the uSDHC will maintain its current state, and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register.<br>• Resume Command: The Host Driver re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The uSDHC will check for a pending busy state before starting write transfers.<br>• Abort Command: If this command is set when executing a read transfer, the uSDHC will stop reads to the buffer. If this command is set when executing a write transfer, the uSDHC will stop driving the DATA line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction).<br>00b - Normal Other commands<br>01b - Suspend CMD52 for writing Bus Suspend in CCCR<br>10b - Resume CMD52 for writing Function Select in CCCR<br>11b - Abort CMD12, CMD52 for writing I/O Abort in CCCR |
| 21 | Data Present Select |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DPSEL | This bit is set to 1 to indicate that data is present and shall be transferred using the DATA line. It is set to 0 for the following:<br>• Commands using only the CMD line (e.g. CMD52).<br>• Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b e.g. CMD38)<br>**NOTE:** In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (i.e. the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored.<br>0b - No Data Present<br>1b - Data Present |
| 20<br>CICEN | Command Index Check Enable<br>If this bit is set to 1, the uSDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.<br>0b - Disable<br>1b - Enable |
| 19<br>CCCEN | Command CRC Check Enable<br>If this bit is set to 1, the uSDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP) .)<br>0b - Disable<br>1b - Enable |
| 18<br>— | -<br>Reserved |
| 17-16<br>RSPTYP | Response Type Select<br>00b - No Response<br>01b - Response Length 136<br>10b - Response Length 48<br>11b - Response Length 48, check Busy after response |
| 15-0<br>— | -<br>Reserved |

## 63.8.1.6 Command Response0 (CMD_RSP0)

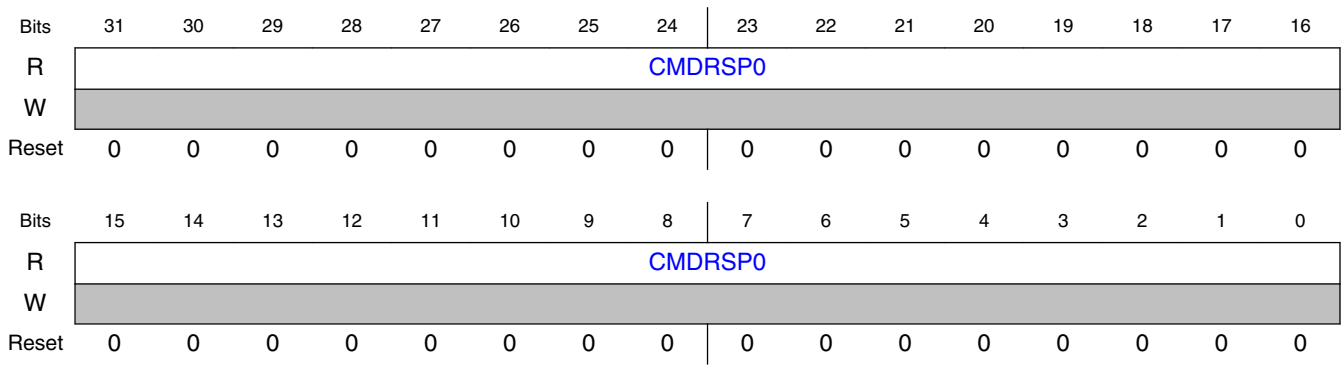
### 63.8.1.6.1 Offset

| Register | Offset |
|---|---|
| CMD_RSP0 | 10h |

#### 63.8.1.6.2 Function

This register is used to store part 0 of the response bits from the card.

#### 63.8.1.6.3 Diagram




#### 63.8.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>CMDRSP0 | Command Response 0<br>Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type. |

### 63.8.1.7 Command Response1 (CMD_RSP1)

#### 63.8.1.7.1 Offset

| Register | Offset |
|---|---|
| CMD_RSP1 | 14h |

#### 63.8.1.7.2 Function

This register is used to store part 1 of the response bits from the card.

#### 63.8.1.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 63.8.1.7.4 Fields

| Field | Function |
|---|---|
| 31-0<br>CMDRSP1 | Command Response 1<br>Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type. |

### 63.8.1.8 Command Response2 (CMD_RSP2)

#### 63.8.1.8.1 Offset

| Register | Offset |
|---|---|
| CMD_RSP2 | 18h |

#### 63.8.1.8.2 Function

This register is used to store part 2 of the response bits from the card.

### 63.8.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 63.8.1.8.4 Fields

| Field | Function |
|---|---|
| 31-0<br>CMDRSP2 | Command Response 2<br>Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type. |

## 63.8.1.9 Command Response3 (CMD_RSP3)

### 63.8.1.9.1 Offset

| Register | Offset |
|---|---|
| CMD_RSP3 | 1Ch |

### 63.8.1.9.2 Function

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus.

**Table 63-7. Response Bit Definition for Each Response Type**

| Response Type | Meaning of Response | Response Field | Response Register |
|---|---|---|---|
| R1,R1b (normal response) | Card Status | R[39:8] | CMDRSP0 |

*Table continues on the next page...*

**Table 63-7. Response Bit Definition for Each Response Type (continued)**

| Response Type | Meaning of Response | Response Field | Response Register |
|---|---|---|---|
| R1b (Auto CMD12 response) | Card Status for Auto CMD12 | R[39:8] | CMDRSP3 |
| R2 (CID, CSD register) | CID/CSD register [127:8] | R[127:8] | {CMDRSP3[23:0],<br>CMDRSP2,<br>CMDRSP1,<br>CMDRSP0} |
| R3 (OCR register) | OCR register for memory | R[39:8] | CMDRSP0 |
| R4 (OCR register) | OCR register for I/O etc. | R[39:8] | CMDRSP0 |
| R5, R5b | SDIO response | R[39:8] | CMDRSP0 |
| R6 (Publish RCA) | New Published RCA[31:16] and card status[15:0] | R[39:9] | CMDRSP0 |

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the uSDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the uSDHC will check R[47:1], and if the response length is 136 the uSDHC will check R[119:1].

Since the uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows the uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

### 63.8.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMDRSP3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 63.8.1.9.4 Fields

| Field | Function |
|---|---|
| 31-0<br>CMDRSP3 | Command Response 3<br>Refer to Command Response3 (CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type. |

## 63.8.1.10 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

### 63.8.1.10.1 Offset

| Register | Offset |
|---|---|
| DATA_BUFF_ACC_PORT | 20h |

### 63.8.1.10.2 Function

This is a 32-bit data port register used to access the internal buffer.

### 63.8.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATCONT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATCONT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 63.8.1.10.4 Fields

| Field | Function |
|---|---|
| 31-0<br>DATCONT | Data Content<br>The Buffer Data Port register is for 32-bit data access by the ARM platform. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s. |

## 63.8.1.11 Present State (PRES_STATE)

### 63.8.1.11.1 Offset

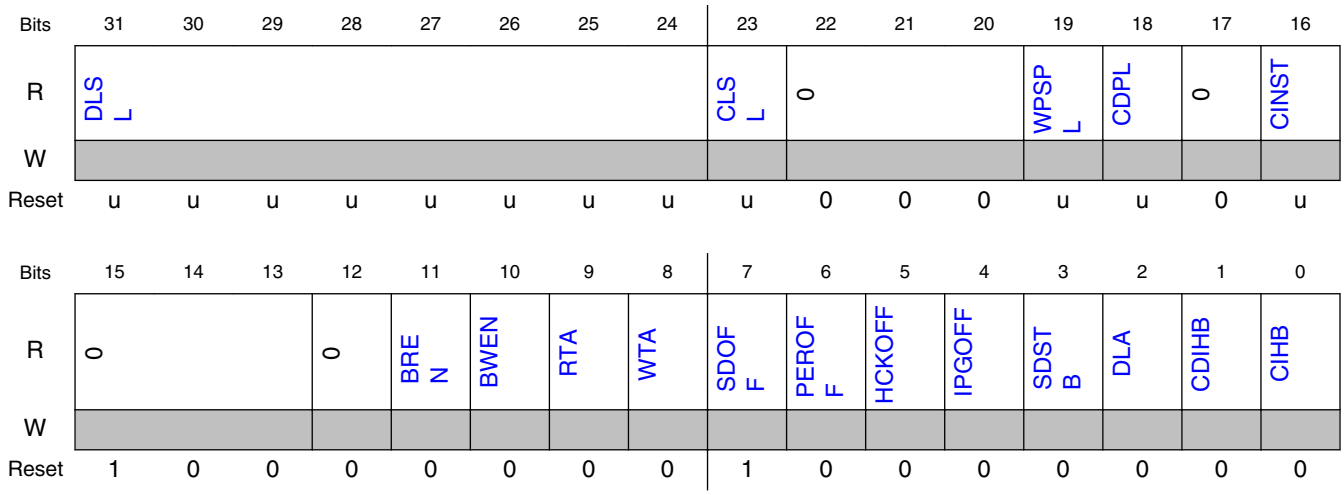
| Register | Offset |
|---|---|
| PRES_STATE | 24h |

### 63.8.1.11.2 Function

The Host Driver can get status of the uSDHC from this 32-bit read only register.

- The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.
- Note: the reset value of Present State Register depend on board connectivity.

## 63.8.1.11.3 Diagram




## 63.8.1.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>DLSL | DATA[7:0] Line Signal Level<br>This status is used to check the DATA line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up.<br>00000000b - Data 0 line signal level<br>00000001b - Data 1 line signal level<br>00000010b - Data 2 line signal level<br>00000011b - Data 3 line signal level<br>00000100b - Data 4 line signal level<br>00000101b - Data 5 line signal level<br>00000110b - Data 6 line signal level<br>00000111b - Data 7 line signal level |
| 23<br>CLSL | CMD Line Signal Level<br>This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up. |
| 22-20<br>— | -<br>Reserved |
| 19<br>WPSPL | Write Protect Switch Pin Level<br>The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled.<br>0b - Write protected (WP = 1)<br>1b - Write enabled (WP = 0) |
| 18 | Card Detect Pin Level |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CDPL | This bit reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the CD_B pin (i.e. when a card is inserted in the socket, it is 0 on the CD_B input, and consequently the CDPL reads 1.)<br>0b - No card present (CD_B = 1)<br>1b - Card present (CD_B = 0) |
| 17<br>— | -<br>Reserved |
| 16<br>CINST | Card Inserted<br>This bit indicates whether a card has been inserted. The uSDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit.<br>The Software Reset For All in the System Control register does not effect this bit. A software reset does not effect this bit.<br>0b - Power on Reset or No Card<br>1b - Card Inserted |
| 15-13<br>— | -<br>Reserved |
| 12<br>— | -<br>Reserved |
| 11<br>BREN | Buffer Read Enable<br>This status bit is used for non-DMA read transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when some reads from the buffer(read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled.<br>0b - Read disable<br>1b - Read enable |
| 10<br>BWEN | Buffer Write Enable<br>This status bit is used for non-DMA write transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when some writes to the buffer(write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled.<br>0b - Write disable<br>1b - Write enable |
| 9<br>RTA | Read Transfer Active<br>This status bit is used for detecting completion of a read transfer.<br>This bit is set for either of the following conditions: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • After the end bit of the read command.<br>• When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer.<br><br>A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions:<br><br>• When the last data block as specified by block length is transferred to the System, i.e. all data are read away from uSDHC internal buffer.<br>• When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1.<br><br>0b - No valid data<br>1b - Transferring data |
| 8<br>WTA | Write Transfer Active<br><br>This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the uSDHC.<br><br>This bit is set in either of the following cases:<br><br>• After the end bit of the write command.<br>• When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer.<br><br>This bit is cleared in either of the following cases:<br><br>• After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple).<br>• After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request.<br><br>During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state.<br><br>0b - No valid data<br>1b - Transferring data |
| 7<br>SDOFF | SD Clock Gated Off Internally<br><br>This status bit indicates that the SD Clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON bit is 0 to stop the SD clock in idle status. Set IPG_PERCLK_SOFT_EN and CARD_CLK_SOFT_EN to 0 also gate off SD clock. This bit is for the Host Driver to debug data transaction on the SD bus.<br><br>0b - SD Clock is active.<br>1b - SD Clock is gated off. |
| 6<br>PEROFF | IPG_PERCLK Gated Off Internally<br><br>This status bit indicates that the IPG_PERCLK is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK will be gated off, otherwise IPG_PERCLK will be always active.<br><br>0b - IPG_PERCLK is active.<br>1b - IPG_PERCLK is gated off. |
| 5<br>HCKOFF | HCLK Gated Off Internally<br><br>This status bit indicates that the HCLK is internally gated off. This bit is for the Host Driver to debug during a data transfer.<br><br>0b - HCLK is active.<br>1b - HCLK is gated off. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 4<br>IPGOFF | IPG_CLK Gated Off Internally<br>This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug.<br>0b - IPG_CLK is active.<br>1b - IPG_CLK is gated off. |
| 3<br>SDSTB | SD Clock Stable<br>This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON bit in System Control register to remove glitches on the card clock when the frequency is changing.<br>*Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.*<br>0b - Clock is changing frequency and not stable.<br>1b - Clock is stable. |
| 2<br>DLA | Data Line Active<br>This status bit indicates whether one of the DATA lines on the SD Bus is in use.<br>In the case of read transactions:<br>This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register.<br>This bit will be set in either of the following cases:<br>• After the end bit of the read command.<br>• When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer.<br>This bit will be cleared in either of the following cases:<br>(1) When the end bit of the last data block is sent from the SD Bus to the uSDHC.<br>(2) When the Read Wait state is stopped by a Suspend command and the DATA2 line is released.<br>The uSDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait.<br>In the case of write transactions:<br>This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register.<br>This bit will be set in either of the following cases:<br>• After the end bit of the write command.<br>• When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer.<br>This bit will be cleared in either of the following cases:<br>• When the SD card releases Write Busy of the last data block, the uSDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the uSDHC shall assume the card drive "Not Busy".<br>• When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request.<br>In the case of command with busy pending:<br>This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DATA0 line is released. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - DATA Line Inactive<br>1b - DATA Line Active |
| 1<br>CDIHB | Command Inhibit (DATA)<br>This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register.<br>**NOTE:** The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0.<br>0b - Can issue command which uses the DATA line<br>1b - Cannot issue command which uses the DATA line |
| 0<br>CIHB | Command Inhibit (CMD)<br>If this status bit is 0, it indicates that the CMD line is not in use and the uSDHC can issue a SD / MMC Command using the CMD line.<br>This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the uSDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit.<br>0b - Can issue command using only CMD line<br>1b - Cannot issue command |

## 63.8.1.12 Protocol Control (PROT_CTRL)

### 63.8.1.12.1 Offset

| Register | Offset |
|---|---|
| PROT_CTRL | 28h |

### 63.8.1.12.2 Function

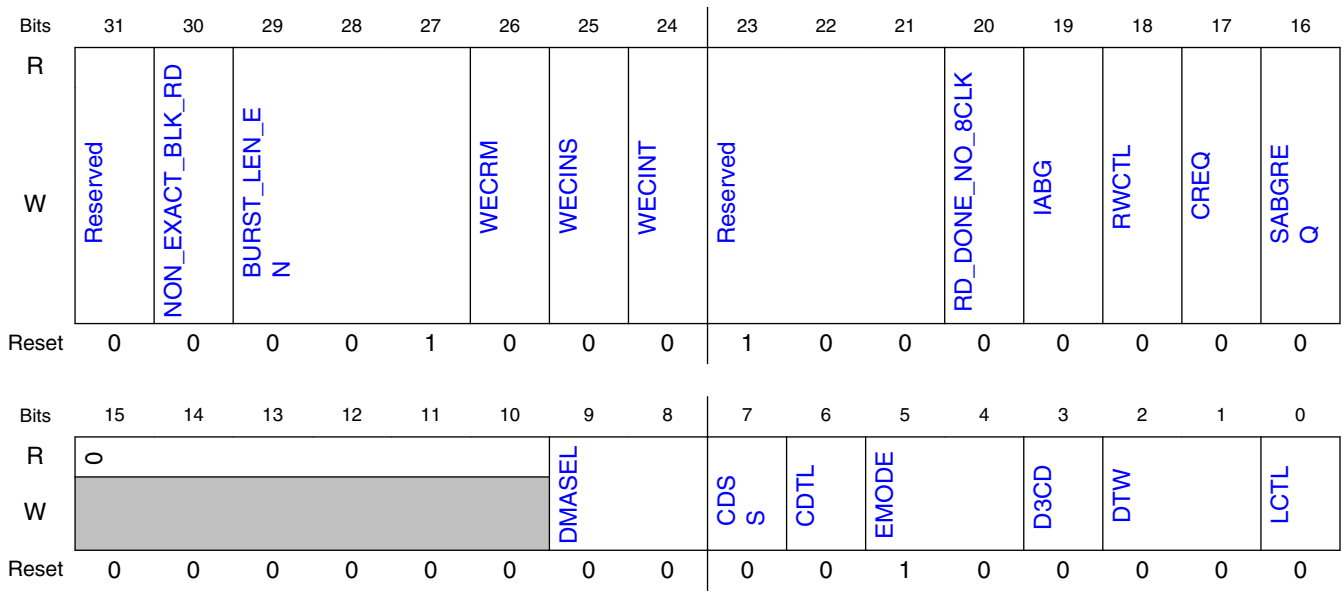
There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the uSDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request shall be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.

3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver shall wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver shall clear the Stop At Block Gap Request before or simultaneously.

### 63.8.1.12.3 Diagram




### 63.8.1.12.4 Fields

| Field | Function |
|---|---|
| 31<br>— | -<br>Reserved. Always write as 0 |
| 30<br>NON_EXACT_BLK_RD | NON_EXACT_BLK_RD<br>Current block read is non-exact block read. It is only used for SDIO.<br>0b - The block read is exact block read. Host driver doesn't need to issue abort command to terminate this multi-block read.<br>1b - The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read. |
| 29-27<br>BURST_LEN_EN | BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP<br>This is used to enable / disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1xxb - Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP<br>x1xb - Burst length is enabled for INCR4 / INCR8 / INCR16<br>xx1b - Burst length is enabled for INCR |
| 26<br>WECRM | Wakeup Event Enable On SD Card Removal<br>This bit enables a wakeup event, via a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Removal Status and the uSDHC interrupt.<br>0b - Disable<br>1b - Enable |
| 25<br>WECINS | Wakeup Event Enable On SD Card Insertion<br>This bit enables a wakeup event, via a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Insertion Status and the uSDHC interrupt.<br>0b - Disable<br>1b - Enable |
| 24<br>WECINT | Wakeup Event Enable On Card Interrupt<br>This bit enables a wakeup event, via a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Interrupt Status and the uSDHC interrupt.<br>0b - Disable<br>1b - Enable |
| 23-21<br>— | -<br>Reserved. Always write as 3'b100 |
| 20<br>RD_DONE_NO_8CLK | RD_DONE_NO_8CLK<br>*Read done no 8 clock:*<br>*According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end bit of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks will be active after the end bit of the last read data transaction.*<br>*However, this 8 clocks should not be active if user wants to use stop at block gap(include the auto stop at block gap in boot mode) feature for read and the RWCTL bit(bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid this 8 clocks. Otherwise, the device may send extra data to uSDHC while uSDHC ignores these data.*<br>*In a summary, this bit should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.* |
| 19<br>IABG | Interrupt At Block Gap<br>This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card.<br>0b - Disabled<br>1b - Enabled |
| 18<br>RWCTL | Read Wait Control |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise the uSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DATA line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the uSDHC will stop the SD Clock to pause reading operation.<br>0b - Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set<br>1b - Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set |
| 17<br>CREQ | Continue Request<br>This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer.<br>The uSDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored.<br>0b - No effect<br>1b - Restart |
| 16<br>SABGREQ | Stop At Block Gap Request<br>This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver shall not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the uSDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver shall not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the uSDHCs behavior is undefined.<br>This bit effects Read Transfer Active, Write Transfer Active, DATA Line Active and Command Inhibit (DATA) in the Present State register.<br>0b - Transfer<br>1b - Stop |
| 15-10<br>— | -<br>Reserved |
| 9-8<br>DMASEL | DMA Select<br>This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation.<br>00b - No DMA or Simple DMA is selected<br>01b - ADMA1 is selected<br>10b - ADMA2 is selected<br>11b - reserved |
| 7<br>CDSS | Card Detect Signal Selection<br>This bit selects the source for the card detection.<br>0b - Card Detection Level is selected (for normal purpose).<br>1b - Card Detection Test Level is selected (for test purpose). |
| 6 | Card Detect Test Level |

*Table continues on the next page...*

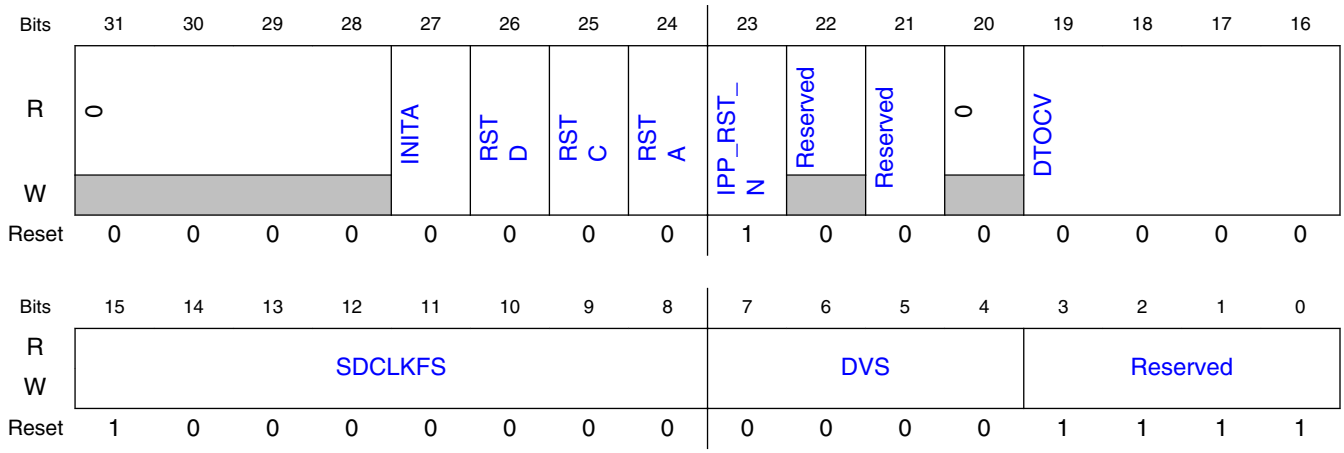
| Field | Function |
|---|---|
| CDTL | This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion.<br>0b - Card Detect Test Level is 0, no card inserted<br>1b - Card Detect Test Level is 1, card inserted |
| 5-4<br>EMODE | Endian Mode<br>The uSDHC supports all three endian modes in data transfer. Refer to Data Buffer for more details.<br>00b - Big Endian Mode<br>01b - Half Word Big Endian Mode<br>10b - Little Endian Mode<br>11b - Reserved |
| 3<br>D3CD | DATA3 as Card Detection Pin<br>If this bit is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the uSDHC does not support.<br>0b - DATA3 does not monitor Card Insertion<br>1b - DATA3 as Card Detection Pin |
| 2-1<br>DTW | Data Transfer Width<br>This bit selects the data width of the SD bus for a data transfer. The Host Driver shall set it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits.<br>00b - 1-bit mode<br>01b - 4-bit mode<br>10b - 8-bit mode<br>11b - Reserved |
| 0<br>LCTL | LED Control<br>This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands.<br>0b - LED off<br>1b - LED on |

## 63.8.1.13 System Control (SYS_CTRL)

### 63.8.1.13.1 Offset

| Register | Offset |
|---|---|
| SYS_CTRL | 2Ch |

## 63.8.1.13.2 Diagram




## 63.8.1.13.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | -<br>Reserved |
| 27<br>INITA | Initialization Active<br>When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (i.e. when command line or data lines are active, write to this bit is not allowed). On the otherhand, when this bit is set, i.e., during intialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self cleared. |
| 26<br>RSTD | Software Reset For DATA Line<br>Only part of the data circuit is reset. DMA circuit is also reset. After this bit is set, SW waits for self-clear.<br>The following registers and bits are cleared by this bit:<br>• Data Port register<br>• Buffer is cleared and initialized.<br>• Present State register<br>• Buffer Read Enable<br>• Buffer Write Enable<br>• Read Transfer Active<br>• Write Transfer Active<br>• DATA Line Active<br>• Command Inhibit (DATA) Protocol Control register<br>• Continue Request<br>• Stop At Block Gap Request Interrupt Status register<br>• Buffer Read Ready<br>• Buffer Write Ready<br>• DMA Interrupt |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • Block Gap Event<br>• Transfer Complete<br><br>**NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC INT_STATUS register is set.<br>0b - No Reset<br>1b - Reset |
| 25<br>RSTC | Software Reset For CMD Line<br>Only part of the command circuit is reset. After this bit is set, SW waits for self-clear.<br>The following registers and bits are cleared by this bit:<br>• Present State register Command Inhibit (CMD)<br>• Interrupt Status register Command Complete<br><br>0b - No Reset<br>1b - Reset |
| 24<br>RSTA | Software Reset For ALL<br>This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver shall set this bit to 1 to reset the uSDHC. The uSDHC shall reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. After this bit is set, SW should wait for self-clear.<br>**NOTE:** When reset, SW must make sure there is no incomplete data transferring. If there is data transfer going on, SW need wait TC or DC INT_STATUS register is set.<br>0b - No Reset<br>1b - Reset |
| 23<br>IPP_RST_N | IPP_RST_N<br>This register's value will be output to CARD from pad directly for hardware reset of the card if the card supports this feature. |
| 22<br>— | -<br>Reserved |
| 21<br>— | -<br>Reserved |
| 20<br>— | -<br>Reserved |
| 19-16<br>DTOCV | Data Timeout Counter Value<br>This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value.<br>The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events.<br>0000b - SDCLK x $2^{14}$<br>0001b - SDCLK x $2^{15}$<br>1101b - SDCLK x $2^{27}$<br>1110b - SDCLK x $2^{28}$<br>1111b - SDCLK x $2^{29}$ |
| 15-8<br>SDCLKFS | SDCLK Frequency Select |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register.<br>*In Single Data Rate mode(DDR_EN bit of MIXERCTRL is '0')*<br>Only the following settings are allowed:<br>80h) Base clock divided by 256<br>40h) Base clock divided by 128<br>20h) Base clock divided by 64<br>10h) Base clock divided by 32<br>08h) Base clock divided by 16<br>04h) Base clock divided by 8<br>02h) Base clock divided by 4<br>01h) Base clock divided by 2<br>00h) Base clock divided by 1<br>*While in Dual Data Rate mode(DDR_EN bit of MIXERCTRL is '1')*<br>*Only the following settings are allowed:*<br>*80h) Base clock divided by 512*<br>*40h) Base clock divided by 256*<br>*20h) Base clock divided by 128*<br>*10h) Base clock divided by 64*<br>*08h) Base clock divided by 32*<br>*04h) Base clock divided by 16*<br>*02h) Base clock divided by 8*<br>*01h) Base clock divided by 4*<br>*00h) Base clock divided by 2*<br>*When S/W changes the DDR_EN bit, SDCLKFS may need to be changed also!*<br>In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD Clock.<br>Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits.<br>The frequency of SDCLK is set by the following formula:<br>Clock Frequency = (Base Clock) / (prescaler x divisor)<br>For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz.<br>The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz.<br>According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and shall never exceed this limit.<br>*Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | *If setting SDCLKFS and DVS can generate same clock frequency,(For example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.* |
| 7-4<br>DVS | Divisor<br>This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle.<br>*Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.*<br>The setting are as following:<br>0000b - Divide-by-1<br>0001b - Divide-by-2<br>1110b - Divide-by-15<br>1111b - Divide-by-16 |
| 3-0<br>— | -<br>Reserved. Always write as 1. |

## 63.8.1.14 Interrupt Status (INT_STATUS)

### 63.8.1.14.1 Offset

| Register | Offset |
|---|---|
| INT_STATUS | 30h |

### 63.8.1.14.2 Function

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the Command Timeout Error and the Command Complete.

**Table 63-8. uSDHC Status for Command Timeout Error/Command Complete Bit Combinations**

| Command Complete | Command Timeout Error | Meaning of the Status |
|---|---|---|
| 0 | 0 | X |

*Table continues on the next page...*

**Table 63-8. uSDHC Status for Command Timeout Error/Command Complete Bit Combinations (continued)**

| Command Complete | Command Timeout Error | Meaning of the Status |
|---|---|---|
| X | 1 | Response not received within 64 SDCLK cycles |
| 1 | 0 | Response received |

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

**Table 63-9. uSDHC Status for Data Timeout Error/Transfer Complete Bit Combinations**

| Transfer Complete | Data Timeout Error | Meaning of the Status |
|---|---|---|
| 0 | 0 | X |
| 0 | 1 | Timeout occurred during transfer |
| 1 | X | Data Transfer Complete |

The table below shows the relationship between the Command CRC Error and Command Timeout Error.

**Table 63-10. uSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations**

| Command Complete | Command Timeout Error | Meaning of the Status |
|---|---|---|
| 0 | 0 | No error |
| 0 | 1 | Response Timeout Error |
| 1 | 0 | Response CRC Error |
| 1 | 1 | CMD line conflict |

## 63.8.1.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DMAE | 0 | 0 | 0 | AC12E | 0 | DEBE | DCE | DTOE | CIE | CEBE | CCE | CTOE |
| W | | | | W1C | | | | W1C | | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | | | CINT | CRM | CINS | BRR | BWR | DINT | BGE | TC | CC |
| W | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 63.8.1.14.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | -<br>Reserved |
| 28<br>DMAE | DMA Error<br>Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the Host Driver shall re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size.<br>0b - No Error<br>1b - Error |
| 27<br>— | -<br>Reserved |
| 26<br>— | Reserved. |
| 25<br>— | -<br>Reserved |
| 24<br>AC12E | Auto CMD12 Error<br>Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error.<br>0b - No Error<br>1b - Error |
| 23<br>— | -<br>Reserved |
| 22<br>DEBE | Data End Bit Error<br>Occurs either when detecting 0 at the end bit position of read data, which uses the DATA line, or at the end bit position of the CRC.<br>0b - No Error<br>1b - Error |
| 21<br>DCE | Data CRC Error<br>Occurs when detecting a CRC error when transferring read data, which uses the DATA line, or when detecting the Write CRC status having a value other than 010.<br>0b - No Error<br>1b - Error |
| 20<br>DTOE | Data Timeout Error<br>Occurs when detecting one of following time-out conditions.<br>• Busy time-out for R1b, R5b type<br>• Busy time-out after Write CRC status<br>• Read Data time-out. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - No Error<br>1b - Time out |
| 19<br>CIE | Command Index Error<br>Occurs if a Command Index error occurs in the command response.<br>0b - No Error<br>1b - Error |
| 18<br>CEBE | Command End Bit Error<br>Occurs when detecting that the end bit of a command response is 0.<br>0b - No Error<br>1b - End Bit Error Generated |
| 17<br>CCE | Command CRC Error<br>Command CRC Error is generated in two cases.<br>• If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response.<br>• The uSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the uSDHC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict.<br>0b - No Error<br>1b - CRC Error Generated. |
| 16<br>CTOE | Command Timeout Error<br>Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the uSDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set (as shown in Interrupt Status (INT_STATUS) ), this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the uSDHC.<br>0b - No Error<br>1b - Time out |
| 15<br>— | -<br>Reserved |
| 14<br>— | -<br>Reserved |
| 13<br>— | -<br>Reserved |
| 12<br>— | -<br>Reserved |
| 11-9<br>— | -<br>Reserved |
| 8<br>CINT | Card Interrupt<br>This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the uSDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing this bit to 1 can clear this bit, but as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | When this status has been set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again.<br>0b - No Card Interrupt<br>1b - Generate Card Interrupt |
| 7<br>CRM | Card Removal<br>This status bit is set if the Card Inserted bit in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register.<br>0b - Card state unstable or inserted<br>1b - Card removed |
| 6<br>CINS | Card Insertion<br>This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register.<br>0b - Card state unstable or removed<br>1b - Card inserted |
| 5<br>BRR | Buffer Read Ready<br>This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information.<br>0b - Not ready to read buffer<br>1b - Ready to read buffer |
| 4<br>BWR | Buffer Write Ready<br>This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information.<br>0b - Not ready to write buffer<br>1b - Ready to write buffer: |
| 3<br>DINT | DMA Interrupt<br>Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set.<br>0b - No DMA Interrupt<br>1b - DMA Interrupt is generated |
| 2<br>BGE | Block Gap Event<br>If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1.<br>In the case of a Read Transaction: This bit is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing).<br>0b - No block gap event<br>1b - Transaction stopped at block gap |
| 1<br>TC | Transfer Complete<br>This bit is set when a read or write transfer is completed.<br>In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System).<br>In the case of a Write Transaction: This bit is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released).<br>In the case of a command with busy, this bit is set when busy is deasserted.<br>0b - Transfer not complete<br>1b - Transfer complete |
| 0<br>CC | Command Complete<br>This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register.<br>0b - Command not complete<br>1b - Command complete |

## 63.8.1.15 Interrupt Status Enable (INT_STATUS_EN)

### 63.8.1.15.1 Offset

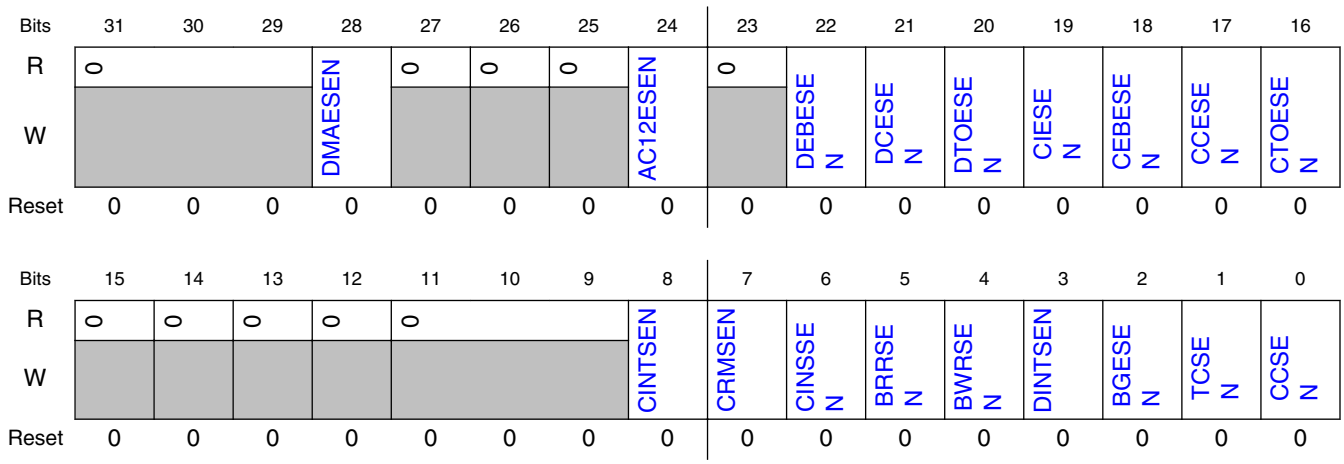
| Register | Offset |
|---|---|
| INT_STATUS_EN | 34h |

### 63.8.1.15.2 Function

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (i.e. when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, uSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

### 63.8.1.15.3 Diagram




### 63.8.1.15.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | -<br>Reserved |
| 28<br>DMAESEN | DMA Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 27<br>— | -<br>Reserved |
| 26<br>— | -<br>Reserved |
| 25<br>— | -<br>Reserved |
| 24<br>AC12ESEN | Auto CMD12 Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 23<br>— | -<br>Reserved |
| 22 | Data End Bit Error Status Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DEBESEN | 0b - Masked<br>1b - Enabled |
| 21<br>DCESEN | Data CRC Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 20<br>DTOESEN | Data Timeout Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 19<br>CIESEN | Command Index Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 18<br>CEBESEN | Command End Bit Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 17<br>CCESEN | Command CRC Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 16<br>CTOESEN | Command Timeout Error Status Enable<br>0b - Masked<br>1b - Enabled |
| 15<br>— | -<br>Reserved |
| 14<br>— | -<br>Reserved |
| 13<br>— | -<br>Reserved |
| 12<br>— | -<br>Reserved |
| 11-9<br>— | -<br>Reserved |
| 8<br>CINTSEN | Card Interrupt Status Enable<br>If this bit is set to 0, the uSDHC will clear the interrupt request to the system. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.<br>0b - Masked<br>1b - Enabled |
| 7<br>CRMSEN | Card Removal Status Enable<br>0b - Masked<br>1b - Enabled |
| 6<br>CINSSEN | Card Insertion Status Enable<br>0b - Masked<br>1b - Enabled |
| 5<br>BRRSEN | Buffer Read Ready Status Enable<br>0b - Masked<br>1b - Enabled |
| 4 | Buffer Write Ready Status Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BWRSEN | 0b - Masked<br>1b - Enabled |
| 3<br>DINTSEN | DMA Interrupt Status Enable<br>0b - Masked<br>1b - Enabled |
| 2<br>BGESEN | Block Gap Event Status Enable<br>0b - Masked<br>1b - Enabled |
| 1<br>TCSEN | Transfer Complete Status Enable<br>0b - Masked<br>1b - Enabled |
| 0<br>CCSEN | Command Complete Status Enable<br>0b - Masked<br>1b - Enabled |

## 63.8.1.16 Interrupt Signal Enable (INT_SIGNAL_EN)

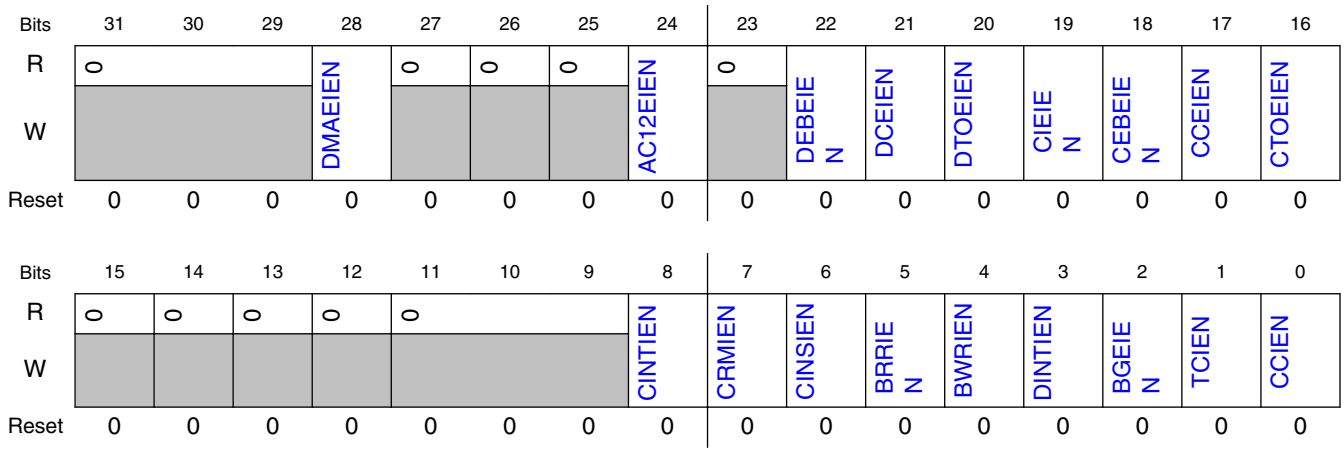
### 63.8.1.16.1 Offset

| Register | Offset |
|---|---|
| INT_SIGNAL_EN | 38h |

### 63.8.1.16.2 Function

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

## 63.8.1.16.3 Diagram




## 63.8.1.16.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | -<br>Reserved |
| 28<br>DMAEIEN | DMA Error Interrupt Enable<br>0b - Masked<br>1b - Enable |
| 27<br>— | -<br>Reserved |
| 26<br>— | -<br>Reserved |
| 25<br>— | -<br>Reserved |
| 24<br>AC12EIEN | Auto CMD12 Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 23<br>— | -<br>Reserved |
| 22<br>DEBEIEN | Data End Bit Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 21<br>DCEIEN | Data CRC Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 20<br>DTOEIEN | Data Timeout Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 19 | Command Index Error Interrupt Enable<br>0b - Masked |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CIEIEN | 1b - Enabled |
| 18<br>CEBEIEN | Command End Bit Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 17<br>CCEIEN | Command CRC Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 16<br>CTOEIEN | Command Timeout Error Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 15<br>— | -<br>Reserved |
| 14<br>— | -<br>Reserved |
| 13<br>— | -<br>Reserved |
| 12<br>— | -<br>Reserved |
| 11-9<br>— | -<br>Reserved |
| 8<br>CINTIEN | Card Interrupt Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 7<br>CRMIEN | Card Removal Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 6<br>CINSIEN | Card Insertion Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 5<br>BRRIEN | Buffer Read Ready Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 4<br>BWRIEN | Buffer Write Ready Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 3<br>DINTIEN | DMA Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 2<br>BGEIEN | Block Gap Event Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 1<br>TCIEN | Transfer Complete Interrupt Enable<br>0b - Masked<br>1b - Enabled |
| 0<br>CCIEN | Command Complete Interrupt Enable<br>0b - Masked<br>1b - Enabled |

## 63.8.1.17 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

### 63.8.1.17.1 Offset

| Register | Offset |
|---|---|
| AUTOCMD12_ERR_STATUS | 3Ch |

### 63.8.1.17.2 Function

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver shall check this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in bit 04-01.This register is valid only when the Auto CMD12 Error status bit is set.

The table bwlow shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

**Table 63-11. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12**

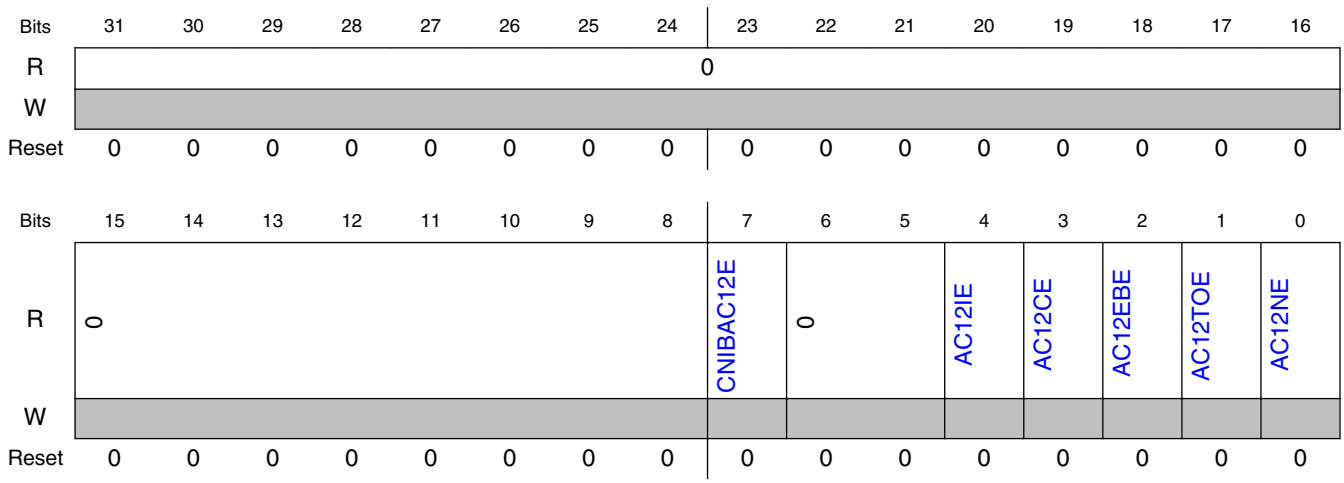
| Auto CMD12 CRC Error | Auto CMD12 Timeout Error | Type of Error |
|---|---|---|
| 0 | 0 | No Error |
| 0 | 1 | Response Timeout Error |
| 1 | 0 | Response CRC Error |
| 1 | 1 | CMD line conflict |

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the uSDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that can't be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the Command register. So it is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

### 63.8.1.17.3 Diagram




### 63.8.1.17.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | -<br>Reserved |
| 7<br>CNIBAC12E | Command Not Issued By Auto CMD12 Error<br>Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.<br>0b - No error<br>1b - Not Issued |
| 6-5<br>— | -<br>Reserved |
| 4<br>AC12IE | Auto CMD12 / 23 Index Error<br>Occurs if the Command Index error occurs in response to a command.<br>0b - No error<br>1b - Error, the CMD index in response is not CMD12/23 |
| 3<br>AC12CE | Auto CMD12 / 23 CRC Error<br>Occurs when detecting a CRC error in the command response.<br>0b - No CRC error |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - CRC Error Met in Auto CMD12/23 Response |
| 2<br>AC12EBE | Auto CMD12 / 23 End Bit Error<br>Occurs when detecting that the end bit of command response is 0 which should be 1.<br>0b - No error<br>1b - End Bit Error Generated |
| 1<br>AC12TOE | Auto CMD12 / 23 Timeout Error<br>Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning.<br>0b - No error<br>1b - Time out |
| 0<br>AC12NE | Auto CMD12 Not Executed<br>If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning.<br>0b - Executed<br>1b - Not executed |

## 63.8.1.18 Host Controller Capabilities (HOST_CTRL_CAP)

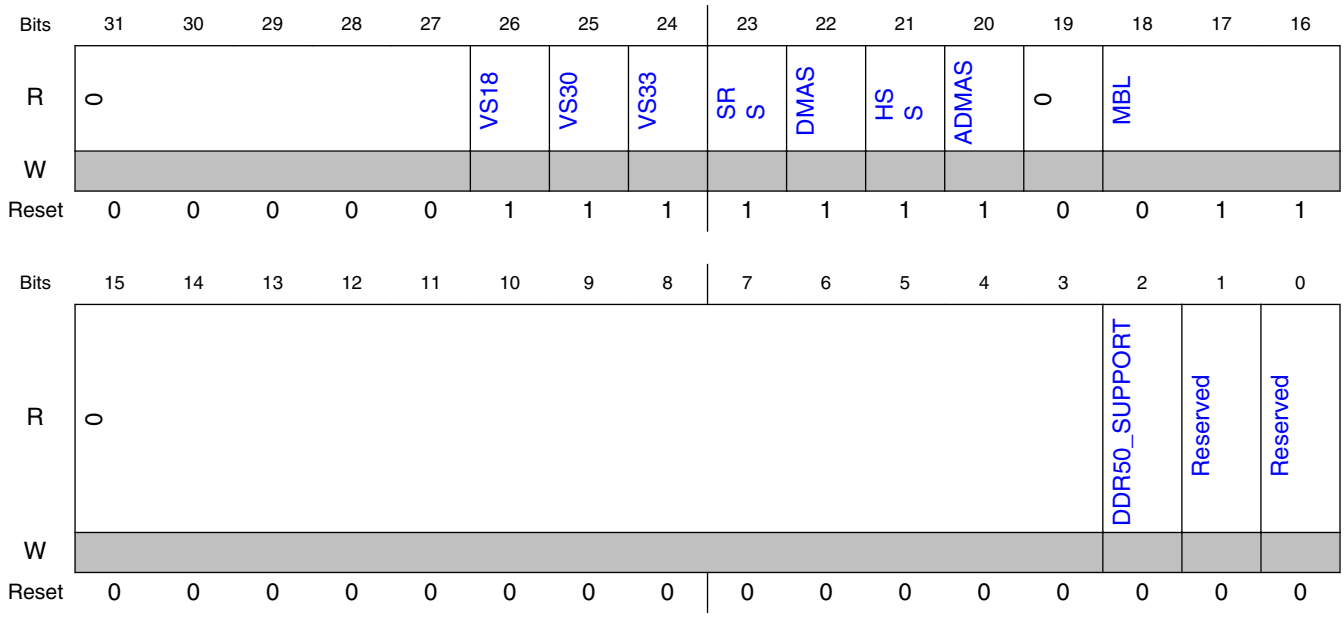
### 63.8.1.18.1 Offset

| Register | Offset |
|---|---|
| HOST_CTRL_CAP | 40h |

### 63.8.1.18.2 Function

This register provides the Host Driver with information specific to the uSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset.

## 63.8.1.18.3 Diagram




## 63.8.1.18.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | -<br>Reserved |
| 26<br>VS18 | Voltage Support 1.8 V<br>This bit shall depend on the Host System ability.<br>0b - 1.8V not supported<br>1b - 1.8V supported |
| 25<br>VS30 | Voltage Support 3.0 V<br>This bit shall depend on the Host System ability.<br>0b - 3.0V not supported<br>1b - 3.0V supported |
| 24<br>VS33 | Voltage Support 3.3V<br>This bit shall depend on the Host System ability.<br>0b - 3.3V not supported<br>1b - 3.3V supported |
| 23<br>SRS | Suspend / Resume Support<br>This bit indicates whether the uSDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver shall not issue either Suspend or Resume commands.<br>0b - Not supported<br>1b - Supported |
| 22 | DMA Support |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DMAS | This bit indicates whether the uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly.<br>0b - DMA not supported<br>1b - DMA Supported |
| 21<br>HSS | High Speed Support<br>This bit indicates whether the uSDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz.<br>0b - High Speed Not Supported<br>1b - High Speed Supported |
| 20<br>ADMAS | ADMA Support<br>This bit indicates whether the uSDHC supports the ADMA feature.<br>0b - Advanced DMA Not supported<br>1b - Advanced DMA Supported |
| 19<br>— | -<br>Reserved |
| 18-16<br>MBL | Max Block Length<br>This value indicates the maximum block size that the Host Driver can read and write to the buffer in the uSDHC. The buffer shall transfer block size without wait cycles.<br>000b - 512 bytes<br>001b - 1024 bytes<br>010b - 2048 bytes<br>011b - 4096 bytes |
| 15-3<br>— | -<br>Reserved |
| 2<br>DDR50_SUPPORT | DDR50 support<br>This bit indicates support of DDR50 mode. |
| 1<br>— | - |
| 0<br>— | - |

## 63.8.1.19 Watermark Level (WTMK_LVL)

### 63.8.1.19.1 Offset

| Register | Offset |
|---|---|
| WTMK_LVL | 44h |

### 63.8.1.19.2 Function

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. There value can range from 1 to 31 words.

### 63.8.1.19.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WR_BRST_LEN | | | | | WR_WML | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | RD_BRST_LEN | | | | | RD_WML | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### 63.8.1.19.4 Fields

| Field | Function |
|---|---|
| 31-29<br>— | -<br>Reserved |
| 28-24<br>WR_BRST_LEN | Write Burst Length Due to system restriction, the actual burst length may not exceed 16.<br>The number of words the uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field). |
| 23-16<br>WR_WML | Write Watermark Level<br>The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128. |
| 15-13<br>— | -<br>Reserved |
| 12-8<br>RD_BRST_LEN | Read Burst Length Due to system restriction, the actual burst length may not exceed 16.<br>The number of words the uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field). |
| 7-0<br>RD_WML | Read Watermark Level<br>The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128. |

## 63.8.1.20 Mixer Control (MIX_CTRL)

### 63.8.1.20.1 Offset

| Register | Offset |
|---|---|
| MIX_CTRL | 48h |

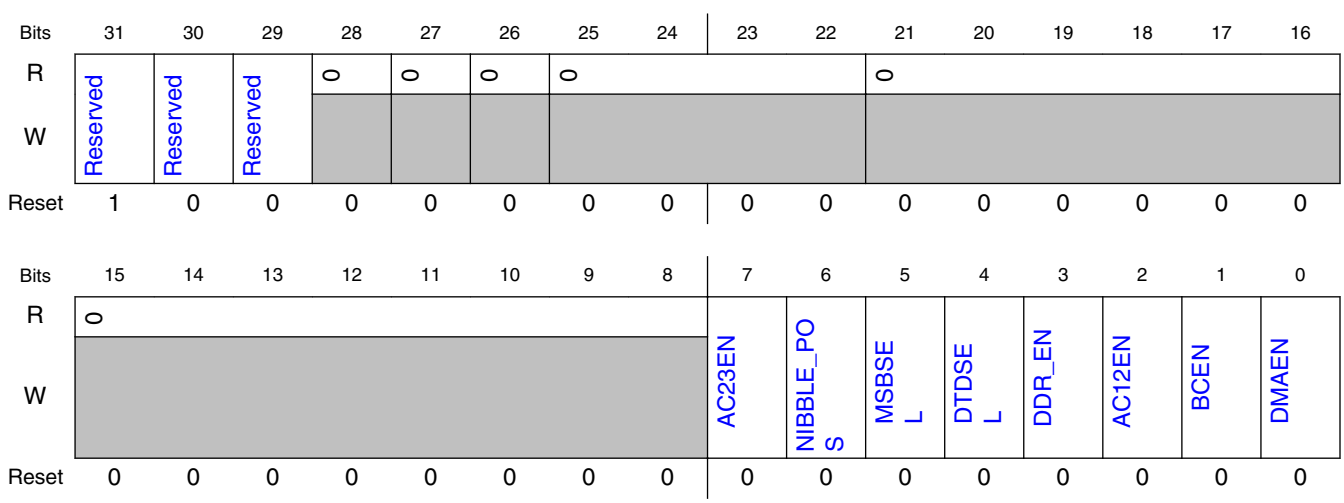
### 63.8.1.20.2 Function

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing this register. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

**Table 63-12. Transfer Type Register Setting for Various Transfer Types**

| Multi/Single Block Select | Block Count Enable | Block Count | Function |
|---|---|---|---|
| 0 | Don't Care | Don't Care | Single Transfer |
| 1 | 0 | Don't Care | Infinite Transfer |
| 1 | 1 | Positive Number | Multiple Transfer |
| 1 | 1 | Zero | No Data Transfer |

### 63.8.1.20.3 Diagram



## 63.8.1.20.4 Fields

| Field | Function |
|---|---|
| 31<br>— | -<br>Reserved. Always write as 1 |
| 30<br>— | -<br>Reserved. Always write as 0. |
| 29<br>— | -<br>Reserved. Always write as 0. |
| 28<br>— | -<br>Reserved |
| 27<br>— | -<br>Reserved |
| 26<br>— | -<br>Reserved |
| 25-22<br>— | -<br>Reserved |
| 21-8<br>— | -<br>Reserved |
| 7<br>AC23EN | Auto CMD23 Enable<br>When this bit is set to 1, the Host Controller issues a CMD23 automatically before issuing a command specified in the Command Register. |
| 6<br>NIBBLE_POS | NIBBLE_POS<br>In DDR 4-bit mode nibble position indictation. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'. |
| 5<br>MSBSEL | Multi / Single Block Select<br>This bit enables multiple block DATA line data transfers. For any other commands, this bit can be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (Refer to Command Transfer Type (CMD_XFR_TYP) ).<br>0b - Single Block<br>1b - Multiple Blocks |
| 4<br>DTDSEL | Data Transfer Direction Select<br>This bit defines the direction of DATA line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the uSDHC and is set to 0 for all other commands.<br>0b - Write (Host to Card)<br>1b - Read (Card to Host) |
| 3<br>DDR_EN | Dual Data Rate mode selection |
| 2<br>AC12EN | Auto CMD12 Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the uSDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the uSDHC will ignore this bit no matter it is set or not.<br>0b - Disable<br>1b - Enable |
| 1<br>BCEN | Block Count Enable<br>This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer.<br>0b - Disable<br>1b - Enable |
| 0<br>DMAEN | DMA Enable<br>This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA or the Advanced DMA is active depends on the DMA Select field of the Protocol Control register.<br>0b - Disable<br>1b - Enable |

## 63.8.1.21 Force Event (FORCE_EVENT)

### 63.8.1.21.1 Offset

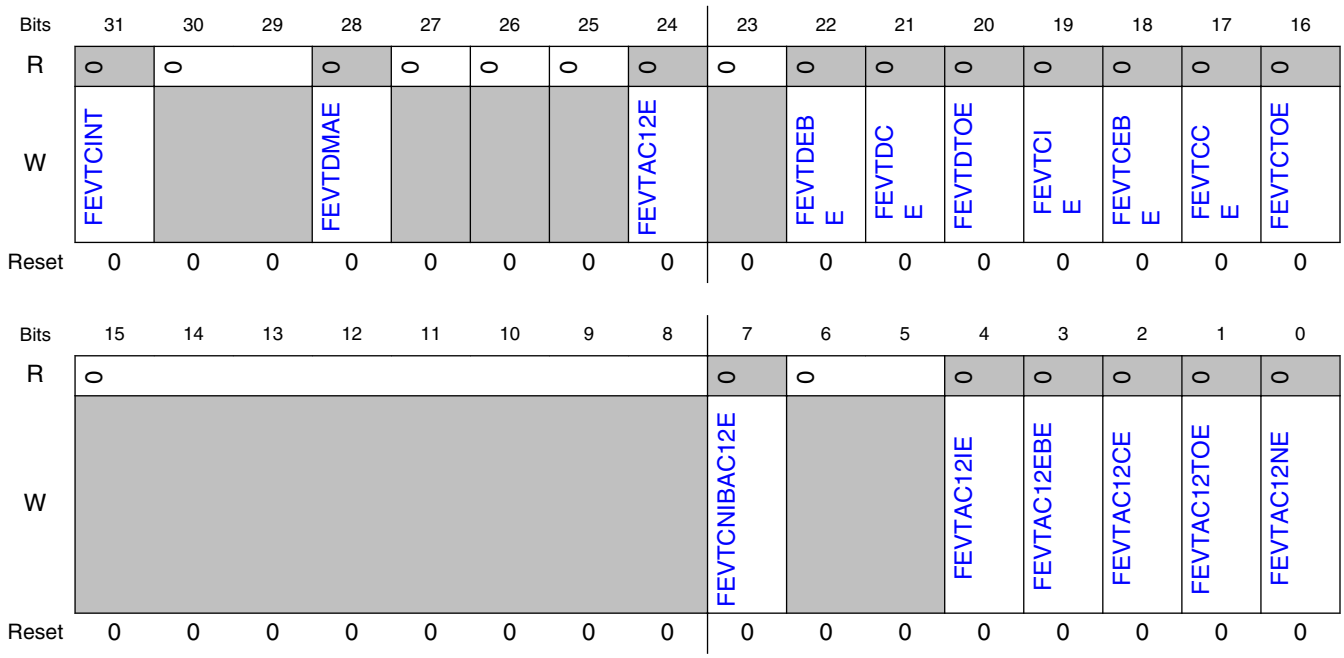
| Register | Offset |
|---|---|
| FORCE_EVENT | 50h |

### 63.8.1.21.2 Function

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, make sure to set IPGEN bit in System Control Register so that IPG_CLK is always active.

Forcing a card interrupt will generate a short pulse on the DATA1 line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is self cleared.

## 63.8.1.21.3 Diagram




## 63.8.1.21.4 Fields

| Field | Function |
|---|---|
| 31<br>FEVTCINT | Force Event Card Interrupt<br>Writing 1 to this bit generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card. |
| 30-29<br>— | -<br>Reserved |
| 28<br>FEVTDMAE | Force Event DMA Error<br>Forces the DMAE bit of Interrupt Status Register to be set. |
| 27<br>— | -<br>Reserved |
| 26<br>— | -<br>Reserved |
| 25<br>— | -<br>Reserved |
| 24<br>FEVTAC12E | Force Event Auto Command 12 Error<br>Forces the AC12E bit of Interrupt Status Register to be set. |
| 23<br>— | -<br>Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 22<br>FEVTDEBE | Force Event Data End Bit Error<br>Forces the DEBE bit of Interrupt Status Register to be set. |
| 21<br>FEVTDCE | Force Event Data CRC Error<br>Forces the DCE bit of Interrupt Status Register to be set. |
| 20<br>FEVTDTOE | Force Event Data Time Out Error<br>Force the DTOE bit of Interrupt Status Register to be set. |
| 19<br>FEVTCIE | Force Event Command Index Error<br>Forces the CCE bit of Interrupt Status Register to be set. |
| 18<br>FEVTCEBE | Force Event Command End Bit Error<br>Forces the CEBE bit of Interrupt Status Register to be set. |
| 17<br>FEVTCCE | Force Event Command CRC Error<br>Forces the CCE bit of Interrupt Status Register to be set. |
| 16<br>FEVTCTOE | Force Event Command Time Out Error<br>Forces the CTOE bit of Interrupt Status Register to be set. |
| 15-8<br>— | -<br>Reserved |
| 7<br>FEVTCNIBAC12E | Force Event Command Not Executed By Auto Command 12 Error<br>Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set. |
| 6-5<br>— | -<br>Reserved |
| 4<br>FEVTAC12IE | Force Event Auto Command 12 Index Error<br>Forces the AC12IE bit in the Auto Command12 Error Status Register to be set. |
| 3<br>FEVTAC12EBE | Force Event Auto Command 12 End Bit Error<br>Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set. |
| 2<br>FEVTAC12CE | Force Event Auto Command 12 CRC Error<br>Forces the AC12CE bit in the Auto Command12 Error Status Register to be set. |
| 1<br>FEVTAC12TOE | Force Event Auto Command 12 Time Out Error<br>Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set. |
| 0<br>FEVTAC12NE | Force Event Auto Command 12 Not Executed<br>Forces the AC12NE bit in the Auto Command12 Error Status Register to be set. |

## 63.8.1.22 ADMA Error Status Register (ADMA_ERR_STATUS)

### 63.8.1.22.1 Offset

| Register | Offset |
|---|---|
| ADMA_ERR_STATUS | 54h |

### 63.8.1.22.2 Function

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

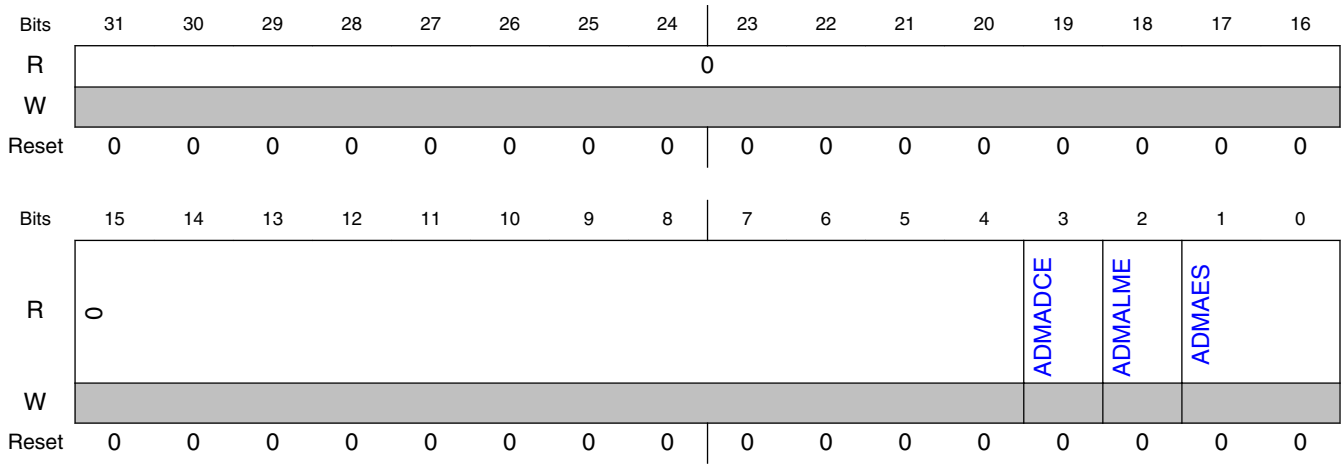
In case of a write operation, the Host Driver should use the ACMD22 to get the number of the written block, rather than using this information, since unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

**Table 63-13. ADMA Error State Coding**

| D01-D00 | ADMA Error State (when error has occurred) | Contents of ADMA System Address Register |
|---|---|---|
| 00 | ST_STOP (Stop DMA) | Holds the address of the next executable Descriptor command |
| 01 | ST_FDS (Fetch Descriptor) | Holds the valid Descriptor address |
| 10 | ST_CADR (Change Address) | No ADMA Error is generated |
| 11 | ST_TFR (Transfer Data) | Holds the address of the next executable Descriptor command |

### 63.8.1.22.3 Diagram




### 63.8.1.22.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | -<br>Reserved |
| 3<br>ADMADCE | ADMA Descriptor Error<br>This error occurs when invalid descriptor fetched by ADMA.<br>0b - No Error<br>1b - Error |
| 2<br>ADMALME | ADMA Length Mismatch Error<br>This error occurs in the following 2 cases:<br>• While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length.<br>• Total data length cannot be divided by the block length.<br>0b - No Error<br>1b - Error |
| 1-0<br>ADMAES | ADMA Error State (when ADMA Error is occurred)<br>This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (ADMA_ERR_STATUS) for more details. |

## 63.8.1.23 ADMA System Address (ADMA_SYS_ADDR)

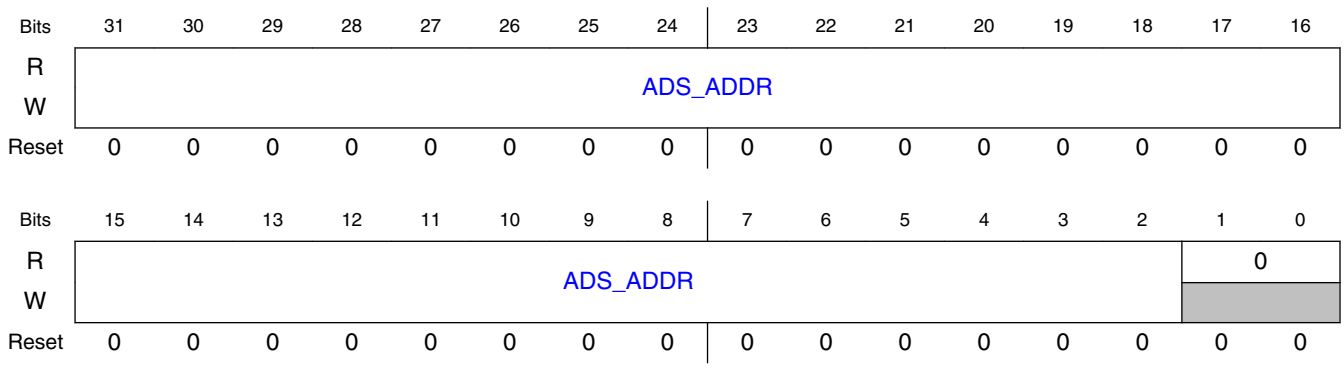
### 63.8.1.23.1 Offset

| Register | Offset |
|---|---|
| ADMA_SYS_ADDR | 58h |

### 63.8.1.23.2 Function

This register contains the physical system memory address used for ADMA transfers.

### 63.8.1.23.3 Diagram




### 63.8.1.23.4 Fields

| Field | Function |
|---|---|
| 31-2<br>ADS_ADDR | ADMA System Address<br>This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register shall hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned.<br>Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions . |
| 1-0<br>— | -<br>Reserved |

## 63.8.1.24 Vendor Specific Register (VEND_SPEC)

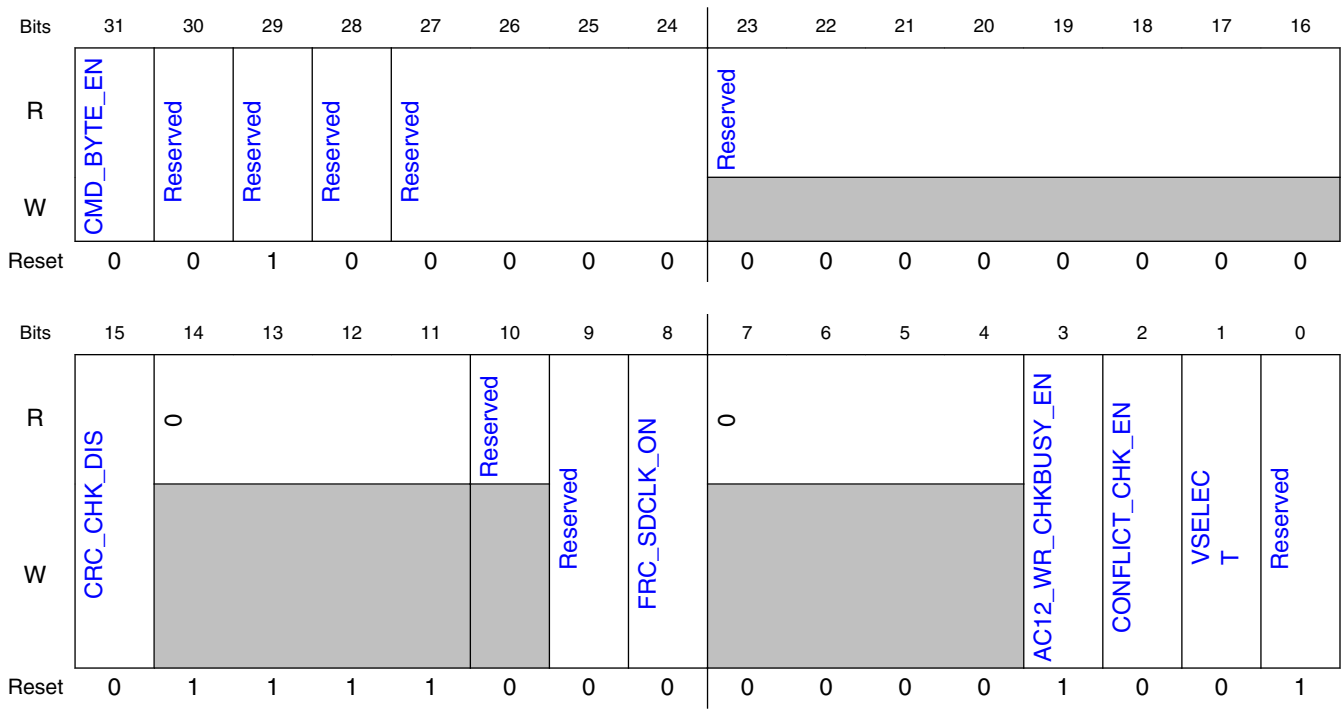
### 63.8.1.24.1 Offset

| Register | Offset |
|---|---|
| VEND_SPEC | C0h |

### 63.8.1.24.2 Function

This register contains the vendor specific control / status register.

### 63.8.1.24.3 Diagram




### 63.8.1.24.4 Fields

| Field | Function |
|---|---|
| 31<br>CMD_BYTE_EN | CMD_BYTE_EN<br>Byte access<br>0b - Disable<br>1b - Enable |
| 30<br>— | -<br>Reserved. Always write as 0. |
| 29 | - |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | Reserved. Always write as 1. |
| 28<br>— | -<br>Reserved. Always write as 0. |
| 27-24<br>— | -<br>Reserved. Always write as 4'b0000. |
| 23-16<br>— | -<br>Reserved. Always write as 8'h00. |
| 15<br>CRC_CHK_DIS | CRC Check Disable<br>0b - Check CRC16 for every read data packet and check CRC bits for every write data packet<br>1b - Ignore CRC16 check for every read data packet and ignore CRC bits check for every write data packet |
| 14-11<br>— | -<br>Reserved |
| 10<br>— | -<br>Reserved. Always write as 0. |
| 9<br>— | -<br>Reserved. Always write as 0. |
| 8<br>FRC_SDCLK_ON | FRC_SDCLK_ON<br>Force CLK output active<br>0b - CLK active or inactive is fully controlled by the hardware.<br>1b - Force CLK active. |
| 7-4<br>— | -<br>Reserved |
| 3<br>AC12_WR_CHKBUSY_EN | AC12_WR_CHKBUSY_EN<br>Check busy enable after auto CMD12 for write data packet<br>0b - Do not check busy after auto CMD12 for write data packet<br>1b - Check busy after auto CMD12 for write data packet |
| 2<br>CONFLICT_CHK_EN | Conflict check enable.<br>It is not implemented in uSDHC IP.<br>0b - Conflict check disable<br>1b - Conflict check enable |
| 1<br>VSELECT | Voltage Selection<br>Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads.<br>0b - Change the voltage to high voltage range, around 3.0 V<br>1b - Change the voltage to low voltage range, around 1.8 V |
| 0<br>— | -<br>Reserved. Always write as 0. |

# 63.8.1.25 MMC Boot Register (MMC_BOOT)

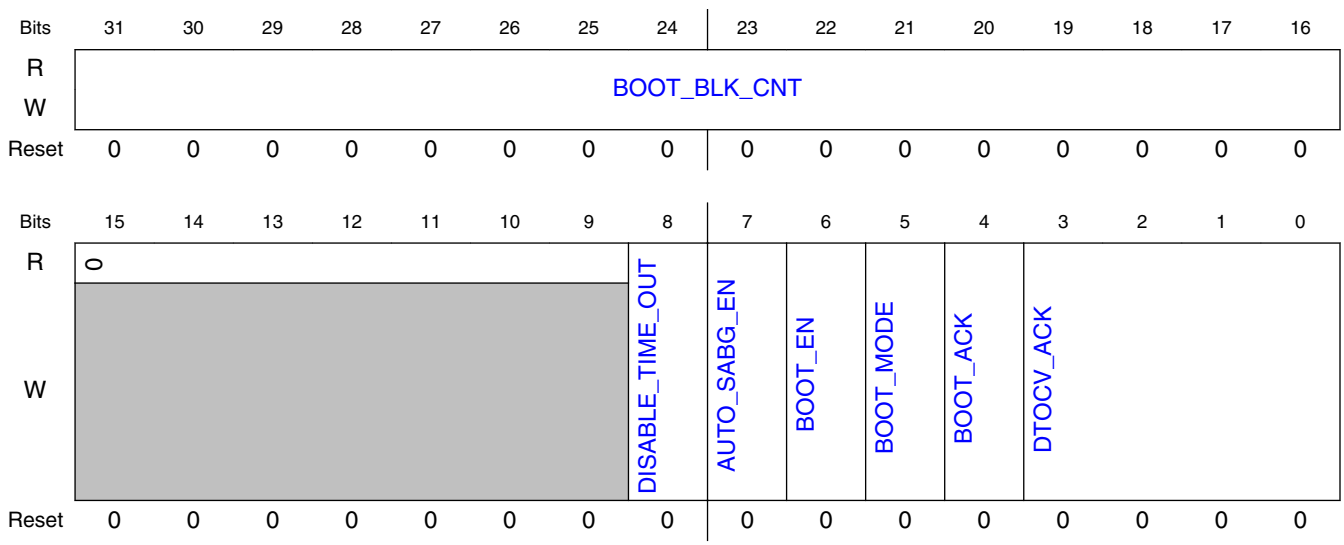
## 63.8.1.25.1 Offset

| Register | Offset |
|---|---|
| MMC_BOOT | C4h |

## 63.8.1.25.2 Function

This register contains the MMC Fast Boot control register.

## 63.8.1.25.3 Diagram




## 63.8.1.25.4 Fields

| Field | Function |
|---|---|
| 31-16<br>BOOT_BLK_CNT | BOOT_BLK_CNT<br>The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap.<br>Here, BLK_CNT is defined in the Block Atrributes Register, bit31 - 16 of 0x04. |
| 15-9<br>— | -<br>Reserved |
| 8<br>DISABLE_TIME_OUT | Disable Time Out<br>**NOTE:** When this bit is set, there is no timeout check no matter whether BOOT_EN is set or not.<br>0b - Enable time out |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Disable time out |
| 7<br>AUTO_SABG_EN | AUTO_SABG_EN<br>During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT). |
| 6<br>BOOT_EN | BOOT_EN<br>Boot mode enable<br>0b - Fast boot disable<br>1b - Fast boot enable |
| 5<br>BOOT_MODE | BOOT_MODE<br>Boot mode select<br>0b - Normal boot<br>1b - Alternative boot |
| 4<br>BOOT_ACK | BOOT_ACK<br>Boot ACK mode select<br>0b - No ack<br>1b - Ack |
| 3-0<br>DTOCV_ACK | DTOCV_ACK<br>Boot ACK time out counter value.<br>0000b - SDCLK x 2^14<br>0001b - SDCLK x 2^15<br>0010b - SDCLK x 2^16<br>0011b - SDCLK x 2^17<br>0100b - SDCLK x 2^18<br>0101b - SDCLK x 2^19<br>0110b - SDCLK x 2^20<br>0111b - SDCLK x 2^21<br>1110b - SDCLK x 2^28<br>1111b - SDCLK x 2^29 |

## 63.8.1.26 Vendor Specific 2 Register (VEND_SPEC2)

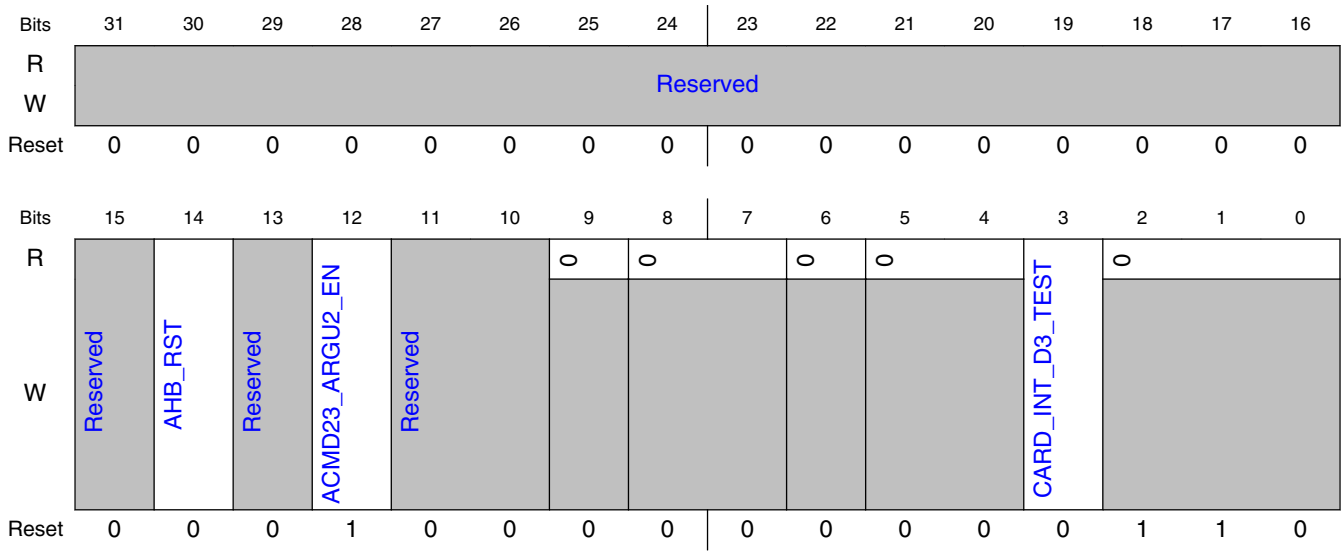
### 63.8.1.26.1 Offset

| Register | Offset |
|---|---|
| VEND_SPEC2 | C8h |

### 63.8.1.26.2 Function

This register contains the vendor specific control 2 register.

## 63.8.1.26.3 Diagram




## 63.8.1.26.4 Fields

| Field | Function |
|---|---|
| 31-15<br>— | -<br>Reserved |
| 14<br>AHB_RST | AHB BUS reset<br>Reset internal AHB bus logic before RST_ALL or RST_DATA to avoid AHB bus hang. |
| 13<br>— | -<br>Reserved |
| 12<br>ACMD23_ARGU2_EN | Argument2 register enable for ACMD23<br>0b - Disable<br>1b - Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enable. |
| 11-10<br>— | -<br>Reserved |
| 9<br>— | - |
| 8-7<br>— | -<br>Reserved |
| 6<br>— | -<br>Reserved |
| 5-4<br>— | -<br>Reserved |
| 3 | Card Interrupt Detection Test<br>This bit only uses for debugging. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CARD_INT_D3_TEST | 0b - Check the card interrupt only when DATA3 is high.<br>1b - Check the card interrupt by ignoring the status of DATA3. |
| 2-0<br>— | -<br>Reserved |

# Chapter 64
# 2.4 GHz Multi-Protocol Radio

## 64.1 Introduction

This document describes the 3rd generation 2.4 GHz Multi-Protocol Radio capable of supporting the modes described below in the 2.4 GHz ISM band.

- IEEE Std. 802.15.1 v4.2 Bluetooth Low Energy (BLE) single-mode device operation
- IEEE Std. 802.15.4-2006 compliant ZigBee RF4CE/IP
- IEEE Std. 802.15.4j-2012 in Medical Body Area Network (MBAN) frequency bands spanning from 2360 MHz to 2400 MHz
- Generic FSK at 1Mbps, 500Kbps, or 250Kbps
- Concurrent 802.15.4 and BLE operation

The Radio is comprised of a Constant-Envelope Transmitter and a Quadrature Zero-IF Receiver.

The Radio block diagram is provided for illustration of the internal organization of the radio.

2.4GHZ RADIO HIERARCHY DIAGRAM

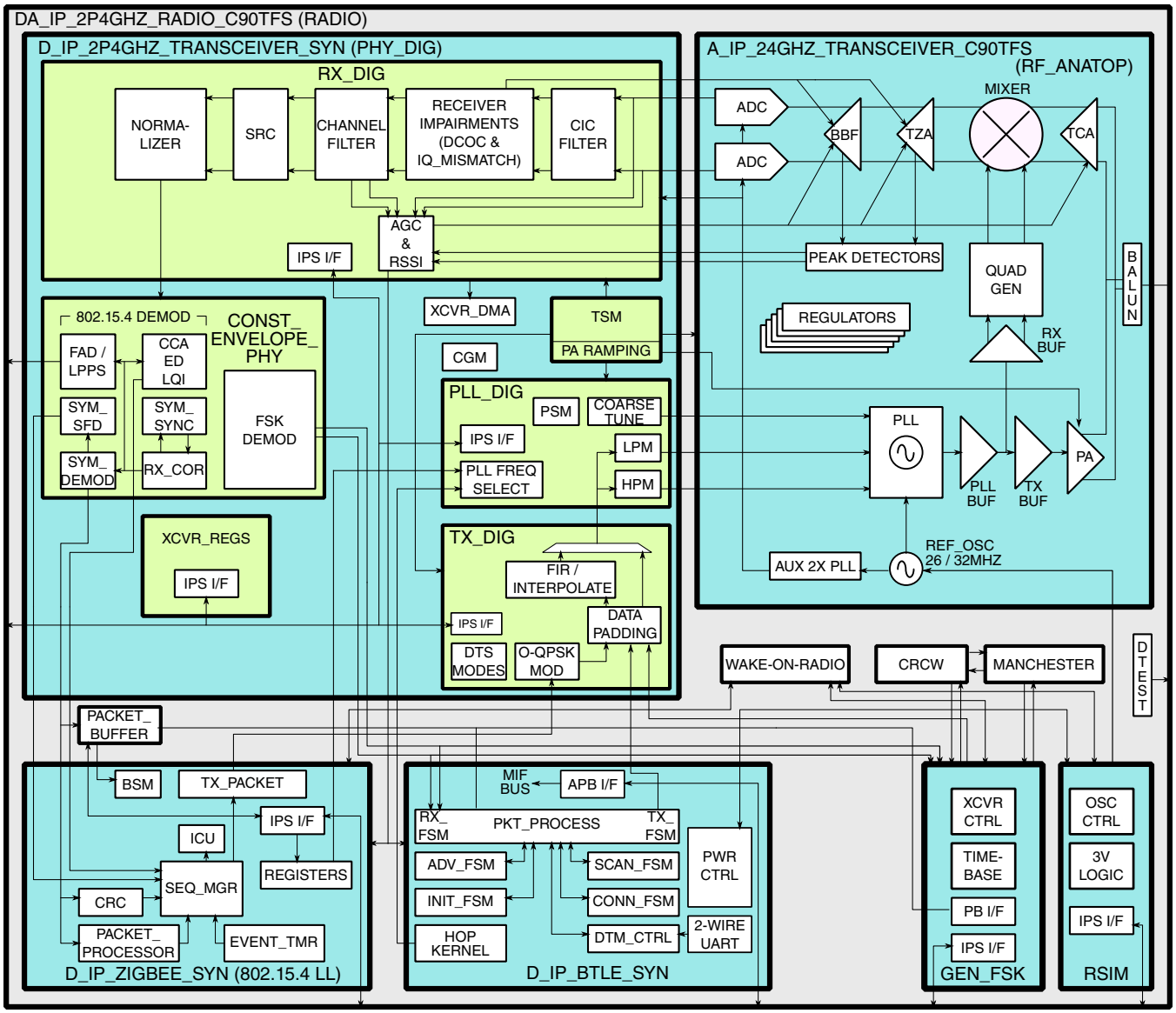



**Figure 64-1. Radio Block diagram**

# 64.2 Radio System Integration Module

## 64.2.1 Introduction

The Radio System Integration Module (RSIM) provides support for the Radio specific System Integration requirements.

#### 64.2.1.1 Features

The RSIM includes the following features:

- Status bit and enable of the BLE Ref Osc (Sysclk) Request
- Interrupt enable and flag for BLE Ref Osc (Sysclk) Request
- Power Mode specific enables for the RF Ref Osc
- Status bit and override control of the RF Ref Osc Ready signal
- Control of the SoC Platform Asynchronous Gasket Bypass
- Override control of the Stop Acknowledge signal from the Radio to the SoC
- Control bit to force BLE to exit deep sleep mode
- Control of the delay time for the BLE Active signal to the Flash
- Output enables for the two pad outputs of the BLE Active signal
- A 40-bit unique MAC address loaded from the SoC Flash IFR at Power On Reset
- SoC padring isolation controls for sensitive RF analog circuits
- Radio Reset blocking and a software Radio Reset control bit to support SoC and Radio concurrent testing
- Control Zigbee/Generic FSK + DSM related.

### 64.2.1.2 Block diagram

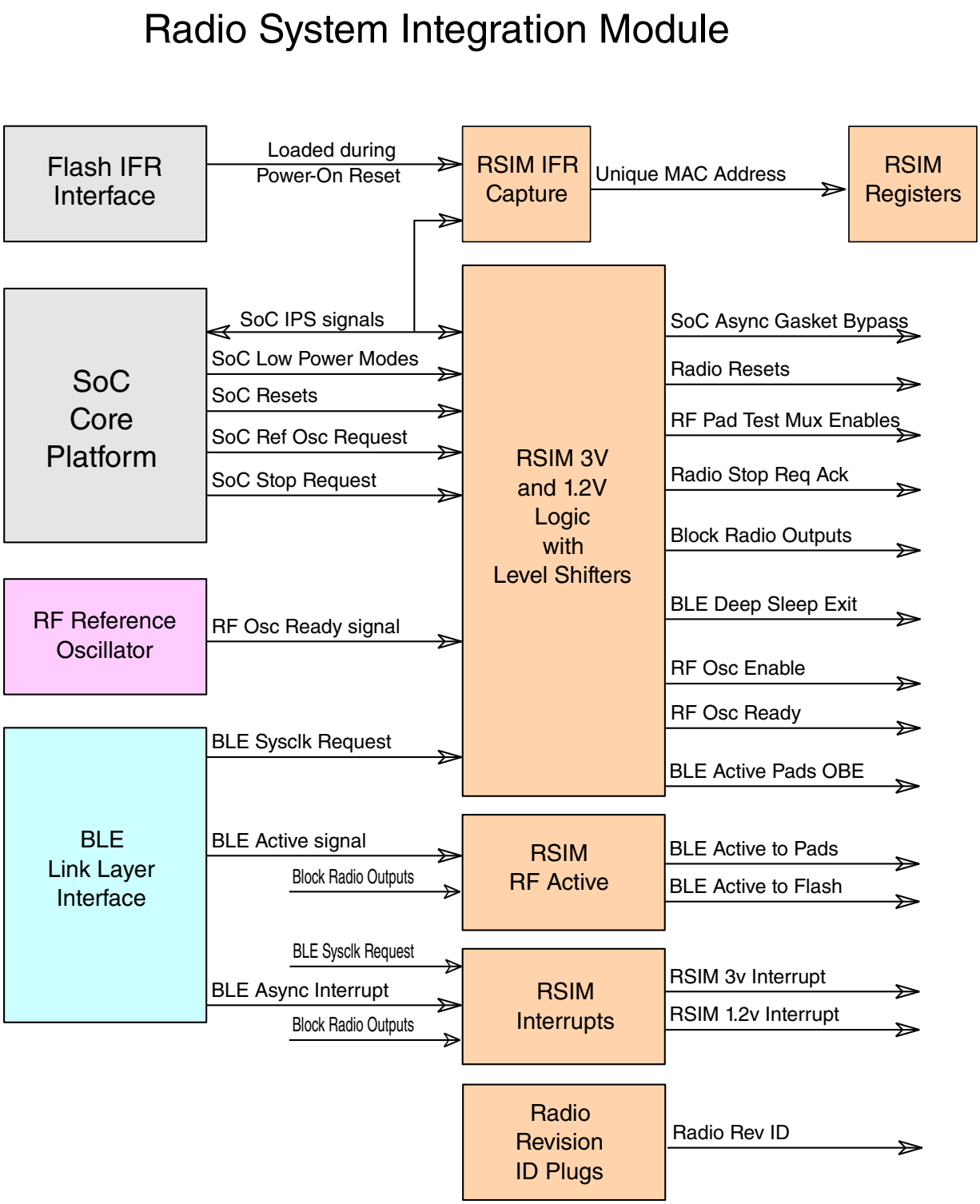



**Figure 64-2. Block diagram**

## 64.2.2 Radio Power

The Radio 1.2v digital operates as an independent power domain from the other SoC 1.2v power domains and this power domain is controlled by hardware in the Radio as described in this section.

### 64.2.2.1 Radio Power Modes

The Radio has four main Power Modes:

1. RVLLS : Radio Off Mode, this is the default mode the Radio enters after any POR, no Radio functionality is available. Whenever RVLLS is entered all of the Radio Link Layer logic will be reset to the POR state.
2. RRun : Radio Run Mode, the Radio enters this mode as described in the Radio Run Request section below.
3. RLLS : Radio State Retention Mode, the Radio digital logic retains it's state in a low power configuration, Deep Sleep Modes can be enabled, Radio Analog is inactive.
4. RLLS-OSC : Radio Osc Ready Mode, the Radio digital logic retains it's state in a low power configuration, Deep Sleep Modes can be enabled, Radio Oscillator is kept active to avoid a warmup delay before going into RRun mode.

Note that when the Radio is in Run Mode the SoC System Power Module (SPM) can be further configured into VLPS mode for some additional power savings.

The Radio Power Mode transitions are illustrated in the diagram below:

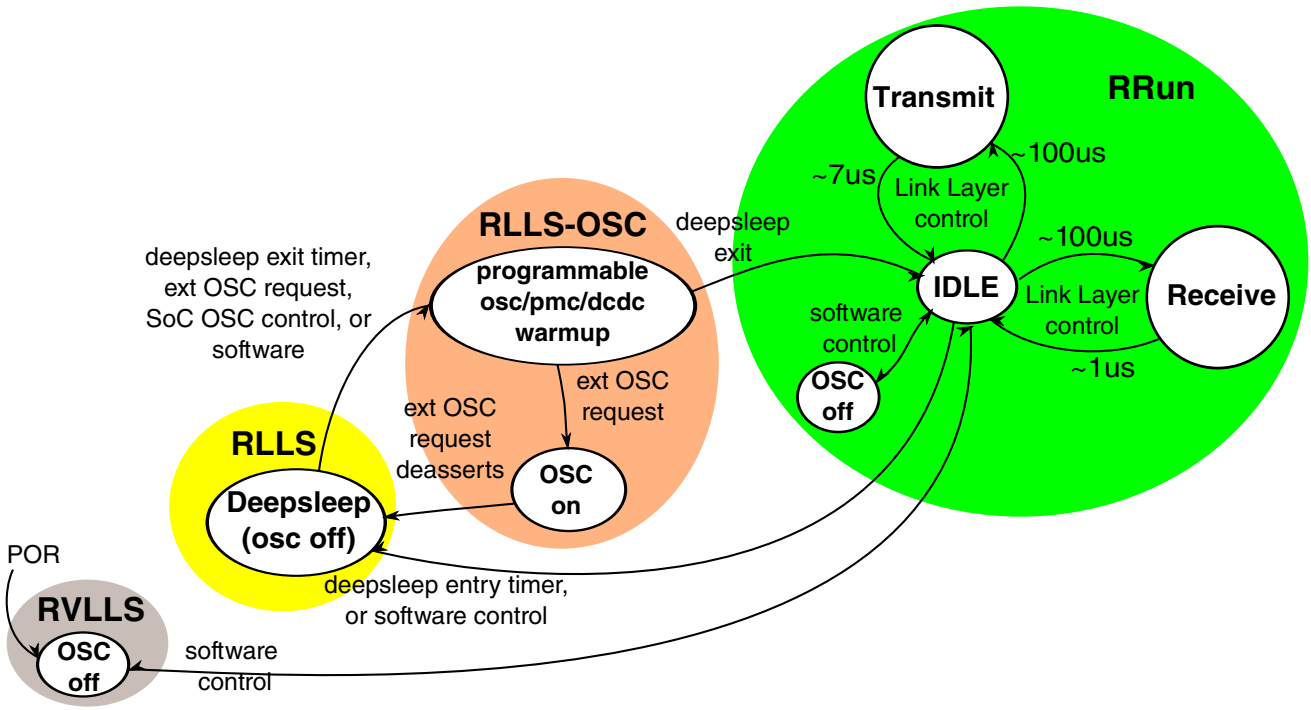


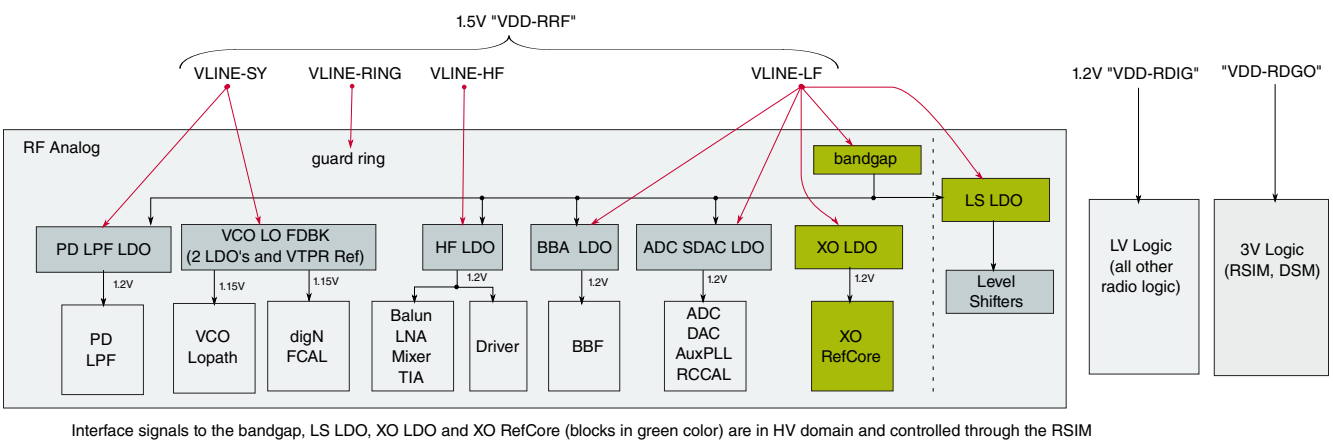

**Figure 64-3. Radio Power Mode Transitions**

## 64.2.2.2 Radio Power Domains

The Radio has three Power Domains:

1. RF Analog Power, VDD-RRF, always powered
2. Radio 1.2v Digital Power, VDD-RDIG, only powered in RRun Mode
3. Radio 1.8v Digital Power, VDD-RDGO, always powered

The Power Domains are connected as shown in the following figure:




Interface signals to the bandgap, LS LDO, XO LDO and XO RefCore (blocks in green color) are in HV domain and controlled through the RSIM

**Figure 64-4. Radio Power Domain**

### 64.2.2.3 RF Analog Bandgap Trim

The RF Analog LDO voltage regulators use the RF Analog Bandgap as their voltage reference, this reference level is trimmed at the factory and the trim values are stored in the Flash IFR.

After any SoC POR event the Radio will load the RF Analog BG_1V_TRIM and BG_IBIAS_5U_TRIM registers with the Flash IFR values; these register values can then be overridden by software as needed.

### 64.2.2.4 Power-On Handshake

The Radio has a Power-On Handshake method with the SPM whereby the Radio asserts or de-assserts a request for the SPM to enter Full Power Run Regulation mode and the SPM acknowledges the request after the transition into or out of Run Regulation is completed.

#### 64.2.2.4.1 Radio Run Request

The Radio can request that the SPM enter Run Regulation Mode in three ways:

1. Software can set the RSIM_RUN_REQUEST register bit, which will force Radio Run mode while it is set.
2. Any of the Radio Deep Sleep State Machines can request that the Radio wakeup from DSM, and the State Machines can then automatically reenter DSM and remove the Run Request without waking up the MCU.
3. An External Osc Request can request the RF Osc which can be configured as a Radio Run Request, and the Run Request will de-assert when the External Osc Request de-asserts.

Each of the Run Request methods acts in an OR fashion so that the Radio stays in Run Mode as long as any one of these is active.

Note that the SPM can be configured in various ways to provide this Run Regulation power, including using an External Power Supply.

The RSIM has several registers that control and observe the Run Request as shown in the following table:

**Table 64-1. RSIM Run Request Bits**

| RSIM Register Field | Function | Description |
|---|---|---|

*Table continues on the next page...*

**Table 64-1. RSIM Run Request Bits (continued)**

| | | |
|---|---|---|
| RSIM_RUN_REQUEST | RSIM Run Regulator Request | When this bit is set, the RSIM will request the SPM Run Regulator to be turned on and the RSIM will then stop requesting Stop after the Run Regulator Acknowledge signal is received from the SPM module. |
| RADIO_RUN_REQ_STAT | Radio Run Request Status | This is the value of the Radio Run Request signal being sent to the SPM module, including any overrides. |
| RSIM_RUN_REQ_OVRD | RSIM Run Request Override | This bit directly controls the RSIM Run Request signal to the SPM when the RSIM Run Request Override is enabled. |
| RSIM_RUN_REQ_OVRD_EN | RSIM Run Request Override Enable | This bit enables the RSIM Run Request Override bit. |

#### 64.2.2.4.2 SPM Observe and Overrides

The RSIM has several registers that observe and override the SPM signals as shown in the following table:

**Table 64-2. RSIM SPM Run Acknowledge Bits**

| RSIM Register Field | Function | Description |
|---|---|---|
| SPM_RUN_ACK_STAT | SPM Run Request Acknowledge Status | This is the value of the SPM Run request Acknowledge signal being sent to the Radio. |
| SPM_RUN_REQ_ACK_OVRD | SPM Run Request Acknowledge Override | This bit directly controls the SPM Run Request Acknowledge signal when the SPM Run Request Acknowledge Override is enabled. |
| SPM_RUN_REQ_ACK_OVRD_EN | SPM Run Request Acknowledge Override Enable | This bit enables the SPM Run Request Acknowledge Override bit. |

#### 64.2.2.4.3 Radio Run Acknowledged

After the Radio has requested that the SPM enter Run Regulation Mode, software can poll the SPM_RUN_ACK_STAT bit to confirm that the SPM has completed the Radio request.

### 64.2.2.5 Power-On Sequencer

The RSIM has a Power-On Sequencer that requests Radio Run Mode, enables the RF Osc, and signals RF_Active. The Power-On Sequencer is programmable and has hardware connections to all of the Radio's Deep Sleep State Machines.

When enabled, the Radio will use the same Radio Power-On Sequence, regardless of the Radio Protocol. The timing of the sequence is controlled by the RSIM DSM_PWR_OFFSET register where the POWER_OFFSET, COARSE_DELAY, FINE_DELAY, and ACTIVE_WARNING values are programmed, and by the BLE WAKEUP_CONFIG and WAKEUP_CONFIG2 registers.

The Power-On timing will always be the same prior to the exit from Deep Sleep Mode, even if the Radio Power is already on. This is needed to support all the various Deep Sleep Modes that each requires precise On-Air timing and co-existence with each other, as well as the case where software may be turning the Radio Power on and off.

Software can always directly turn on the Radio Power by setting the RSIM_POWER register bit RSIM_RUN_REQUEST, and can then directly turn on the RF OSC by setting the RSIM_CTRL bit RF_OSC_EN

The Radio Power-On Sequence can also be used when the SoC External Osc Request pin is used to request the RF OSC by setting the RSIM OSC_CONTROL bit EXT_ACTIVE_EN.

The Radio Power-On Sequence is illustrated in the following diagram.

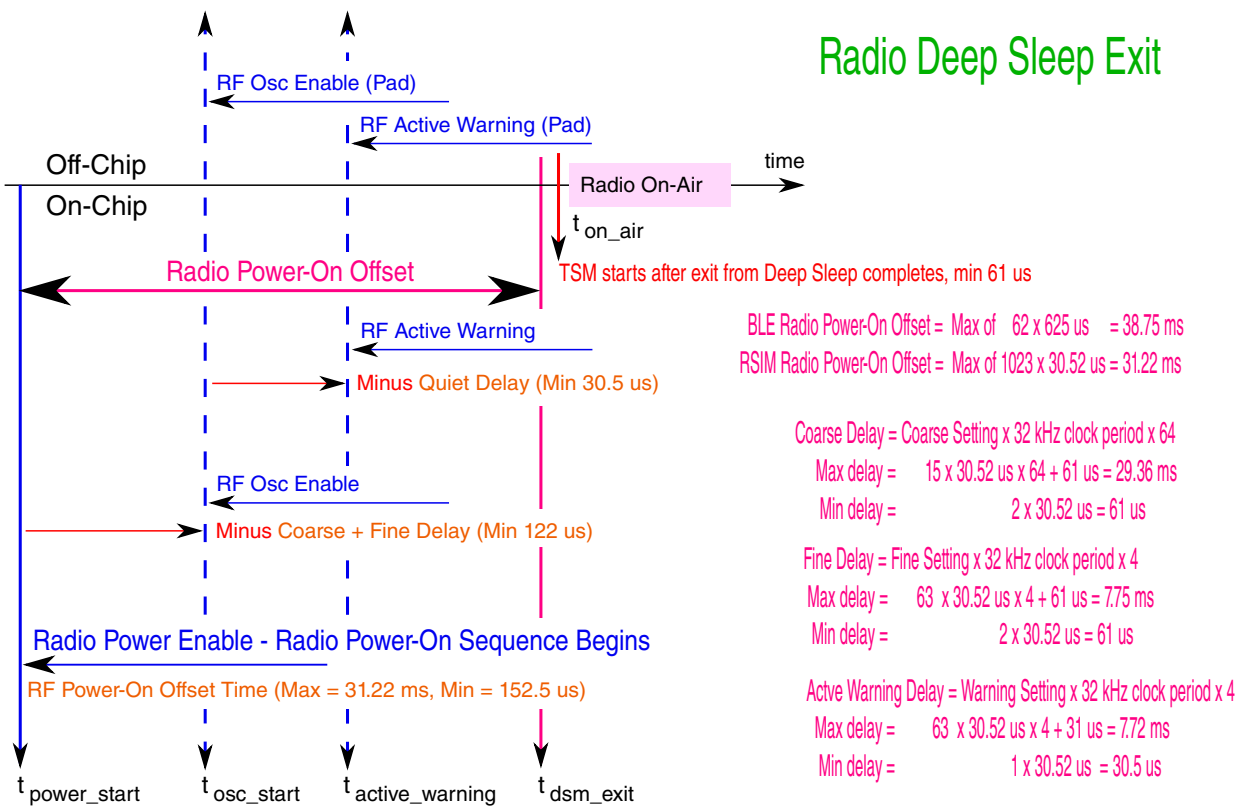



**Figure 64-5. Radio Power-On Sequence Diagram**

#### 64.2.2.5.1 Milestones

The Radio Power-On Sequence has four timing milestones:

1. $\mathbf{T_{power_start}}$ : A Run Request is sent to request the SPM to enter full power Run Mode
2. $\mathbf{T_{osc_start}}$ : The RF Analog controls are set to warm-up the RF OSC and a signal is presented on the RF_OSC_EN SoC pin
3. $\mathbf{T_{active_warning}}$ : A signal is presented on the RF_ACTIVE SoC pin to indicate the upcoming Radio Operation
4. $\mathbf{T_{dsm_exit}}$ : Link Layer exits Deep Sleep mode
5. $\mathbf{T_{on_air}}$ : TSM sequence starts and Radio goes On-Air

The Link Layers all have methods of computing their DSM Exit times which then determine when the Radio goes On-Air ($\mathbf{T_{on_air}}$).

The RSIM and BLE Deep Sleep State Machines have a hardware handshake system with the Radio Power-On Sequencer, and there are separate Start Time offsets to achieve the RSIM and BLE DSM Exit times.

##### 64.2.2.5.1.1 RSIM Offset

For the RSIM Deep Sleep State Machines the POWER_OFFSET value is programmed into the RSIM DSM_PWR_OFFSET register. The amount of the Start Time offset can be calculated as:

- **POWER_OFFSET[9:0]** : This is the number of 32.768 kHz clocks (30.518 us) to use as a Start Time offset prior to the RSIM Deep Sleep Machines entering the Active State.

##### 64.2.2.5.1.2 BLE Offset

Please refer to the BLE Link Layer Controller Programming section for details on the BLE Offset.

#### 64.2.2.5.2 Settings and Calculations

The Radio Power-On Sequence Time is programmed using the RSIM DSM_PWR_OFFSET register which contains three Radio Power-On Sequence settings, please refer to the Radio Deep Sleep section for more details.

##### 64.2.2.5.2.1 Calculating the Power-On Milestones

The Radio Power-On Sequence Milestones can be calculated as follows:

1. $\mathbf{T_{power_start}}$ : Power-On Offset time = Deep Sleep Machine enters state WAIT_OSC_STABLE.

2. $\mathbf{T_{osc_start}}$ : RF_OSC_EN pin time = $\mathbf{T_{power_start}}$ + COARSE_DELAY + FINE_DELAY.
3. $\mathbf{T_{active_warning}}$ : RF_ACTIVE pin time = $\mathbf{T_{osc_start}}$ + ACTIVE_WARNING.
4. $\mathbf{T_{dsm_exit}}$ : The Link Layer exit from Deep Sleep takes a minimum of 61 us to complete.
5. $\mathbf{T_{on_air}}$ : The time it takes to go On-Air after exiting Deep Sleep depends on the Protocol and TSM timings.

#### 64.2.2.5.2.2 Minimum Power-On Time

The minimum Radio Power-On Sequence time can be calculated as follows:

If all of the DSM_PWR_OFFSET registers are set to zero:

- **COARSE_DELAY** = 0
- **FINE_DELAY** = 0
- **ACTIVE_WARNING** = 0

Then the Minimum Radio Power-On Sequence time =

- Minimum Coarse delay of 30.52 us +
- Minimum Fine delay of 30.52 us +
- Minimum Active delay of 30.52 us +
- Minimum Additional Hardware delay to complete the exit from Deep Sleep mode of 122.08 us

Which gives a minimum Power-On Sequence Time of 213.64 us.

And the Minimum Radio Power-On Offset time =

- Minimum Additional Hardware delay to start the Power-On Sequence of 61.04 us +
- Minimum Coarse delay of 30.52 us +
- Minimum Fine delay of 30.52 us +
- Minimum Active delay of 30.52 us

Which gives a minimum Power-On Offset Time of 152.6 us.

If the Minimum Power-On Offset Time is programmed, then the RF_Active pin will be asserted 61.04 us before the Deep Sleep mode exit.

If the POWER_OFFSET value is increased beyond the minimum, or other calculated value, the effect will be to increase the time between the RF_ACTIVE pin signal asserting and the Deep Sleep mode exit.

#### 64.2.2.5.3 Software Control of RF Active Warning

Software can manually control the RF_ACTIVE pin using these three methods:

1. Software RF Active bits in the RSIM : The SW_RF_ACTIVE_EN and SW_RF_ACTIVE_BIT can be used to assert a Software RF Active Warning, which will be OR'd with the three Deep Sleep Machine RF_ACTIVE warning signals and presented on the RF_ACTIVE pin.
2. RF Active Override bits in the RSIM : These can be used to either block all of the Deep Sleep Machine's from asserting the RF_ACTIVE pin, or they can be used to always assert the RF_ACTIVE pin. If the DSM RF_ACTICE signals are forced low then software has complete control of RF_ACTIVE.
3. Software can also use the SW_RF_ACTIVE_ENDS_WITH_TSM_EN bit to auto clear the RF_ACTIVE pin after the next TSM operation.

Please refer to the Wi-Fi Coexistence section for more details.

#### 64.2.2.5.4 Clearing of RF Active Warning

The default behavior is for the three Deep Sleep Machine's RF_ACTIVE warning signals to clear after the Radio Operations complete and the Link Layers go back into Deep Sleep mode.

There is an RSIM bit, RF_ACTIVE_ENDS_WITH_TSM, which instead will clear the three Deep Sleep Machines RF_ACTIVE warning signals at the end of the next TSM operation.

### 64.2.3 Radio Clocks

The Radio uses the RF Analog Reference Oscillator or an equivalently precise external clock for all Radio Operations. This RF Osc is also available for the SoC to use as it's clock source. There is also a method that allows the SoC to supply a less precise clock to the Radio that can be used to configure the Radio registers before the Radio is enabled.

#### 64.2.3.1 Clock Diagram

The Radio Clock Tree is illustrated in the following diagram.

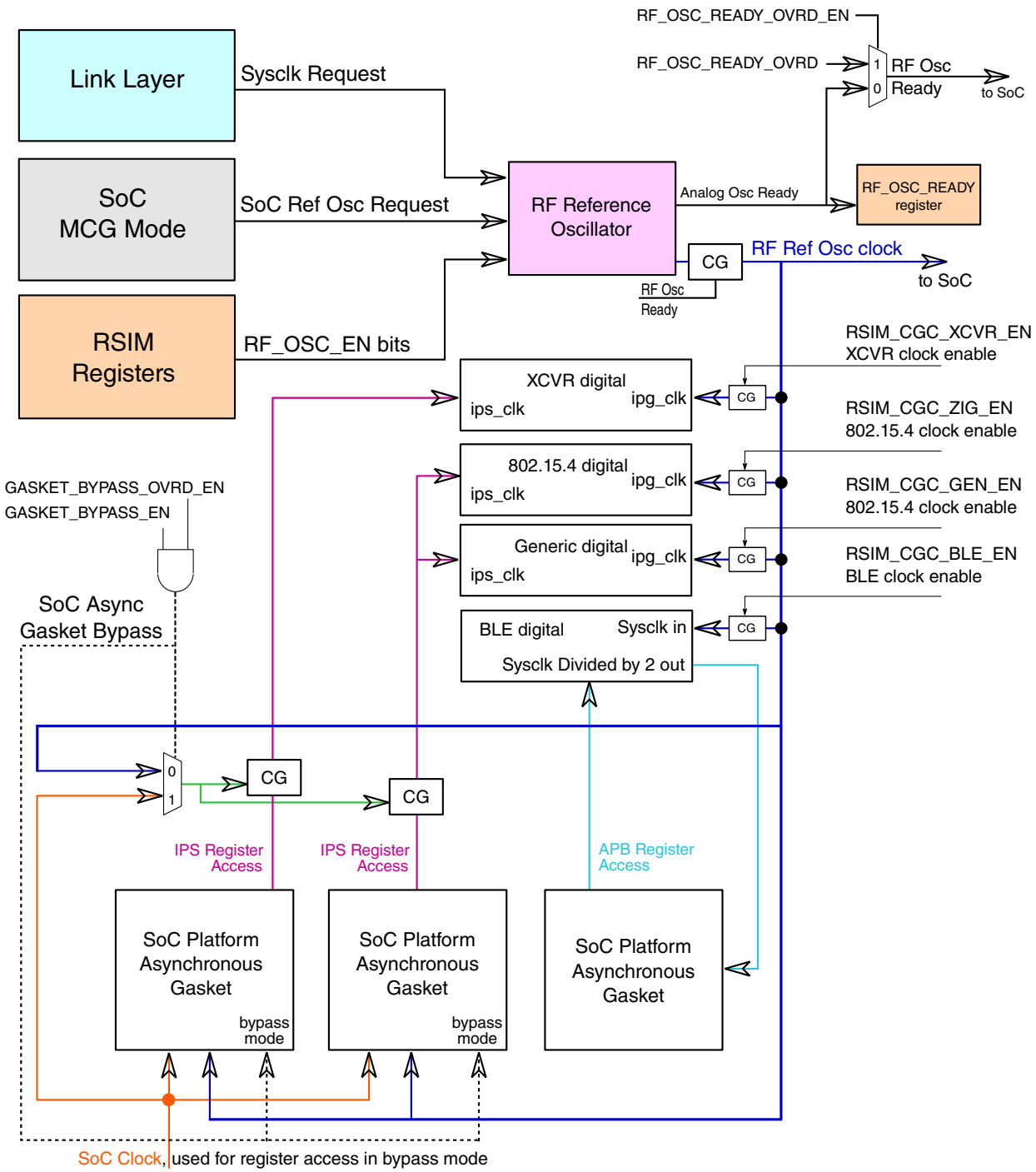



**Figure 64-6. Radio Clocking Configuration**

## 64.2.3.2 RF Osc Enable

The RF Analog Reference Oscillator must be enabled and ready before it can be used as a clock source. There are three methods of enabling the RF Osc as listed here:

1. The SoC can request the RF Osc.

2. One of the Radio Link Layers can request the RF Osc, either in RRun mode or in RLLS mode using a Deep Sleep State Machine.
3. The RF_OSC_EN bit can be set, more details in the table below.

| RF_OSC_EN | Radio Run Mode Functionality | Radio Stop Modes Functionality |
|---|---|---|
| 0 | RF Ref Osc will be Off unless it is turned On by the SoC, by a Radio link layer using RRun Mode, or by an External Pin request. | Warning: Radio will not go into Stop mode while the RF Ref Osc is being used by the SoC, or by a Radio link layer unless the Link Layer is using a Deep Sleep State Machine.<br>RF Ref Osc will be Off unless there is an external pin request. |
| 1 | The RF Ref Osc will be On.<br>Warning: The RF Ref Osc can not be enabled until the Radio has entered RRun Mode. | Radio will not go into Stop mode while RF_OSC_EN is set |

### 64.2.3.3 RF Osc Ready

The RF Analog Reference Oscillator has a programmable count time that elapses before the RF_OSC_READY signal is asserted. This count time is set using the BB_XTAL_READY_COUNT_SEL register as shown in the table below:

| BB_XTAL_READY_COUNT_SEL | Count Value | Count Time, us (32MHz Ref Osc) |
|---|---|---|
| 0 | 1024 | 32 |
| 1 | 2048 | 64 |
| 2 | 4096 | 128 |
| 3 | 8192 | 256 |

The current status of RF_OSC_READY signal from the RF Analog can be read using the RF_OSC_READY bit.

The RF_OSC_READY signal from the RF Analog can be overridden using the RF_OSC_READY_OVRD_EN and RF_OSC_READY_OVRD bits. The status bit will continue to show the RF Analog status but everywhere else will use the overridden value.

#### 64.2.3.4 RF Osc Bypass

The RF_OSC_BYPASS_EN bit engages the RF Ref Osc analog bypass circuit if the RF Ref Osc is enabled. When the RF Ref Osc is in bypass mode it passes the RF EXTAL clock as the RF Ref Osc clock. Note that the RF Ref Osc Ready signal functions normally in RF OSC Bypass mode, unless overridden with the RF_OSC_READY_OVRD_EN bit.

#### 64.2.3.5 Clock Gating Bits

To conserve power, the clocks to most Radio modules can be turned off using the RSIM_CGC_EN bits in the RSIM registers. These bits are cleared after any reset, which disables the clock to the corresponding module. Prior to initializing a module, set the corresponding RSIM_CGC_EN bit to enable the clock. Before turning off the clock, make sure to disable the module.

#### 64.2.3.6 Radio Registers Access (CAUTION)

WARNING: If the Radio Register Clocks are not configured correctly before a Memory Access the SoC Core will halt. The descriptions below and the RSIM Register descriptions are intended to help avoid such a result.

##### 64.2.3.6.1 BLE Link Layer Registers Access

The Radio Registers for the BLE Link Layer are accessed through an SoC Asynchronous Gasket in the Core Platform, and can only be accessed when the RF Ref Osc is Enabled and Ready, this is because the BLE Link Layer requires an accurate time-base clock for its time-keeping functions.

Software must use the RSIM_RUN_REQUEST bit to put the Radio in RRun Mode and then enable the RF_OSC_EN bit to enable the RF Ref Osc. After the RF Ref Osc is enabled, software can monitor the RF_OSC_READY bit to determine when the RF Ref Osc is Enabled and Ready.

WARNING: If the RF Ref Osc is not Enabled and Ready, and a BLE Link Layer Memory Access is attempted, then the SoC Asynchronous Gasket will block the SoC Core from proceeding as it waits for the Radio to send it the RF Ref Osc.

#### 64.2.3.6.2 XCVR and Link Layer Registers Access

The Radio Registers for the XCVR and the Link Layers are accessed through an SoC Asynchronous Gasket in the Core Platform, and they can be accessed with either the RF Ref Osc (Not Bypassed) or the SoC IPG clock (Bypassed).

The default behavior of this Gasket is Not Bypassed, which means the Radio registers are accessed using the RF Ref Osc clock.

When the Gasket is Not Bypassed, the Radio sends the RF Ref Osc to the asynchronous gasket which then uses that clock for SoC register accesses of the Radio registers. This requires the RF Ref Osc to be Enabled and Ready. Software must use the RSIM_RUN_REQUEST bit to put the Radio in RRun Mode and then enable the RF_OSC_EN bit to enable the RF Ref Osc. After the RF Ref Osc is enabled, software can monitor the RF_OSC_READY bit to determine when the RF Ref Osc is Enabled and Ready.

If the RF Ref Osc is not Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, which forces the use of the SoC IPG clock as the register access clock.

If the RF Ref Osc IS Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, but this is an ILLEGAL ACCESS if the Radio is operational, DO NOT write Radio registers with the Asynchronous Gasket Bypassed when the Radio is Operating. Test mode access is allowed but glitches may occur.

WARNING: If the RF Ref Osc is not Enabled and Ready, and a XCVR or Link Layer Memory Access is attempted, then the SoC Asynchronous Gasket will block the SoC Core from proceeding as it waits for the Radio to send it the RF Ref Osc.

#### 64.2.3.6.3 Asynchronous Gasket Bypass

The intent of Bypass Mode is to allow software to configure the Radio XCVR and Link Layer registers before the RF Ref Osc is Enabled and Ready.

The following table shows which clock is being used to access the Radio XCVR and Link Layer registers.

| Gasket Bypass Override Enable | Gasket Bypass Override | XCVR and Link Layer Register Clock |
|---|---|---|
| 1 | 0 | RF Ref Osc Clock |
| 1 | 1 | SoC IPG Clock |
| 0 | x | RF Ref Osc Clock |

More details are available in the Radio Clocks diagram, the Radio Gasket Bypass section, and in the RSIM Gasket Bypass Registers descriptions.

### 64.2.3.7 External Osc Request

The RF Analog Reference Oscillator has an External Osc mode in which the RF Osc Analog circuits can be enabled and the RF Osc can be ready and either presented externally on an SoC pin, or not.

#### 64.2.3.7.1 SoC XTAL_OUT Pin Enable

The RF Analog Driver for the SoC XTAL_OUT pin has to be enabled using the XTAL_OUT_BUF_EN bit in the RSIM ANA_TEST register.

Warning: If this bit is not set then no RF Osc signal will be presented on the SoC Pin.

#### 64.2.3.7.2 External Osc Request Register Bits

The RSIM has several bits that are used to select the External Osc mode configuration as shown in the table below:

**Table 64-3. External Osc Mode Configuration Bits**

| RSIM Register Field | Function | Setting | Description |
|---|---|---|---|
| BLOCK_EXT_OSC_PWR_REQ | Block External Requests for RF OSC from starting a Radio Power Wakeup Sequence | 0 | Configuring the RF Osc in External Osc mode causes the Radio to request Run Regulation mode. |
| | | 1 | Radio will not request Run Regulation mode when the RF Osc is configured in External Osc mode and the Radio is otherwise in a Stop mode. |
| RADIO_EXT_OSC_RF_EN_SEL | Radio External Request for RF OSC Select | 0 | SoC pin connected to EXT_OSC_RF_EN_0 is used to configure the RF Osc in External Osc mode. |
| | | 1 | SoC pin connected to EXT_OSC_RF_EN_1 is used to configure the RF Osc in External Osc mode. |
| RADIO_EXT_OSC_OVRD | Radio External Request for RF OSC Override | 0 | If the RADIO_EXT_OSC_OVRD_EN bit is set, then the RF Osc will not be configured in External Osc mode. |

*Table continues on the next page...*

**Table 64-3. External Osc Mode Configuration Bits (continued)**

| | | | |
|---|---|---|---|
| | | 1 | If the RADIO_EXT_OSC_OVRD_EN bit is set, then the RF Osc will be configured in External Osc mode. |
| RADIO_EXT_OSC_OVRD_EN | Radio External Request for RF OSC Override Enable | 0 | SoC external pin requests will control the RF Osc External Osc mode configuration. |
| | | 1 | The RADIO_EXT_OSC_OVRD bit will control the RF Osc External Osc mode configuration and any external pin request will be ignored. |

### 64.2.3.8 SoC RF Osc Pins

The SoC Pins related to the RF Osc are shown in the table below.

| Radio Pin Name | Pin Direction | SoC Pin Name | Pin Description |
|---|---|---|---|
| RF_INT_OSC_EN | I | RF_RFOSC_EN | External request to turn on the Radio's internal RF Osc. The Osc will be present on the XTAL_OUT pin if the RF Osc XTAL_OUT buffer is enabled by writing the Radio XTAL_OUT_BUF_EN bit in the RSIM ANA_TEST register. |
| RF_EXT_OSC_EN | O | RF_EXT_OSC_EN | Internal request to turn on an External Osc for use by the internal Radio, this can also be a request from the SoC if it is using the RF Osc as it's clock. |

## 64.2.4 Radio Stop Modes

The Radio hardware automatically exits and reenters Stop Mode as part of its interaction with the Deep Sleep State Machines. Software can also control the entry and exit using the register bits described in this section.

### 64.2.4.1 Radio Stop Signals

The Radio Stop mode transitions are handled by a hardware handshake with the SPM as shown in the diagram below:

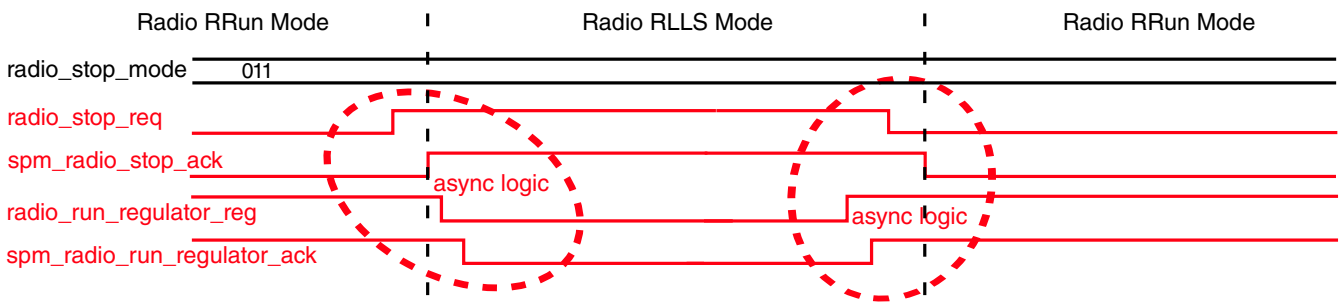



**Figure 64-7. Radio Stop Signals Diagram**

### 64.2.4.2 Radio Stop Request

The RSIM controls the Stop Request signal based upon Link Layer and software requests. Normally the Stop Request signal only changes while the Radio is powered, but it is a legal SPM mode to change the Stop Request while the Radio is powered off.

This signal can be observed and overridden as shown in the table below:

| RSIM Register Field | Function | Description |
|---|---|---|
| RADIO_STOP_REQ_STAT | Radio Stop Request Status | This is the value of the Radio Stop Request signal being sent to the SPM module, including any overrides. |
| RSIM_STOP_REQ_OVRD | Radio Stop Request Override | This bit directly controls the Radio Stop Request signal when the Radio Stop Request Override is enabled |
| RSIM_STOP_REQ_OVRD_EN | Radio Stop Request Override Enable | This bit enables the Radio Stop Request Override bit |

### 64.2.4.3 SPM Stop Acknowledge

The SPM acknowledges the Radio Stop Request after the SPM has changed the state of all the Radio Stop and Power Gating logic. This signal can be observed and overridden as shown in the table below:

| RSIM Register Field | Function | Description |
|---|---|---|

*Table continues on the next page...*

| | | |
|---|---|---|
| SPM_STOP_ACK_STAT | SPM Stop Acknowledge Status | This is the status of the Stop request Acknowledge signal received from the SPM module |
| SPM_STOP_REQ_ACK_OVRD | SPM Stop Request Acknowledge Override | This bit directly controls the SPM Stop Request Acknowledge signal when the SPM Stop Request Acknowledge Override is enabled |
| SPM_STOP_REQ_ACK_OVRD_EN | SPM Stop Request Acknowledge Override Enable | This bit enables the SPM Stop Request Acknowledge Override bit |
| RADIO_STOP_ACK_STAT | Radio Stop Acknowledge Status | This is the status of the Stop request Acknowledge signal being used by the Radio, including any override of the SPM module signal. |

### 64.2.4.4 Radio Stop Mode

The Radio software cannot change the Stop Mode value after a Stop Request has been made, typically the Stop Mode value is set to RLLS as part of the Radio configuration process.

The mode can be set, observed and overridden as shown in the table below:

| RSIM Register Field | Function | Description |
|---|---|---|
| RSIM_STOP_MODE | RSIM lowest allowed Stop Mode | This is the Stop Mode that will be requested when the RSIM asserts a Stop request to the SPM module.<br><br>000 - RSTOP (SPM supports, no planned use case)<br><br>001 - RVLPS (SPM supports, no planned use case)<br><br>011 - RLLS mode (Radio State Retention mode)<br><br>111 - RVLLS mode (This is the POR setting) |
| RADIO_STOP_MODE_OVRD | Radio Stop Mode Override | This bit directly controls the Radio Stop Mode signal when the Radio Stop Mode Override is enabled |
| RADIO_STOP_MODE_OVRD_EN | Radio Stop Mode Override Enable | This bit enables the Radio Stop Mode Override bit |
| RADIO_STOP_MODE_STAT | Radio Stop Mode Status | This is the value of the Radio Stop Mode bus that is sent to the SPM module |

### 64.2.4.5 SPM Isolation Signal

The SPM provides the Stop Mode isolation signal which is used as part of the RF Analog isolation logic. This signal can be observed and overridden as shown in the table below:

| RSIM Register Field | Function | Description |
|---|---|---|
| SPM_ISO_STAT | SPM ISO Status | This is the value of the SPM ISO signal being sent to the Radio. |
| RSIM_ISO_OVRD_EN | RSIM ISO Override Enable | This bit enables the RSIM ISO Override bit |
| RSIM_ISO_OVRD | RSIM ISO Override | This bit directly controls the Radio ISO signal when the RSIM ISO Override is enabled |
| RSIM_RADIO_ISO_POR_OVRD | RSIM ISO_POR Override | This bit directly controls the Radio ISO_POR signal when the RSIM_ISO_OVRD_EN bit is set |
| RADIO_ISO_STAT | Radio Isolation Status | This is the value of the RSIM ISO signal being sent to the Radio Isolation logic, including any overrides. |

### 64.2.4.6 RF Analog Isolation

The RSIM controls the isolation state of the RF Analog in Stop Modes, the detailed signaling description is shown in the following diagram:

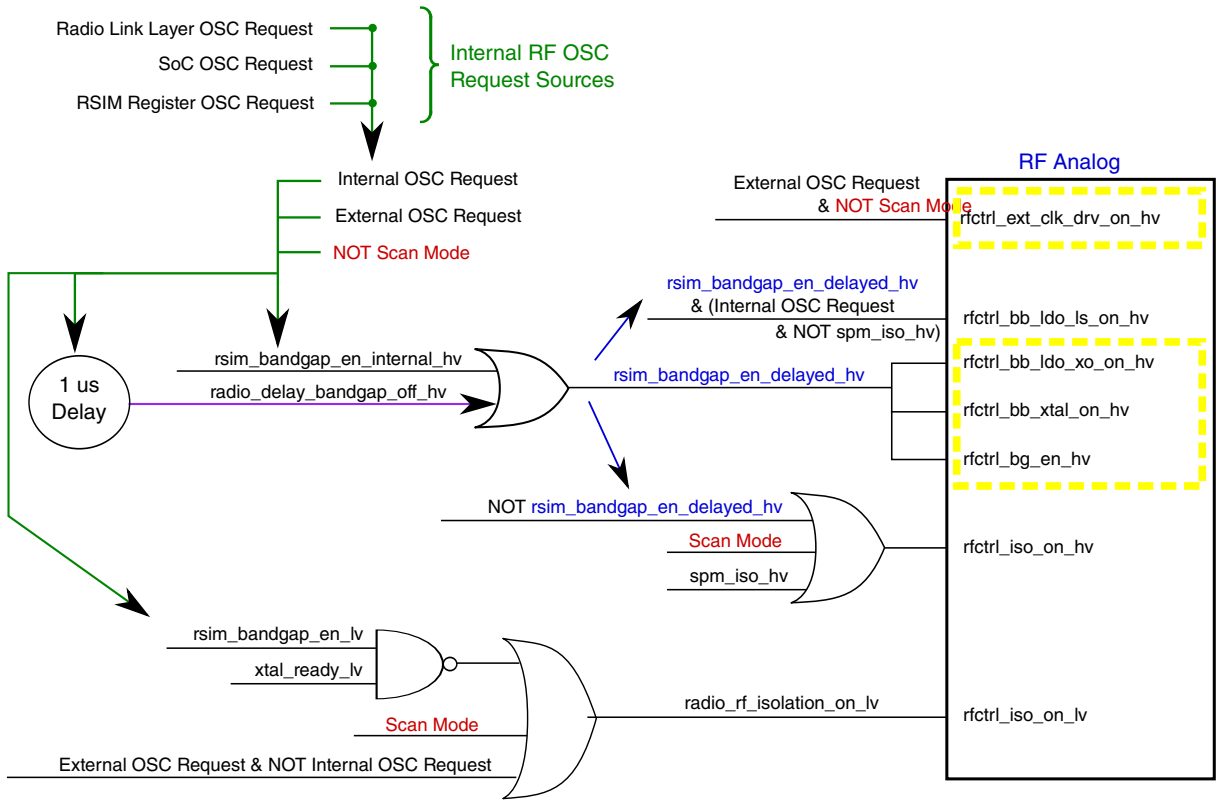



**Figure 64-8. RF Analog Isolation Diagram**

## 64.2.4.7 RAM Retention

The Radio has two internal RAM blocks that are allowed to go into a low power state when they are not being used by a link layer. If the RADIO_RAM_ACCESS_OVRD_EN and RADIO_RAM_ACCESS_OVRD bits are set, then the RAMs are kept in an active power state to allow software to access them.

WARNING: Software should not cause the RAMs to lose power if the Radio Link Layers are using the Deep Sleep State Machines and are configured to use the RAM to store data.

## 64.2.4.8 Radio Idle Mode

When the Radio is in a low power Stop Mode, such as RLLS with the SPM Run Regulator in low power mode, there is a software method that allows the RF Analog Reference Oscillator to be kept enabled in a low power state and thereby warmed up and ready for a faster transition from Stop to Run mode.

In order to configure for this Idle Mode software should write these bits before going into Stop:

- RSIM_ANA_TEST_XTAL_OUT_BUF_EN = 0, Turn off the External Pin Drive Buffer so that no Osc Clock will be present.
- RSIM_SW_CONFIG_BLOCK_EXT_OSC_PWR_REQ = 1, Radio will not request full Run Regulation mode when EXT_OSC is requested.
- RSIM_RF_OSC_CTRL_RADIO_EXT_OSC_OVRD_EN = 1, Enable the override bit.
- RSIM_RF_OSC_CTRL_RADIO_EXT_OSC_OVRD = 1, This is the software request to enable the Osc in Idle Mode.

## 64.2.5 Radio Deep Sleep

The RSIM has two Deep Sleep State Machines to support Wake on Radio and Manual Deep Sleep configurations.

Note that the BLE Link Layer has it's own Deep Sleep FSM, please refer to the BLE Link Layer chapters for additional information.

This section covers the portions of the Deep Sleep Hardware contained in the RSIM.

### 64.2.5.1 Deep Sleep Entry

The RSIM Deep Sleep State Machines will enter sleep automatically based on hardware requests from the link layer or the time as counted in the DSM_TIMER. Please refer to the Wake-On-Radio and the Manual DSM sections for more details.

### 64.2.5.2 Deep Sleep Wakeup

The RSIM Deep Sleep State Machines will automatically wakeup based upon the time as counted in the DSM_TIMER.

The DSM_TIMER is enabled using the DSM_TIMER_EN bit, and the timer can be cleared using the DSM_TIMER_CLR bit.

#### 64.2.5.2.1 Wakeup Sequence Settings

Software must configure the Deep Sleep Wakeup Sequence using these RSIM register bits. The Wakeup Sequence will be the same for all of the Radio Deep Sleep Wakeups.

The required settings and the method of calculating them are shown in the following table.

**Table 64-4. Wakeup Sequence Settings**

| RSIM Register Field | Function | Description |
|---|---|---|
| DSM_POWER_OFFSET_TIME | Deep Sleep Wakeup Power Offset Time | This register is the calculated Radio Power-On Sequence Time. |
| COARSE_DELAY | Deep Sleep Wakeup Coarse Delay Time | This is the first delay after the beginning of the Power-On sequence, the range is 0-15 and the resulting delay is 1.953 ms x COARSE_DELAY + 30.52 us, with a minimum delay of 30.52 us. |
| FINE_DELAY | Deep Sleep Wakeup Fine Delay Time | This is the second delay after the beginning of the Power-On sequence, the range is 0-63 and the resulting delay is 122.07 us x FINE_DELAY + 30.52 us, with a minimum delay of 30.52 us. This is when the RF OSC is enabled and so the Radio Analog Power must be stable at this milestone. |
| ACTIVE_WARNING | Deep Sleep Wakeup RF Active Warning Time | This is the delay after RF OSC has been enabled, the range is 0-63 and the resulting delay is 122.07 us x ACTIVE_WARNING + 30.52 us, with a minimum delay of 30.52 us. This is typically when the RF Osc should be stable. |

#### 64.2.5.2.2 BLE Link Layer Controller Programming

For BLE, the Link Layer Controller needs to be programmed to account for the RSIM DSM total wakeup time, including the Coarse, Fine, and Active Warning delay times. The BLE Link Layer Controller initiates the wakeup process by asserting **orf_sysclk_req_3v** to RSIM. At the same instant, the BLE DSM FSM transitions from DEEP_SLEEP to WAIT_OSC_STABLE state, as shown in the following diagram:

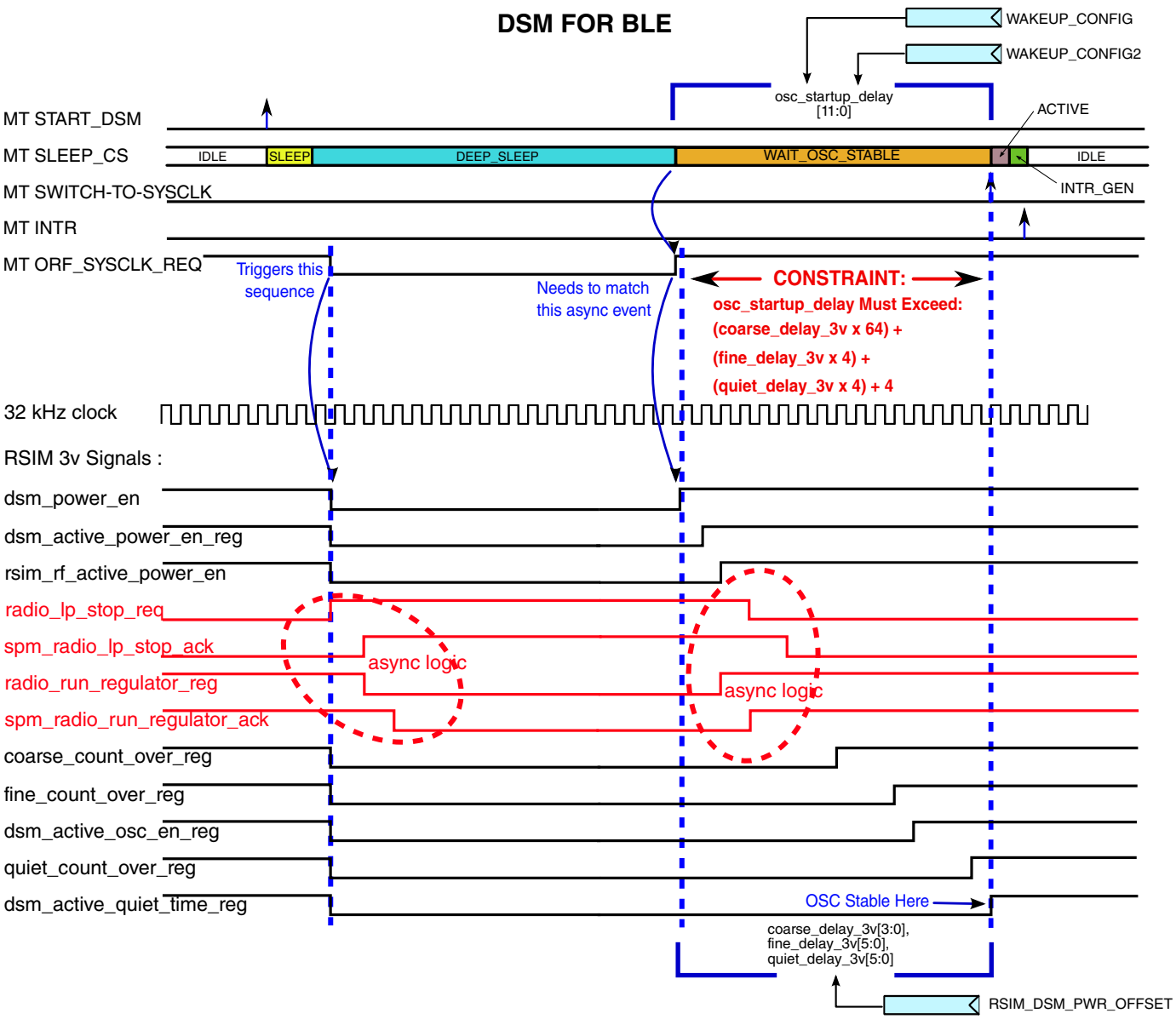



**Figure 64-9. Deep Sleep Mode For BLE**

The BLE FSM holds in WAIT_OSC_STABLE until the radio is fully powered and the RF Oscillator has stabilized. This requirement is met when the following equation is satisfied:

RSIM TOTAL WAKEUP = (COARSE * 64) + 1 + (FINE * 4) + 1 + (ACTIVE_WARNING * 4) + 1 + 2 , in units of 32.768KHz clock periods

The duration of the WAIT_OSC_STABLE state must exceed RSIM TOTAL WARMUP time, to ensure the BLE Link Layer sees a stable oscillator to resume RF operations. However, the amount by which WAIT_OSC_STABLE duration exceeds this time, must be minimized, to avoid excessive power consumption.

The duration of WAIT_OSC_STABLE is determined by 2 BLE registers:

| BLE Register | Description |
|---|---|
| WAKEUP_CONFIG[OSC_STARTUP_DELAY[4:0] | This is the Fine Setting for the BLE Radio Power-On Offset and it is the number of 16.384 kHz clocks (61.04 us) to use as a Start Time offset prior to the BLE Deep Sleep Machine entering the Active State. |
| WAKEUP_CONFIG2[OSC_STARTUP_DELAY[11:5] | This is the Coarse Setting for the BLE Radio Power-On Offset and it is the number of BT Slots (625 us) to use as a Start Time offset prior to the BLE Deep Sleep Machine entering the Active State. |

Please refer to the BLE Link Layer chapters for additional information.

Note: The BLE Radio Power-On Offset must be Longer than the Radio Power-On Sequence. The RF Osc warm-up must complete before the BLE Deep Sleep Machine enters the Active State.

Note: In order to generate a 625 us average BT Slot time from a 32.768 kHz Osc clock there is a hardware method used to occasionally increase the number of 61.04 us clocks counted while creating the 625 us clock. This results in an uncertainty of one 16 kHz clock period while calculating the Coarse Setting for the BLE Radio Power-On Offset. Since this uncertainty is always positive and always increases the actual time of the BLE Radio Power-On Offset it can be neglected for purposes of ensuring that the BLE Radio Power-On Offset is Longer than the Radio Power-On Sequence.

The sum of the intervals determined by the values programmed into these 2 registers, represents the WAIT_OSC_STABLE duration.

**Example:**

- Configure BLE Link Layer Controller for an RSIM 3/4/5 DSM Wakeup (232 clocks):
  - RSIM 3/4/5 = COARSE + FINE + ACTIVE_WARNING + 2
  - RSIM 3/4/5 = (3 x 64) + 1 + (4 x 4) + 1 + (5 x 4) + 1 + 2 = 233
  - 233 / 32768 = 7.1105957ms
  - OSC_STARTUP_DELAY >= 7.1105957ms (constraint)
  - OSC_STARTUP_DELAY_SLOTS + OSC_STARTUP_DELAY_CLKS >= 7.1105957ms (constraint)
  - (11 x 0.625ms) + (6 x [2 / 32.768]) = 7.24121ms (contraint met)
- Software Example, DUT: Config RSIM for DSM
  - `ips3.w32(`RSIM_DSM_PWR_OFFSET, (3 << `COARSE_DELAY_3V) | (4 << `FINE_DELAY_3V) | (5 << `ACTIVE_WARNING_DELAY_3V));
- Software Example, DUT: Config BLE for DSM
  - dsm_offset_to_instant = 3;
  - osc_startup_delay_slots = 11;

- osc_startup_delay_clks = 6;
- `apb.w16(`WAKEUP_CONFIG, (dsm_offset_to_instant << 10) | (osc_startup_delay_clks << 0));
- `apb.w16(`WAKEUP_CONFIG2, (osc_startup_delay_slots << 0));

### 64.2.5.3 Deep Sleep Timer

When the Deep Sleep Timer is enabled, using the DSM_TIMER_EN bit, the DSM_TIMER register represents the 24-bit internal counter that counts the 32 kHz clock cycles. The DSM_TIMER register is cleared while the DSM_TIMER_CLR bit is set.

The Deep Sleep State Machines use the DSM_TIMER value to calculate sleep and wakeup times.

### 64.2.5.4 Deep Sleep Forced Exit

While the Deep Sleep State Machines are in sleep mode software can force a wakeup using the BLE_DSM_EXIT and WOR_DSM_EXIT register bits.

The State Machines will hold in their initial idle state until the corresponding EXIT register bits are cleared.

### 64.2.5.5 Deep Sleep State Machines

The RSIM registers have fields that are used by software to configure, control, and observe the Deep Sleep State Machines, these are described in the following tables.

**Table 64-5. Wake on Radio Deep Sleep State Machine Registers**

| RSIM Register Field | Function | Description |
|---|---|---|
| WOR_FSM_STATE | WOR Deep Sleep State Machine State | This is the current State that the WOR Deep Sleep Finite State Machine is in. |
| DSM_WOR_READY | WOR Ready for Deep Sleep Mode | This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |
| WOR_DEEP_SLEEP_STATUS | WOR Deep Sleep Mode Status | This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |

*Table continues on the next page...*

**Table 64-5. Wake on Radio Deep Sleep State Machine Registers (continued)**

| | | |
|---|---|---|
| DSM_WOR_FINISHED | WOR Deep Sleep Time Finished | This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| WOR_WAKEUP_REQUEST_EN | Enable WOR Deep Sleep Module to initiate a Radio Wakeup | This bit allows the WOR Deep Sleep Module to initiate a Radio Wakeup. |
| WOR_SLEEP_REQUEST | WOR Deep Sleep Requested | From the WOR Scheduler. When the signal is asserted WOR Deep Sleep Mode is entered. |
| WOR_WAKEUP_REQ | WOR Deep Sleep Module Radio Wakeup Status | This bit shows the status of the WOR Deep Sleep Module Request for the Radio to Wakeup. If the WOR_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| WOR_WAKEUP_INTERRUPT_EN | WOR Deep Sleep Module Radio Wakeup Interrupt Enable | If this bit is set, then a change from a 0 to a 1 on WOR_WAKEUP_REQ will generate an interrupt which can be used by the SoC. |
| WOR_WAKEUP_REQ_INT | Interrupt Flag from an WOR Deep Sleep Module Radio Wakeup | If the WOR_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an WOR Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| WOR_DSM_DURATION | WOR Deep Sleep Time Elapsed | The RSIM calculates the duration of the WOR deep sleep time and presents it here. |

**Table 64-6. Manual Deep Sleep State Machine Registers**

| RSIM Register Field | Function | Description |
|---|---|---|
| MAN_FSM_STATE | MAN Deep Sleep State Machine State | This is the current State that the MAN Deep Sleep Finite State Machine is in. |
| DSM_MAN_READY | MAN Ready for Deep Sleep Mode | This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |
| MAN_DEEP_SLEEP_STATUS | MAN Deep Sleep Mode Status | This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |
| DSM_MAN_FINISHED | MAN Deep Sleep Time Finished | This register is for debug purposes. This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| MAN_WAKEUP_REQUEST_EN | Enable MAN Deep Sleep Module to initiate a Radio Wakeup | This bit allows the MAN Deep Sleep Module to initiate a Radio Wakeup. |

*Table continues on the next page...*

**Table 64-6. Manual Deep Sleep State Machine Registers (continued)**

| | | |
|---|---|---|
| MAN_SLEEP_REQUEST | MAN Deep Sleep Requested | From the MAN Sleep Enable register. This enables a match of MAN_SLEEP[23:0] to SLEEP_TMR[23:0]; when the match occurs Deep Sleep Mode is entered. |
| MAN_WAKEUP_REQ | MAN Deep Sleep Module Radio Wakeup Status | This bit shows the status of the MAN Deep Sleep Module Request for the Radio to Wakeup. If the MAN_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| MAN_WAKEUP_INTERRUPT_EN | MAN Deep Sleep Module Radio Wakeup Interrupt Enable | If this bit is set, then a change from a 0 to a 1 on MAN_WAKEUP_REQ will generate an interrupt which can be used by the SoC. |
| MAN_WAKEUP_REQ_INT | Interrupt Flag from an MAN Deep Sleep Module Radio Wakeup | If the MAN_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an MAN Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| MAN_SLEEP_TIME | MAN Deep Sleep Module Sleep Time | Software determines the future time at which it would like to enter Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM start-time, and writing that value into this register. The value programmed into this register should be no fewer than 4 clocks greater (in the future) than the current time as read from DSM_TIMER. |
| MAN_WAKE_TIME | MAN Deep Sleep Module Wake Time | Software determines the future time at which it would like to exit Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM exit time, and writing that value into this register. |

## 64.2.6 Wi-Fi Coexistence

Hardware and Software methods allow the Radio to coexist in the same space as a Wi-Fi transceiver IC, which shares the same 2.4GHz frequency band. The coexistence scheme designates the Wi-Fi transceiver as the master, and the Radio as the slave. The objective of the coexistence strategy is to prevent both the Wi-Fi and Radio from transmitting simultaneously; a configuration option exists to also prevent the Radio from receiving when the Wi-Fi transceiver owns the RF channel.

A very basic mechanism is used in cases where the Radio does not generate requests, but instead defers to the Wi-Fi transceiver to grant permission to use the medium. The Wi-Fi IC generates a signal, RF_NOT_ALLOWED. If this signal is asserted, then the Radio does not perform any communication. When this signal is de-asserted, then the Radio is free to perform communications. In case the signal RF_NOT_ALLOWED is de-asserted when the Radio has already initiated the transmission/reception of a packet, then the Radio must stop its activity immediately .

Inside the Radio, all protocols are potentially affected by the coexistence strategy. However, coexistence-driven aborting can be enabled or disabled for the different protocols, under software control.

### 64.2.6.1 Wi-Fi Coexistence Pins

The Wi-Fi Coexistence Pins are described in the following table.

| Wi-Fi Coexist Function | Pin Direction | SoC Pin Name | Pin Description |
|---|---|---|---|
| RF_ACTIVE | O | RF_ACTIVE | An output which is asserted prior to any Radio event and remains asserted for the duration of the event. |
| RF_STATUS | O | RF_STATUS | An output which indicates when the Radio is in an RX or TX event, software can also control this signal directly. |
| RF_PRIORITY | O | RF_PRIORITY | An output which indicates to the external WiFi device that the Radio event is a high priority and it needs access to the 2.4GHz antenna. |
| RF_EARLY_WARNING | O | RF_EARLY_WARNING | BLE LL generated signal which can be used to wake an external sensor to make a measurement before a BLE event |
| RF_NOT_ALLOWED | I | RF_NOT_ALLOWED | External signal which causes the internal Radio to cease radio activity. |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| RF_TX_CONF | I | RF_TX_CONF | Signal from an external Radio which indicates the availability of the 2.4GHz antenna to the internal Radio. |

### 64.2.6.2 RF Not Allowed Register Bits

Several register bits have been provided in the XCVR_CTRL registers to enable, control, and provide status for RF_NOT_ALLOWED events. These bits are described in the following table.

| Field | R/W | Description |
|---|---|---|
| RF_NOT_ALLOWED_NO_TX | RW | 1: Allow assertions on RF_NOT_ALLOWED to abort a TSM TX sequence<br>0: Don't allow such aborts on TX |
| RF_NOT_ALLOWED_NO_RX | RW | 1: Allow assertions on RF_NOT_ALLOWED to abort a TSM RX sequence<br>0: Don't allow such aborts on RX |
| RF_NOT_ALLOWED_ASSERTED | R, Write a 1 to clear | 1: An assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared<br>0: No such assertion has occurred since the last time bit was cleared<br>**Note**: this bit will become set regardless of whether aborting on RF_NOT_ALLOWED events is enabled or not |
| RF_NOT_ALLOWED_TX_ABORT | R, Write a 1 to clear | 1: An assertion on RF_NOT_ALLOWED has aborted a TSM TX sequence since the last time this bit was cleared<br>0: No such TX abort has occurred since the last time bit was cleared |
| RF_NOT_ALLOWED_RX_ABORT | R | 1: An assertion on RF_NOT_ALLOWED has aborted a TSM RX sequence since the last time this bit was cleared<br>0: No such RX abort has occurred since the last time bit was cleared |
| RF_NOT_ALLOWED | R | Reflects the raw state of the RF_NOT_ALLOWED GPIO pin |

### 64.2.6.3 RF Active Register Bits

Several register bits have been provided in the RSIM registers to enable and control the RF_ACTIVE signaling. These bits are described in the following table.

| | | |
|---|---|---|
| ACTIVE_WARNING | Deep Sleep Wakeup RF Active Warning Time | This is the delay after RF OSC has been enabled, the range is 0-63 and the resulting delay is 122.07 us x ACTIVE_WARNING + 30.5 us. This is typically when the RF Osc should be stable. |

*Table continues on the next page...*

| | | |
|---|---|---|
| WIFI_COEXIST_1 | RF_ACTIVE Source | If this bit is set the RF_ACTIVE pin will be controlled by TSM Spare 1, otherwise it will be controlled by the RSIM RF Active Sequence along with any override by the SW_RF_ACTIVE_BIT. |
| WIFI_COEXIST_2 | RF_STATUS Source | If this bit is set the RF_STATUS pin will be controlled by TSM Spare 3, otherwise it will be controlled by TSM Spare 2. |
| WIFI_COEXIST_3 | RF_EARLY_WARNING Source | If this bit is set the RF_EARLY_WARNING pin will be controlled by TSM Spare 3, otherwise it will be controlled by the BLE Link Layer. |
| RF_ACTIVE_ENDS_WITH_TSM | RF_ACTIVE clearing mechanism | If this bit is set then the RSIM RF_ACTIVE OR source will clear when the TSM sequence ends. |
| SW_RF_ACTIVE_EN | Software RF_ACTIVE Control Enable | If this bit is set the SW_RF_ACTIVE_BIT acts as an OR source for the RF_ACTIVE pin. |
| SW_RF_ACTIVE_BIT | Software RF_ACTIVE Control Bit | If the SW_RF_ACTIVE_EN bit is set this bit acts as an OR source for the RF_ACTIVE pin. |
| SW_RF_ACTIVE_ENDS_WITH_TSM | Software RF_ACTIVE clearing mechanism | If this bit is set along with the SW_RF_ACTIVE_EN bit, then the Software RF_ACTIVE OR source will clear when the TSM sequence ends. |
| RSIM_RF_ACTIVE_OVRD | RF Active Internal Override | If the RSIM_RF_ACTIVE_OVRD_EN bit is set, then this bit controls the Internal version of RF Active instead of the Link Layers. The Internal version of RF Active whether overridden or not has a lower priority and can always be overridden by software using the SW_RF_ACTIVE_BIT or the TSM Spare 1 bit if it is selected using the WIFI_COEXIST_1 select bit. |
| RSIM_RF_ACTIVE_OVRD_EN | RF Active Internal Override Enable | This bit enables the RF Active Internal Override |
| RF_NOT_ALLOWED_OVRD | RF Not Allowed Override | If the RF_NOT_ALLOWED_OVRD_EN bit is set, then this bit controls the RF Not Allowed functionality instead of the SoC pin. |
| RF_NOT_ALLOWED_OVRD_EN | RF Not Allowed Override Enable | This bit enables the RF Not Allowed Override |

### 64.2.7 RF Early Warning

The BLE Link Layer can be programmed to provide an Early Warning signal that can assert up to 38.75 ms before related the Radio Operation begins. One use of this signal is to wakeup an external sensor so that it can collect and prepare any required data that needs to be available at the time of the Radio Operation.

Please refer to the BLE RF_ACTIVE_PERIOD register description for more details.

Note that this signal can be reconfigured as part of the Wi-Fi Coexistence Method as described in that section.

## 64.2.8 Radio Interrupts

The Radio can generate two asynchronous interrupts and one asynchronous MCU wakeup signal.

The interrupt configurations are illustrated in the diagram below and detailed in the following sections.

### 64.2.8.1 Interrupts Diagram

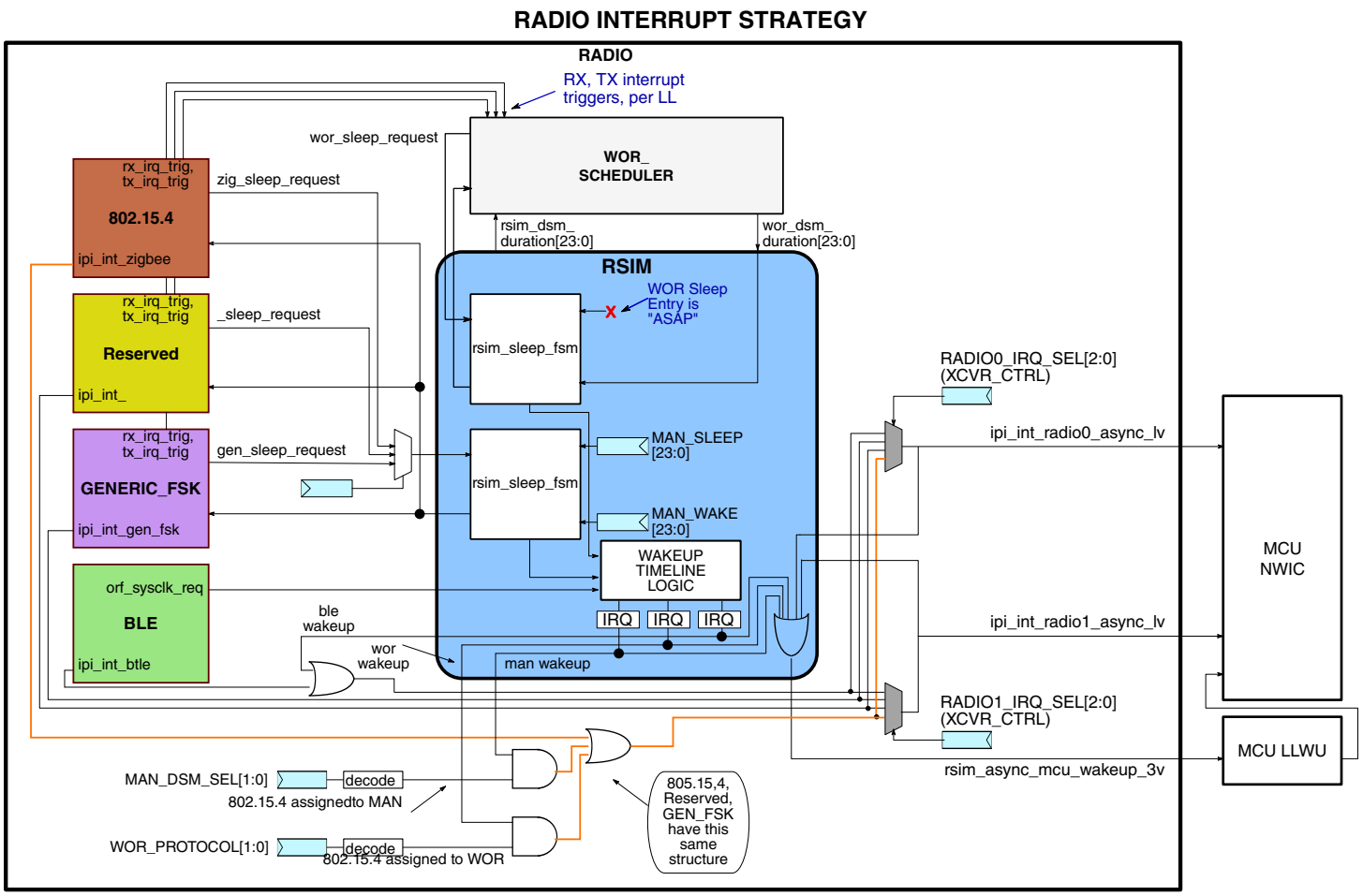



**Figure 64-10. Radio Interrupts**

### 64.2.8.2 Enabling Interrupts

There are three places that the interrupt strategy can be programmed:

1. The individual Link Layers interrupts may be configured.
2. The RSIM Deep Sleep Wakeup source may be configured.
3. The XCVR_CTRL register must be configured.

### 64.2.8.3 Deep Sleep Wakeups

Each Deep Sleep State Machine can generate a wakeup interrupt based upon this Wakeup Source selection.

The Radio Power-On Sequence can generate this wakeup interrupt at three times in the sequence, as shown in the following table:

| WAKEUP_INTERRUPT_SOURCE | Functionality |
|---|---|
| 0 | No Radio Power-On Sequence interrupt will be generated. |
| 1 | A Power-On Sequence interrupt will be generated when the RF Power Request occurs, including unblocked requests from an external source to use the RF OSC. |
| 2 | A Power-On Sequence interrupt will be generated when the RF OSC Request occurs, but not if the RF OSC request was from an external source. |
| 3 | A Power-On Sequence interrupt will be generated when the RSIM RF Active Warning occurs |

### 64.2.8.4 XCVR CTRL IRQ Select

Each of the Link Layer interrupts that can be selected here are an OR of the Link Layer interrupt, the WOR wakeup interrupt selected, or the MAN wakeup interrupt selected.

The two OR'd interrupts must be selected using the XCVR_CTRL RADIO0_IRQ_SEL and RADIO1_IRQ_SEL registers in the Transceiver as shown in the following tables:

Assign ipi_int_radio0_async_lv Interrupt (ipi_int_radio0) to a Protocol :

| RADIO0_IRQ_SEL | Functionality |
|---|---|
| 0 | Assign ipi_int_radio0_async_lv Interrupt to BLE |
| 1 | Assign ipi_int_radio0_async_lv Interrupt to 802.15.4 |
| 3 | Assign ipi_int_radio0_async_lv Interrupt to GENERIC_FSK |
| 2,4-7 | ipi_int_radio0_async_lv Interrupt unassigned |

Assign ipi_int_radio1_async_lv Interrupt (ipi_int_radio1) to a Protocol :

| RADIO1_IRQ_SEL | Functionality |
|---|---|
| 0 | Assign ipi_int_radio1_async_lv Interrupt to BLE |
| 1 | Assign ipi_int_radio1_async_lv Interrupt to 802.15.4 |
| 3 | Assign ipi_int_radio1_async_lv Interrupt to GENERIC_FSK |
| 2,4-7 | ipi_int_radio0_async_lv Interrupt unassigned |

### 64.2.8.5 MCU Wakeup

The RSIM Asynchronous MCU Wakeup signal is an OR of the enabled RADIO0_INTERRUPT, RADIO1_INTERRUPT, and the WAKEUP_INTERRUPT_SOURCE selections.

The control and observe bits for this wakeup source are shown in the following table:

| RSIM Register Field | Function | Description |
|---|---|---|
| RADIO0_INTERRUPT_EN | Radio0 Interrupt Enable | This bit enables the Radio0 interrupt source, as selected by the Radio0 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal. |
| RADIO1_INTERRUPT_EN | Radio1 Interrupt Enable | This bit enables the Radio1 interrupt source, as selected by the Radio1 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal. |
| BLE_RF_POWER_REQ_INT_EN | BLE RF Power Request Interrupt Enable | This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the BLE RF Power Request is asserted high. |
| BLE_RF_POWER_REQ_INT | BLE RF Power Request Interrupt Flag | This bit is the BLE_RF_POWER_REQ_INT interrupt flag. |
| WOR_WAKEUP_INTERRUPT_EN | WOR Deep Sleep Module Radio Wakeup Interrupt Enable | This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the WOR Wakeup Request is asserted high. |

*Table continues on the next page...*

| | | |
|---|---|---|
| WOR_WAKEUP_REQ_INT | Interrupt Flag from an WOR Deep Sleep Module Radio Wakeup | This bit is the WOR_WAKEUP_REQ_INT interrupt flag. |
| MAN_WAKEUP_INTERRUPT_EN | MAN Deep Sleep Module Radio Wakeup Interrupt Enable | This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the MAN Wakeup Request is asserted high. |
| MAN_WAKEUP_REQ_INT | Interrupt Flag from an MAN Deep Sleep Module Radio Wakeup | This bit is the MAN_WAKEUP_REQ_INT interrupt flag. |

## 64.2.9 Radio Configuration

The Radio Software Configuration (SW_CONFIG) register contains the software configuration and control bits. These bits can be written by software as part of configuring the Radio and then they can be read as needed to confirm that the Radio has not been reset by either a System Reset or a Power On event.

Please read the Radio Resets section for more information about the reset signals.

The configuration bits are described in the following table:

| RSIM Register Field | Function | Description |
|---|---|---|
| RADIO_CONFIGURED_POR_RESET | Radio Configuration Bit, cleared by Radio Power On Reset | This is a bit software can write when it wants to be able to check if a Radio Power On Reset occurs. |
| RADIO_CONFIGURED_SYS_RESET | Radio Configuration Bit, cleared by Radio System Reset | This is a bit software can write when it wants to be able to check if a Radio System Reset occurs. |
| BLOCK_SOC_RESETS | Block SoC Resets of the Radio, cleared by Radio System Reset | Most SoC System or Power On Resets will be blocked and the Radio will not be affected by them when this bit is set. The Radio Power Mode logic is not blocked by this bit and will reset on any SoC Power On Reset. |
| RADIO_POR_BIT | Software Power On Reset for the Radio. This bit only clears if there is an SoC Power On Reset. | Setting this bit will reset all the radio logic that uses the Radio Power On Reset or Radio VLLS Reset signal, it will be held in reset until this bit is cleared. |
| RADIO_RESET_BIT | Software System Reset for the Radio. This bit only clears if there is an SoC Power On Reset. | Setting this bit will reset the Global Radio and RSIM logic, including all Link Layer logic. The logic will stay in reset until this bit is cleared. This bit will not reset the Power Mode logic or the Deep Sleep Mode related logic. |

*Table continues on the next page...*

| | | |
|---|---|---|
| BLOCK_EXT_OSC_PWR_REQ | Block External Requests for RF OSC from starting a Radio Power Wakeup Sequence | If this bit is set then any External request for the OSC will not start a Radio Power Wakeup Sequence, otherwise any such request does request full Run Regulation mode. |

## 64.2.10 Radio Resets

The Radio can be reset by the SoC System Reset, the SoC Power On Reset, or software can block the SoC System Reset and instead perform a Radio Resets using register bits in the RSIM. Please read the Radio Configuration section for descriptions of those bits.

### 64.2.10.1 Reset Tree Diagram

The Radio Reset Tree is illustrated in the diagram below

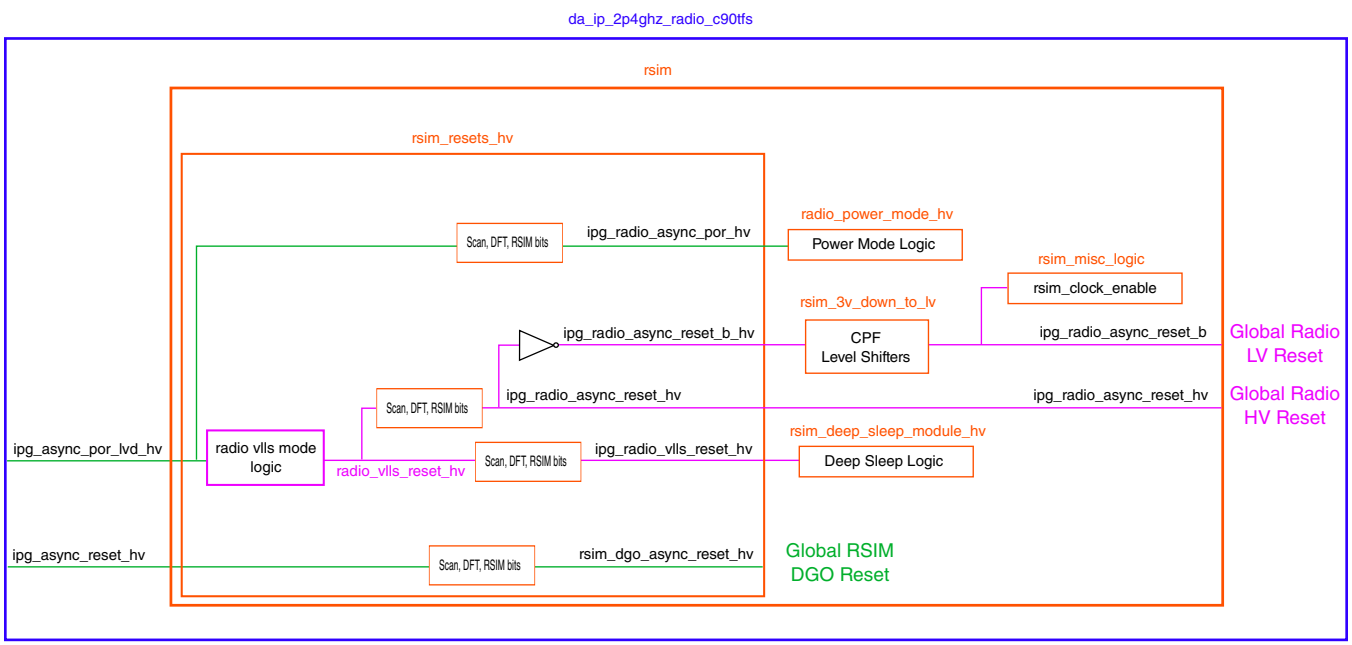



**Figure 64-11. Radio Reset Tree**

### 64.2.10.2 Blocking Resets

Software can choose to block SoC resets from reseting the Radio configuration and instead software can then monitor the SoC Resets, and also the Radio Configuration Status using the RADIO_CONFIGURED_SYS_RESET and RADIO_CONFIGURED_POR_RESET bits.

To use this method software should do the following:

- Set RSIM_SW_CONFIG_BLOCK_SOC_RESETS = 1 to start blocking SoC System Resets.
- Set RSIM_SW_CONFIG_RADIO_CONFIGURED_SYS_POR = 1 and monitor it to see if a POR has occurred, which will return all Radio settings to default.
- Monitor the SoC resets and if the Radio needs to be reset use the RSIM_SW_CONFIG_RADIO_RESET_BIT to reset the non-POR only logic.
- To return the entire Radio to the default settings use the RSIM_SW_CONFIG_RADIO_POR_BIT too.

### 64.2.10.3 Using Radio Reset Bits

In order to reset the Radio completely, software should toggle the RADIO_RESET_BIT first, and then toggle the RADIO_POR_BIT.

## 64.2.11 Radio Gasket Bypass

The SoC platform has an asynchronous gasket that allows register access for the Radio registers in all SoC clocking modes. The RADIO_GASKET_BYPASS_OVRD bit allows software to directly control the SoC platform asynchronous gasket bypass signal.

The default behavior of the SoC Asynchronous Gasket is Not Bypassed, which means the Radio registers are accessed using the RF Ref Osc clock. The Radio sends the RF Ref Osc to the asynchronous gasket which then uses that clock for SoC register accesses of the Radio registers. This requires the RF Ref Osc to be Enabled and Ready. If the RF Ref Osc is not Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, which forces the use of the SoC IPG clock as the register access clock. If the RF Ref Osc Is Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, but this is an ILLEGAL ACCESS if the Radio is operational, DO NOT write Radio registers with the Asynchronous Gasket Bypassed when the Radio is Operating. Test mode access is allowed but glitches may occur. The intent of Bypass Mode is to allow software to configure the Radio before the RF Ref Osc is Enabled and Ready. Note that the BLE Link Layer registers can only be accessed when the RF Ref Osc is Enabled and Ready.

Please also see the Radio Registers Access portion of the Radio Clocks section.

## 64.2.12 Radio Version ID

The RADIO_VERSION register value can be read to identify the Version of the Radio.

Note: If the Radio digital is in a low power stop mode the Version ID will read as zero.

## 64.2.13 Memory Map and Register Definition

### 64.2.13.1 RSIM register descriptions

#### 64.2.13.1.1 RSIM Memory map

RSIM base address: 4102_F000h

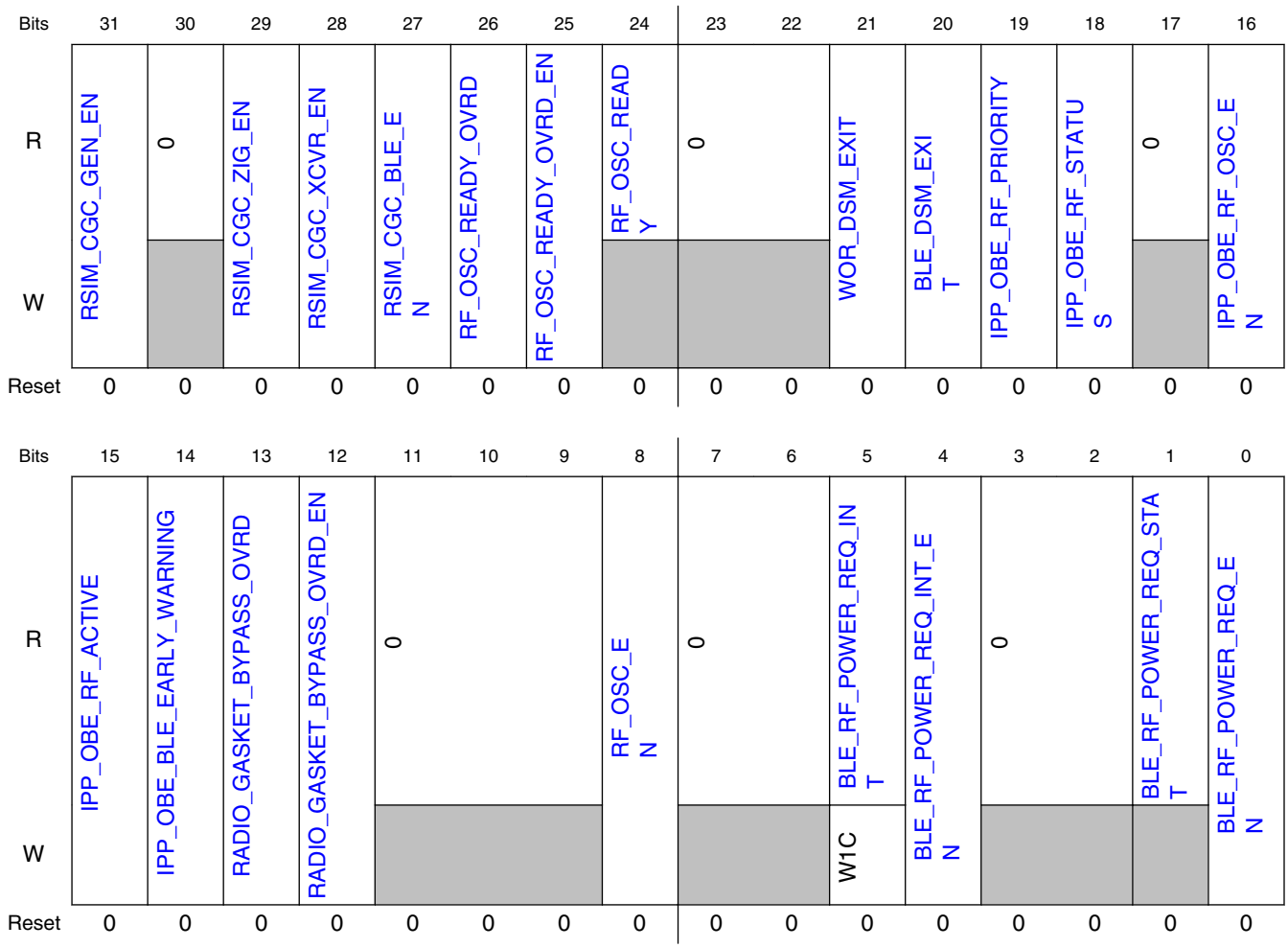
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Radio System Control (CONTROL) | 32 | RW | 0000_0000h |
| 10h | Radio Miscellaneous (MISC) | 32 | RW | 0500_0000h |
| 14h | RSIM Power Control (POWER) | 32 | RW | 7000_0307h |
| 18h | Radio Software Configuration (SW_CONFIG) | 32 | RW | 0000_0020h |
| 100h | Deep Sleep Timer (DSM_TIMER) | 32 | RO | 0000_0000h |
| 104h | Deep Sleep Timer Control (DSM_CONTROL) | 32 | RW | 0000_0000h |
| 108h | Deep Sleep Wakeup Sequence (DSM_WAKEUP) | 32 | RW | 0000_0000h |
| 10Ch | WOR Deep Sleep Duration (WOR_DURATION) | 32 | RO | 0000_0000h |
| 110h | WOR Deep Sleep Wake Time (WOR_WAKE) | 32 | RW | 0000_0000h |
| 11Ch | MAN Deep Sleep Time (MAN_SLEEP) | 32 | RW | 0000_0000h |
| 120h | MAN Deep Sleep Wake Time (MAN_WAKE) | 32 | RW | 0000_0000h |
| 124h | Radio Oscillator Control (RF_OSC_CTRL) | 32 | RW | A020_3806h |
| 128h | Radio Analog Test Registers (ANA_TEST) | 32 | RW | 0000_0000h |
| 12Ch | Radio Analog Trim Registers (ANA_TRIM) | 32 | RW | 784B_0000h |

#### 64.2.13.1.2 Radio System Control (CONTROL)

### 64.2.13.1.2.1 Offset

| Register | Offset |
|---|---|
| CONTROL | 0h |

### 64.2.13.1.2.2 Diagram




### 64.2.13.1.2.3 Fields

| Field | Function |
|---|---|
| 31<br>RSIM_CGC_GEN_EN | GEN Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the Generic Link Layer.<br>0b - Clock disabled<br>1b - Clock enabled |
| 30 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 29<br>RSIM_CGC_ZIG_EN | ZIG Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the 802.15.4 Link Layer.<br>0b - Clock disabled<br>1b - Clock enabled |
| 28<br>RSIM_CGC_XCVR_EN | XCVR Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the Transceiver.<br>0b - Clock disabled<br>1b - Clock enabled |
| 27<br>RSIM_CGC_BLE_EN | BLE Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the BLE Link Layer.<br>0b - Clock disabled<br>1b - Clock enabled |
| 26<br>RF_OSC_READY_OVRD | RF Ref Osc Ready Override<br>This bit directly controls the Radio RF Ref Osc Ready signal when the RF Ref Osc Ready Override is enabled. All Radio and SoC signals that are derived from the RF Ref Osc Ready signal can be overridden using this bit. |
| 25<br>RF_OSC_READY_OVRD_EN | RF Ref Osc Ready Override Enable<br>This bit enables the RF Ref Osc Ready Override bit. |
| 24<br>RF_OSC_READY | RF Ref Osc Ready<br>The RF Reference Oscillator has an internal counter that gates off the RF Ref Osc clock until the selected count is reached. This bit shows the status of the RF Ref Osc ready signal that comes from that counter. The RF Ref Osc Ready signal can be overridden using the RF_OSC_READY_OVRD bit. |
| 23-22<br>— | Reserved |
| 21<br>WOR_DSM_EXIT | Wake on Radio Force Deep Sleep Mode Exit<br>This bit forces the Wake on Radio State Machine to wakeup from Deep Sleep Mode. |
| 20<br>BLE_DSM_EXIT | BLE Force Deep Sleep Mode Exit<br>This bit forces the BLE link layer to wakeup from Deep Sleep Mode. |
| 19<br>IPP_OBE_RF_PRIORITY | IPP_OBE_RF_PRIORITY<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Priority signal as a pin interface option. |
| 18<br>IPP_OBE_RF_STATUS | IPP_OBE_RF_STATUS<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Status signal as a pin interface option. |
| 17<br>— | Reserved |
| 16<br>IPP_OBE_RF_OSC_EN | IPP_OBE_RF_OSC_EN<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Osc Enable signal as a pin interface option. |
| 15<br>IPP_OBE_RF_ACTIVE | IPP_OBE_RF_ACTIVE<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Active signal as a pin interface option. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 14<br>IPP_OBE_BLE_EARLY_WARNING | IPP_OBE_BLE_EARLY_WARNING<br>This bit enables the Output Driver (OBE) on the SoC port that provides the BLE Early Warning signal as a pin interface option. |
| 13<br>RADIO_GASKET_BYPASS_OVRD | Radio Gasket Bypass Override<br>This bit directly controls the SoC platform asynchronous gasket bypass signal when the Gasket Bypass Override is enabled.The default behavior of the SoC Asynchronous Gasket is Not Bypassed, which means the Radio registers are accessed using the RF Ref Osc clock. The Radio sends the RF Ref Osc to the asynchronous gasket which then uses that clock for SoC register accesses of the Radio registers. This requires the RF Ref Osc to be Enabled and Ready. If the RF Ref Osc is not Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, which forces the use of the SoC IPG clock as the register access clock. If the RF Ref Osc Is Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, but this is an ILLEGAL ACCESS if the Radio is operational, DO NOT write Radio registers with the Asynchronous Gasket Bypassed when the Radio is Operating. Test mode access is allowed but glitches may occur. The intent of Bypass Mode is to allow software to configure the Radio before the RF Ref Osc is Enabled and Ready. Note that the BLE Link Layer registers can only be accessed when the RF Ref Osc is Enabled and Ready.<br>0b - XCVR and Link Layer Register Clock is the RF Ref Osc Clock<br>1b - XCVR and Link Layer Register Clock is the SoC IPG Clock |
| 12<br>RADIO_GASKET_BYPASS_OVRD_EN | Radio Gasket Bypass Override Enable<br>The SoC platform has an asynchronous gasket that allows register access for the Radio registers in all SoC clocking modes. This bit allows software to directly control the SoC platform asynchronous gasket bypass signal. |
| 11-9<br>— | Reserved |
| 8<br>RF_OSC_EN | RF Ref Osc Enable<br>The RF Reference Oscillator can be enabled by a Radio link layer (Including the Link Layer use of Deep Sleep Modes), by an internal SoC clock mode, by an External Pin request, or by this bit.<br>If this bit is 0, then in Radio Run Mode, the RF Ref Osc can be turned on by the SoC, or by a Radio link layer using RRun Mode (Including the Link Layer use of Deep Sleep Modes), or by an External Pin request. Warning: The RF Ref Osc can not be enabled by the SoC until the Radio has entered RRun Mode.<br>If this bit is 0, then in Radio Stop Modes, the RF Ref Osc will be off (except that the external pin request can turn the RF Osc on even if the RF Power is off). Warning: Radio will not go into Stop mode while the RF Ref Osc is being used by the SoC, or by a Radio link layer unless the Link Layer is using a Deep Sleep State Machine.<br>If this bit is 1, then the RF Ref Osc will be on in Radio Run Mode. Warning: The RF Ref Osc can not be enabled until the Radio has entered RRun Mode.<br>If this bit is 1, then the RF Ref Osc will be off in Radio Stop Modes (except that the external pin request can turn the RF Osc on even if the RF Power is off). Warning: Radio will not go into Stop mode while the RF Ref Osc is On. |
| 7-6<br>— | Reserved |
| 5<br>BLE_RF_POWER_REQ_INT | BLE RF Power Request Interrupt Flag<br>This bit is the BLE_RF_POWER_REQ_INT interrupt flag.<br>This interrupt flag is cleared by writing a 1 to it. |
| 4 | BLE RF Power Request Interrupt Enable |

*Table continues on the next page...*

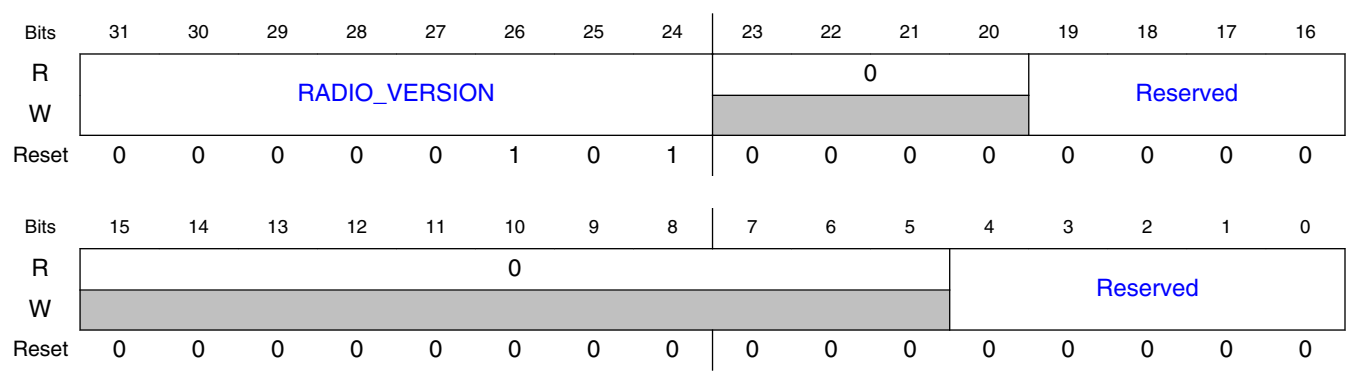
| Field | Function |
|---|---|
| BLE_RF_POWER_REQ_INT_EN | This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the BLE RF Power Request is asserted high. |
| 3-2<br>— | Reserved |
| 1<br>BLE_RF_POWER_REQ_STAT | BLE RF Power Request Status<br>This bit indicates the current status of the BLE link layer request to turn on the RF Power. |
| 0<br>BLE_RF_POWER_REQ_EN | BLE RF Power Request Enable<br>This bit resets on POR only.<br>If this bit is cleared (the default state), then all BLE link layer requests to turn on the RF Power will be blocked and ignored.<br>In BLE protocols the BLE link layer will always restart when exiting reset by first Requesting the RF Power, this bit blocks that behavior until software configures the Radio and enables the requests. |

## 64.2.13.1.3 Radio Miscellaneous (MISC)

### 64.2.13.1.3.1 Offset

| Register | Offset |
|---|---|
| MISC | 10h |

### 64.2.13.1.3.2 Diagram




### 64.2.13.1.3.3 Fields

| Field | Function |
|---|---|
| 31-24 | Radio Version ID number |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RADIO_VERSION | This register value can be read to identify the Version of the Radio. Note that the value will read as zero unless the Radio is in Run Mode.<br>B = 2.4 GHz Radio 2.0<br>8 = 2.4 GHz Radio 2.1<br>4 = 2.4 GHz Radio 3.0<br>5 = 2.4 GHz Radio 3.1 |
| 23-20<br>— | Reserved |
| 19-16<br>— | Reserved |
| 15-5<br>— | Reserved |
| 4-0<br>— | Reserved |

## 64.2.13.1.4 RSIM Power Control (POWER)

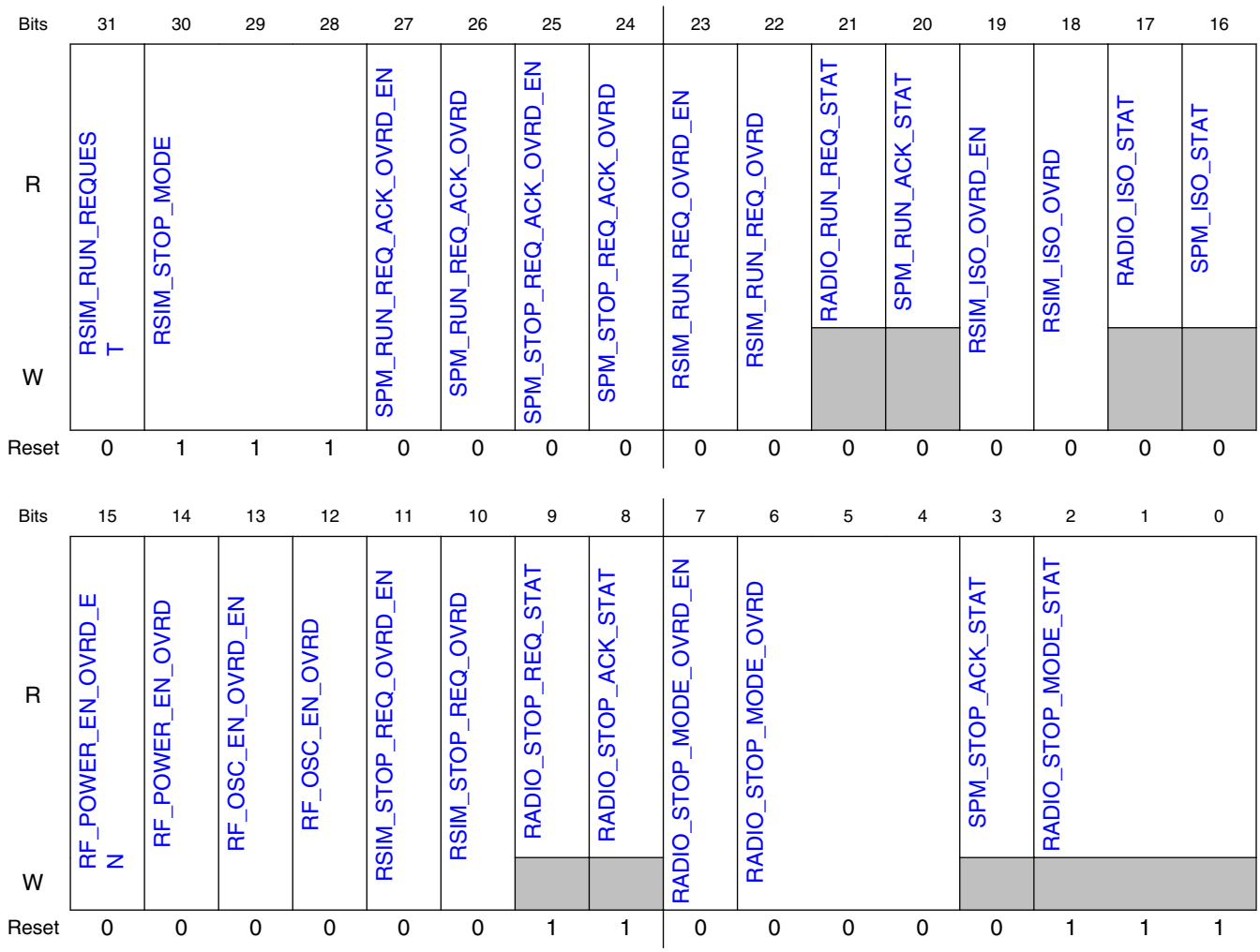
### 64.2.13.1.4.1 Offset

| Register | Offset |
|---|---|
| POWER | 14h |

### 64.2.13.1.4.2 Function

This register contains the power control and status bits.

### 64.2.13.1.4.3 Diagram




### 64.2.13.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>RSIM_RUN_REQUEST | RSIM Run Regulator Request<br>When this bit is set, the RSIM will request the SPM Run Regulator to be turned on and the RSIM will then stop requesting Stop after the Run Regulator Acknowledge signal is received from the SPM module. |
| 30-28<br>RSIM_STOP_MODE | RSIM lowest allowed Stop Mode<br>This is the Stop Mode that will be requested when the RSIM asserts a Stop request to the SPM module.<br>000b - Reserved<br>001b - Reserved<br>011b - RLLS mode (Radio State Retention mode)<br>111b - RVLLS mode (This is the POR setting) |
| 27 | SPM Run Request Acknowledge Override Enable<br>This bit enables the SPM Run Request Acknowledge Override bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SPM_RUN_REQ_ACK_OVRD_EN | |
| 26<br>SPM_RUN_REQ_ACK_OVRD | SPM Run Request Acknowledge Override<br>This bit directly controls the SPM Run Request Acknowledge signal when the SPM Run Request Acknowledge Override is enabled. |
| 25<br>SPM_STOP_REQ_ACK_OVRD_EN | SPM Stop Request Acknowledge Override Enable<br>This bit enables the SPM Stop Request Acknowledge Override bit. |
| 24<br>SPM_STOP_REQ_ACK_OVRD | SPM Stop Request Acknowledge Override<br>This bit directly controls the SPM Stop Request Acknowledge signal when the SPM Stop Request Acknowledge Override is enabled. |
| 23<br>RSIM_RUN_REQ_OVRD_EN | RSIM Run Request Override Enable<br>This bit enables the RSIM Run Request Override bit. |
| 22<br>RSIM_RUN_REQ_OVRD | RSIM Run Request Override<br>This bit directly controls the RSIM Run Request signal to the SPM when the RSIM Run Request Override is enabled. |
| 21<br>RADIO_RUN_REQ_STAT | Radio Run Request Status<br>This is the value of the Radio Run Request signal being sent to the SPM module, including any overrides. |
| 20<br>SPM_RUN_ACK_STAT | SPM Run Request Acknowledge Status<br>This is the value of the SPM Run request Acknowledge signal being sent to the Radio. |
| 19<br>RSIM_ISO_OVRD_EN | RSIM ISO Override Enable<br>This bit enables the RSIM ISO Override bit. |
| 18<br>RSIM_ISO_OVRD | RSIM ISO Override<br>This bit directly controls the Radio ISO signal when the RSIM ISO Override is enabled. |
| 17<br>RADIO_ISO_STAT | Radio Isolation Status<br>This is the value of the RSIM ISO signal being sent to the Radio Isolation logic, including any overrides. |
| 16<br>SPM_ISO_STAT | SPM ISO Status<br>This is the value of the SPM ISO signal being sent to the Radio. |
| 15<br>RF_POWER_EN_OVRD_EN | Radio Power Enable Override Enable<br>This bit enables the Radio Power Enable Override bit. |
| 14<br>RF_POWER_EN_OVRD | Radio Power Enable Override<br>This bit directly controls the RF_POWER_EN signal when the Radio Power Enable Override is enabled. |
| 13 | Radio Osc Enable Override Enable<br>This bit enables the Radio Osc Enable Override bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RF_OSC_EN_OVRD_EN | |
| 12<br>RF_OSC_EN_OVRD | Radio Osc Enable Override<br>This bit directly controls the RF_OSC_EN SoC pin signal when the Radio Osc Enable Override is enabled. |
| 11<br>RSIM_STOP_REQ_OVRD_EN | Radio Stop Request Override Enable<br>This bit enables the Radio Stop Request Override bit. |
| 10<br>RSIM_STOP_REQ_OVRD | Radio Stop Request Override<br>This bit directly controls the Radio Stop Request signal when the Radio Stop Request Override is enabled. |
| 9<br>RADIO_STOP_REQ_STAT | Radio Stop Request Status<br>This is the value of the Radio Stop Request signal being sent to the SPM module, including any overrides. |
| 8<br>RADIO_STOP_ACK_STAT | Radio Stop Acknowledge Status<br>This is the status of the Stop request Acknowledge signal being used by the Radio, including any override of the SPM module signal. |
| 7<br>RADIO_STOP_MODE_OVRD_EN | Radio Stop Mode Override Enable<br>This bit enables the Radio Stop Mode Override bit. |
| 6-4<br>RADIO_STOP_MODE_OVRD | Radio Stop Mode Override<br>This bit directly controls the Radio Stop Mode signal when the Radio Stop Mode Override is enabled. |
| 3<br>SPM_STOP_ACK_STAT | SPM Stop Acknowledge Status<br>This is the status of the Stop request Acknowledge signal received from the SPM module. |
| 2-0<br>RADIO_STOP_MODE_STAT | Radio Stop Mode Status<br>This is the value of the Radio Stop Mode bus that is sent to the SPM module. |

## 64.2.13.1.5 Radio Software Configuration (SW_CONFIG)

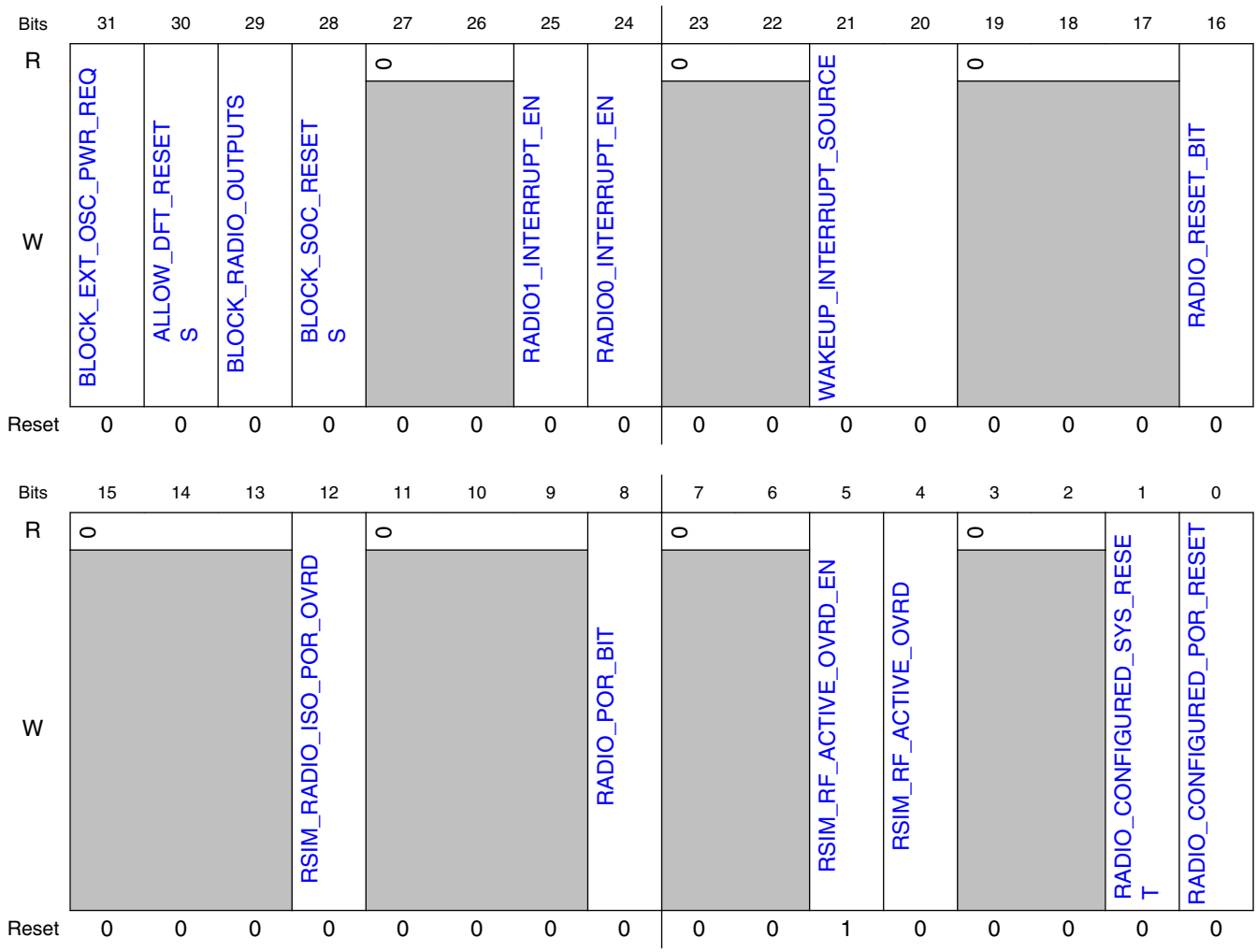
### 64.2.13.1.5.1 Offset

| Register | Offset |
|---|---|
| SW_CONFIG | 18h |

### 64.2.13.1.5.2 Function

This register contains the software configuration and control bits.

### 64.2.13.1.5.3 Diagram




### 64.2.13.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>BLOCK_EXT_OSC_PWR_REQ | Block External Requests for RF OSC from starting a Radio Power Wakeup Sequence<br>If this bit is set then any External request for the OSC will not start a Radio Power Wakeup Sequence, otherwise any such request does request full Run Regulation mode. |
| 30<br>ALLOW_DFT_RESETS | Allow the DFT Reset Pin to Reset the Radio<br>The Radio provides a port that can be connected to a test mode pin in order to provide a method of resetting the Radio independently from SoC resets. This bit enables that DFT functionality. |
| 29<br>BLOCK_RADIO_OUTPUTS | Block Radio Outputs<br>This bit resets on POR only. This bit is intended to allow the SoC to perform concurrent testing of various SoC logic while the Radio is operating independently. Any Radio output signals that go to the SoC will be blocked so as to not affect the SoC testing when this bit is set. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 28<br>BLOCK_SOC_RESETS | Block SoC Resets of the Radio, cleared by Radio System Reset<br>Most SoC System or Power On Resets will be blocked and the Radio will not be affected by them when this bit is set.<br>The Radio Power Mode logic is not blocked by this bit and will reset on any SoC Power On Reset. |
| 27-26<br>— | Reserved |
| 25<br>RADIO1_INTERRUPT_EN | Radio1 Interrupt Enable<br>This bit enables the Radio1 interrupt source, as selected by the Radio1 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal (typically connected to the MCU LLWU). |
| 24<br>RADIO0_INTERRUPT_EN | Radio0 Interrupt Enable<br>This bit enables the Radio0 interrupt source, as selected by the Radio0 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal (typically connected to the MCU LLWU). |
| 23-22<br>— | Reserved |
| 21-20<br>WAKEUP_INTERRUPT_SOURCE | RSIM Wakeup Interrupt Source Selector<br>The Radio Power-On Sequence can generate an interrupt at three times in the sequence, these bits make that selection.<br>00b - No Radio Power-On Sequence interrupt will be generated.<br>01b - A Power-On Sequence interrupt will be generated when the RF Power Request occurs, including unblocked requests from an external source to use the RF OSC.<br>10b - A Power-On Sequence interrupt will be generated when the RF OSC Request occurs, but not if the RF OSC request was from an external source.<br>11b - A Power-On Sequence interrupt will be generated when the RSIM RF Active Warning occurs |
| 19-17<br>— | Reserved |
| 16<br>RADIO_RESET_BIT | Software System Reset for the Radio<br>This bit only clears if there is an SoC Power On Reset.<br>Setting this bit will reset the Global Radio and RSIM logic, including all Link Layer logic.<br>The logic will stay in reset until this bit is cleared.<br>This bit will not reset the Power Mode logic or the Deep Sleep Mode related logic. |
| 15-13<br>— | Reserved |
| 12<br>RSIM_RADIO_ISO_POR_OVRD | RSIM ISO_POR Override<br>This bit directly controls the Radio ISO_POR signal when the RSIM_ISO_OVRD_EN bit is set. |
| 11-9<br>— | Reserved |
| 8<br>RADIO_POR_BIT | Software Power On Reset for the Radio<br>This bit only clears if there is an SoC Power On Reset.<br>Setting this bit will reset all the radio logic that uses the Radio Power On Reset or Radio VLLS Reset signal, it will be held in reset until this bit is cleared. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-6<br>— | Reserved |
| 5<br>RSIM_RF_ACTIVE_OVRD_EN | RF Active Internal Override Enable<br>This bit enables the RF Active Internal Override bit. |
| 4<br>RSIM_RF_ACTIVE_OVRD | RF Active Internal Override<br>If the RSIM_RF_ACTIVE_OVRD_EN bit is set, then this bit controls the Internal version of RF Active instead of the Link Layers. The Internal version of RF Active whether overridden or not has a lower priority and can always be overridden by software using the SW_RF_ACTIVE_BIT or the TSM Spare 1 bit if it is selected using the WIFI_COEXIST_1 select bit. |
| 3-2<br>— | Reserved |
| 1<br>RADIO_CONFIGURED_SYS_RESET | Radio Configuration Bit, cleared by Radio System Reset<br>This is a bit software can write when it wants to be able to check if a Radio System Reset occurs. |
| 0<br>RADIO_CONFIGURED_POR_RESET | Radio Configuration Bit, cleared by Radio Power On Reset<br>This is a bit software can write when it wants to be able to check if a Radio Power On Reset occurs. |

## 64.2.13.1.6 Deep Sleep Timer (DSM_TIMER)

### 64.2.13.1.6.1 Offset

| Register | Offset |
|---|---|
| DSM_TIMER | 100h |

### 64.2.13.1.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | DSM_TIMER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DSM_TIMER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.2.13.1.6.3 Fields

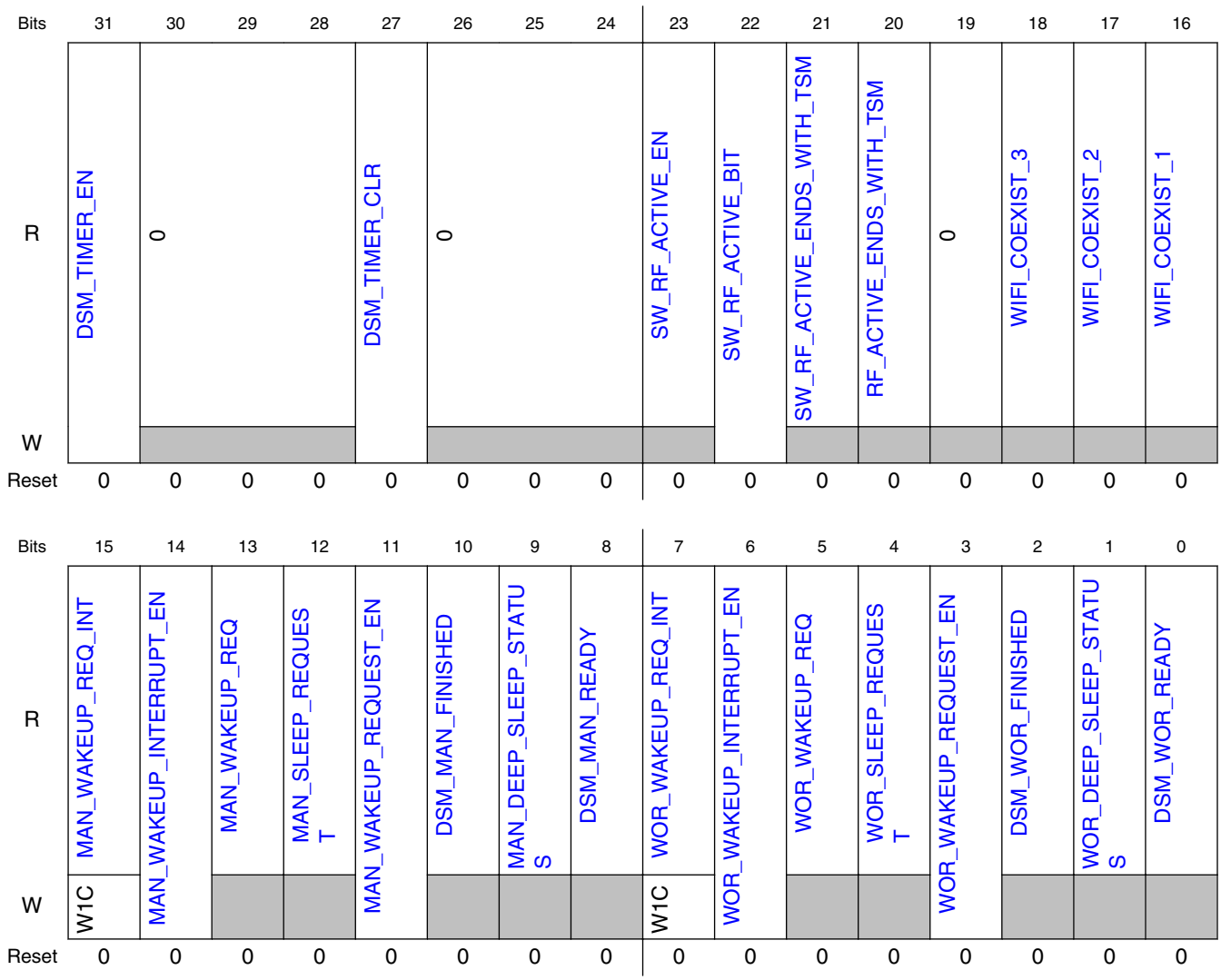
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>DSM_TIMER | Deep Sleep Mode Timer<br>This read only register shows the value of the Deep Sleep Timer. The timer counts clock cycles of the 32 kHz SoC oscillator whenever the DSM_TIMER_EN bit is set. The timer is reset to zero whenever the DSM_TIMER_CLR bit is set. |

### 64.2.13.1.7 Deep Sleep Timer Control (DSM_CONTROL)

#### 64.2.13.1.7.1 Offset

| Register | Offset |
|---|---|
| DSM_CONTROL | 104h |

## 64.2.13.1.7.2 Diagram




## 64.2.13.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>DSM_TIMER_EN | Deep Sleep Mode Timer Enable<br>Whenever this bit is set the Deep Sleep Mode Timer counts clock cycles of the 32 kHz SoC oscillator. |
| 30-28<br>— | Reserved |
| 27<br>DSM_TIMER_CLR | Deep Sleep Mode Timer Clear<br>Whenever this bit is set the Deep Sleep Mode Timer is reset to zero. |
| 26-24 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 23<br>SW_RF_ACTIVE_EN | Software RF_ACTIVE Control Enable<br>If this bit is set the SW_RF_ACTIVE_BIT acts as an OR source for the RF_ACTIVE pin. |
| 22<br>SW_RF_ACTIVE_BIT | Software RF_ACTIVE Control Bit<br>If the SW_RF_ACTIVE_EN bit is set this bit acts as an OR source for the RF_ACTIVE pin. |
| 21<br>SW_RF_ACTIVE_ENDS_WITH_TSM | Software RF_ACTIVE clearing mechanism<br>If this bit is set along with the SW_RF_ACTIVE_EN bit, then the Software RF_ACTIVE OR source will clear when the TSM sequence ends. |
| 20<br>RF_ACTIVE_ENDS_WITH_TSM | RF_ACTIVE clearing mechanism<br>If this bit is set then the RSIM RF_ACTIVE OR source will clear when the TSM sequence ends. |
| 19<br>— | Reserved |
| 18<br>WIFI_COEXIST_3 | RF_EARLY_WARNING Source<br>If this bit is set the RF_EARLY_WARNING pin will be controlled by TSM Spare 3, otherwise it will be controlled by the BLE Link Layer. |
| 17<br>WIFI_COEXIST_2 | RF_STATUS Source<br>If this bit is set the RF_STATUS pin will be controlled by TSM Spare 3, otherwise it will be controlled by TSM Spare 2. |
| 16<br>WIFI_COEXIST_1 | RF_ACTIVE Source<br>If this bit is set the RF_ACTIVE pin will be controlled by TSM Spare 1, otherwise it will be controlled by the RSIM RF Active Sequence along with any override by the SW_RF_ACTIVE_BIT. |
| 15<br>MAN_WAKEUP_REQ_INT | Interrupt Flag from an MAN Deep Sleep Module Radio Wakeup<br>If the MAN_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an MAN Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| 14<br>MAN_WAKEUP_INTERRUPT_EN | MAN Deep Sleep Module Radio Wakeup Interrupt Enable<br>This bit enables an interrupt request, that is also added to the RSIM of the WOR Wakeup Request is asserted high. Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version |
| 13<br>MAN_WAKEUP_REQ | MAN Deep Sleep Module Radio Wakeup Status<br>This bit shows the status of the MAN Deep Sleep Module Request for the Radio to Wakeup. If the MAN_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| 12<br>MAN_SLEEP_REQUEST | MAN Deep Sleep Requested<br>From the MAN Sleep Enable register. This enables a match of MAN_SLEEP[23:0] to SLEEP_TMR[23:0]; when the match occurs Deep Sleep Mode is entered.<br>Warning: This bit will read as X if the Radio is not in RRun mode. |
| 11<br>MAN_WAKEUP_REQUEST_EN | Enable MAN Deep Sleep Module to initiate a Radio Wakeup<br>This bit allows the MAN Deep Sleep Module to initiate a Radio Wakeup. |
| 10 | MAN Deep Sleep Time Finished |

*Table continues on the next page...*

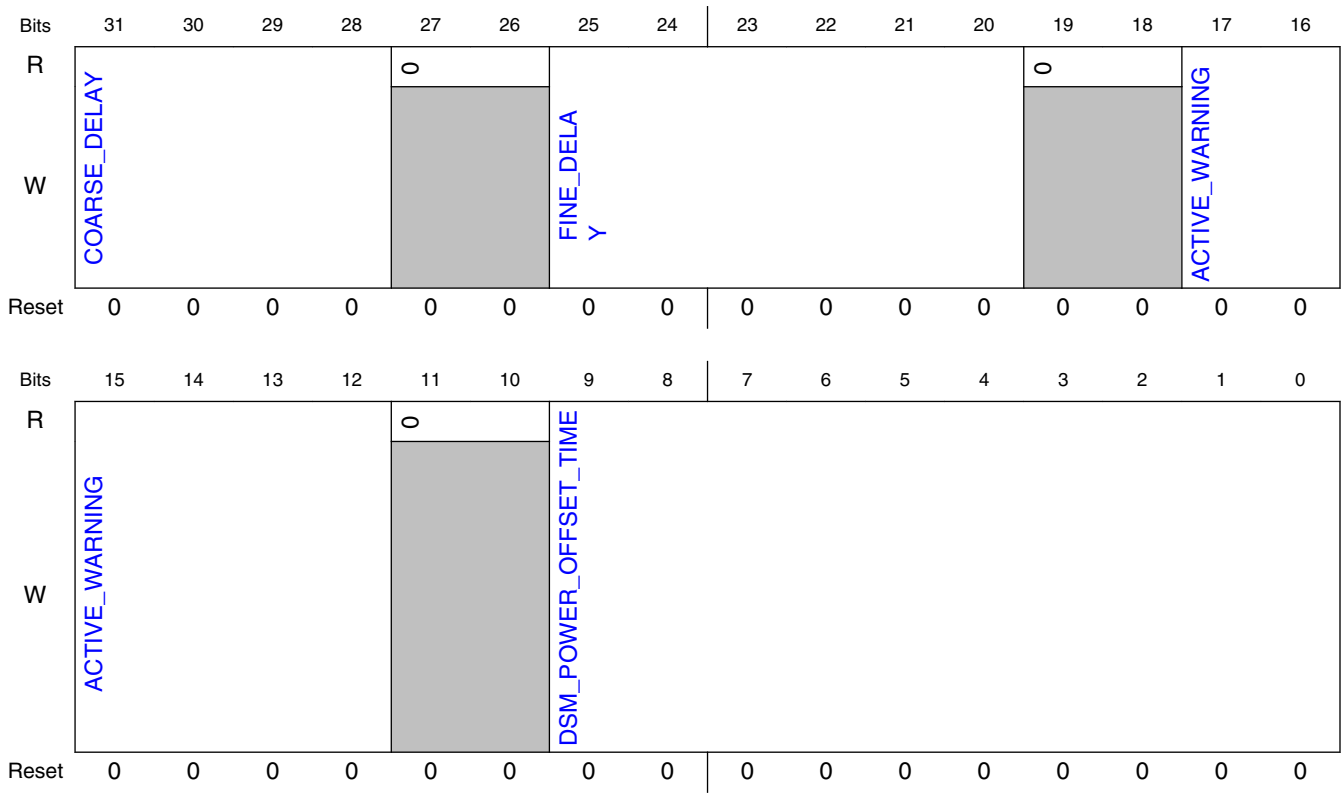
| Field | Function |
|---|---|
| DSM_MAN_FINISHED | This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| 9<br>MAN_DEEP_SLEEP_STATUS | MAN Deep Sleep Mode Status<br>This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |
| 8<br>DSM_MAN_READY | MAN Ready for Deep Sleep Mode<br>This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |
| 7<br>WOR_WAKEUP_REQ_INT | Interrupt Flag from an WOR Deep Sleep Module Radio Wakeup<br>If the WOR_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an WOR Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| 6<br>WOR_WAKEUP_INTERRUPT_EN | WOR Deep Sleep Module Radio Wakeup Interrupt Enable<br>This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the WOR Wakeup Request is asserted high. |
| 5<br>WOR_WAKEUP_REQ | WOR Deep Sleep Module Radio Wakeup Status<br>This bit shows the status of the WOR Deep Sleep Module Request for the Radio to Wakeup. If the WOR_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| 4<br>WOR_SLEEP_REQUEST | WOR Deep Sleep Requested<br>From the WOR Sleep Enable register. This enables a match of WOR_SLEEP[23:0] to SLEEP_TMR[23:0]; when the match occurs Deep Sleep Mode is entered.<br>**warning:** This bit will read as X if the Radio is not in RRun mode. |
| 3<br>WOR_WAKEUP_REQUEST_EN | Enable WOR Deep Sleep Module to initiate a Radio Wakeup<br>This bit allows the WOR Deep Sleep Module to initiate a Radio Wakeup. |
| 2<br>DSM_WOR_FINISHED | WOR Deep Sleep Time Finished<br>This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| 1<br>WOR_DEEP_SLEEP_STATUS | WOR Deep Sleep Mode Status<br>This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |
| 0<br>DSM_WOR_READY | WOR Ready for Deep Sleep Mode<br>This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |

## 64.2.13.1.8 Deep Sleep Wakeup Sequence (DSM_WAKEUP)

### 64.2.13.1.8.1 Offset

| Register | Offset |
|---|---|
| DSM_WAKEUP | 108h |

### 64.2.13.1.8.2 Diagram




### 64.2.13.1.8.3 Fields

| Field | Function |
|---|---|
| 31-28<br>COARSE_DELAY | Deep Sleep Wakeup Coarse Delay Time<br>This is the first delay after the beginning of the Power-On sequence, the range is 0-15 and the resulting delay is 1.953 ms x COARSE_DELAY + 61 us, with a minimum delay of 61 us. |
| 27-26<br>— | Reserved |
| 25-20<br>FINE_DELAY | Deep Sleep Wakeup Fine Delay Time<br>This is the second delay after the beginning of the Power-On sequence, the range is 0-63 and the resulting delay is 122.07 us x FINE_DELAY + 61 us, with a minimum delay of 61 us. This is when the RF OSC is enabled and so the Radio Analog Power must be stable at this milestone. |
| 19-18<br>— | Reserved |
| 17-12<br>ACTIVE_WARNING | Deep Sleep Wakeup RF Active Warning Time<br>This is the delay after RF OSC has been enabled, the range is 0-63 and the resulting delay is 122.07 us x ACTIVE_WARNING + 30.5 us. This is typically when the RF Osc should be stable. |
| 11-10<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 9-0<br>DSM_POWER_OFFSET_TIME | Deep Sleep Wakeup Power Offset Time<br>This register is the calculated Radio Power-On Sequence Time. |

## 64.2.13.1.9 WOR Deep Sleep Duration (WOR_DURATION)

### 64.2.13.1.9.1 Offset

| Register | Offset |
|---|---|
| WOR_DURATION | 10Ch |

### 64.2.13.1.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WOR_DSM_DURATION | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WOR_DSM_DURATION | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.2.13.1.9.3 Fields

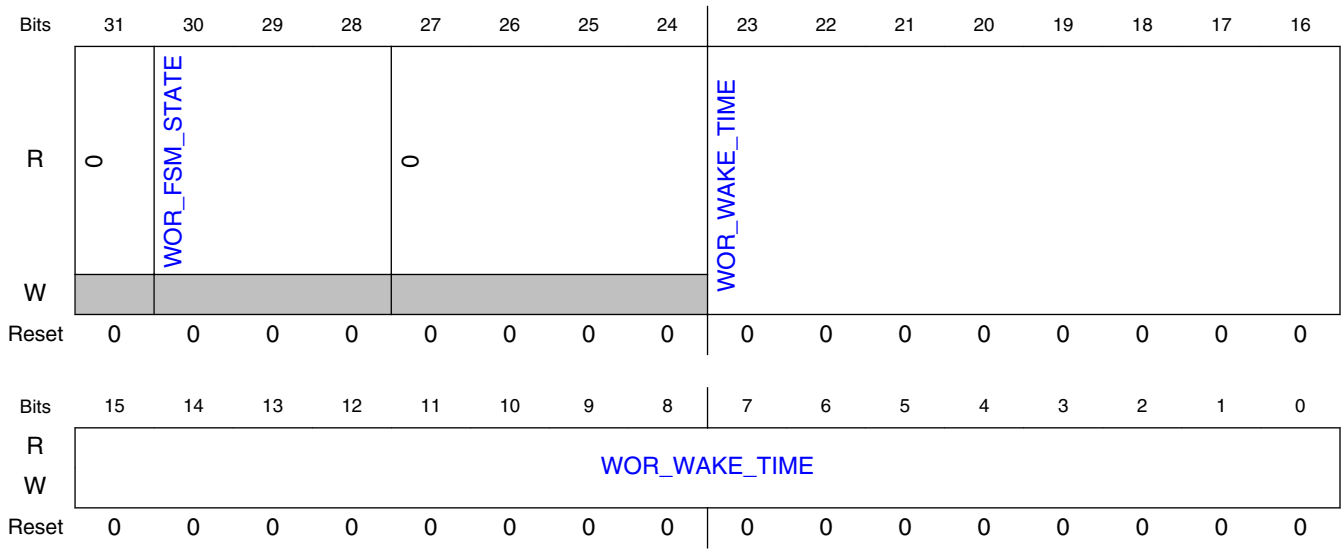
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>WOR_DSM_DURATION | WOR Deep Sleep Time Elapsed<br>The RSIM calculates the duration of the WOR deep sleep time and presents it here. |

## 64.2.13.1.10 WOR Deep Sleep Wake Time (WOR_WAKE)

### 64.2.13.1.10.1 Offset

| Register | Offset |
|---|---|
| WOR_WAKE | 110h |

### 64.2.13.1.10.2 Diagram




### 64.2.13.1.10.3 Fields

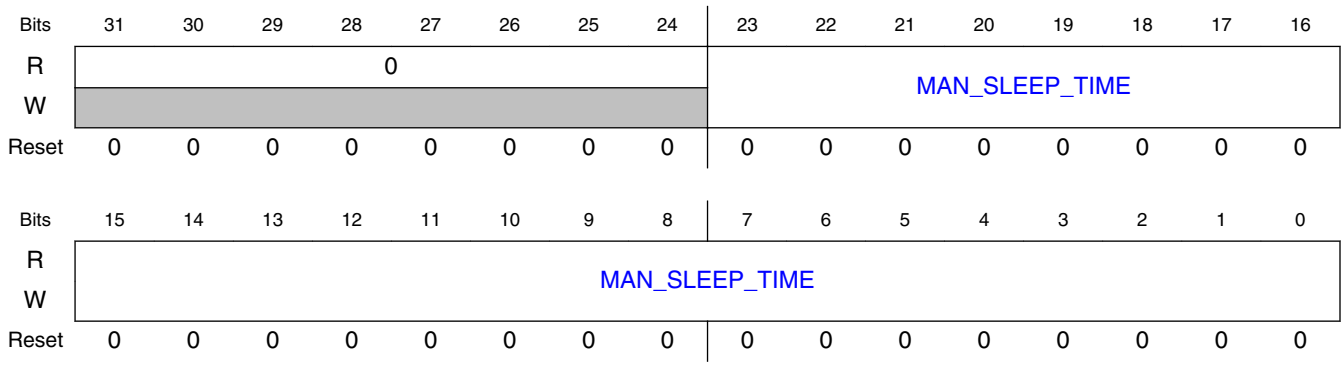
| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>WOR_FSM_STATE | WOR Deep Sleep State Machine State<br>This is the current State that the WOR Deep Sleep Finite State Machine is in. |
| 27-24<br>— | Reserved |
| 23-0<br>WOR_WAKE_TIME | WOR Deep Sleep Module Wake Time<br>Hardware determines the future time at which to exit Deep Sleep Mode by reading the DSM_TIMER and computing the number of 32 kHz clock cycles needed to match the desired DSM Duration. The calculated value of this Wake Time is shown in this register. |

## 64.2.13.1.11 MAN Deep Sleep Time (MAN_SLEEP)

### 64.2.13.1.11.1 Offset

| Register | Offset |
|---|---|
| MAN_SLEEP | 11Ch |

### 64.2.13.1.11.2 Diagram




### 64.2.13.1.11.3 Fields

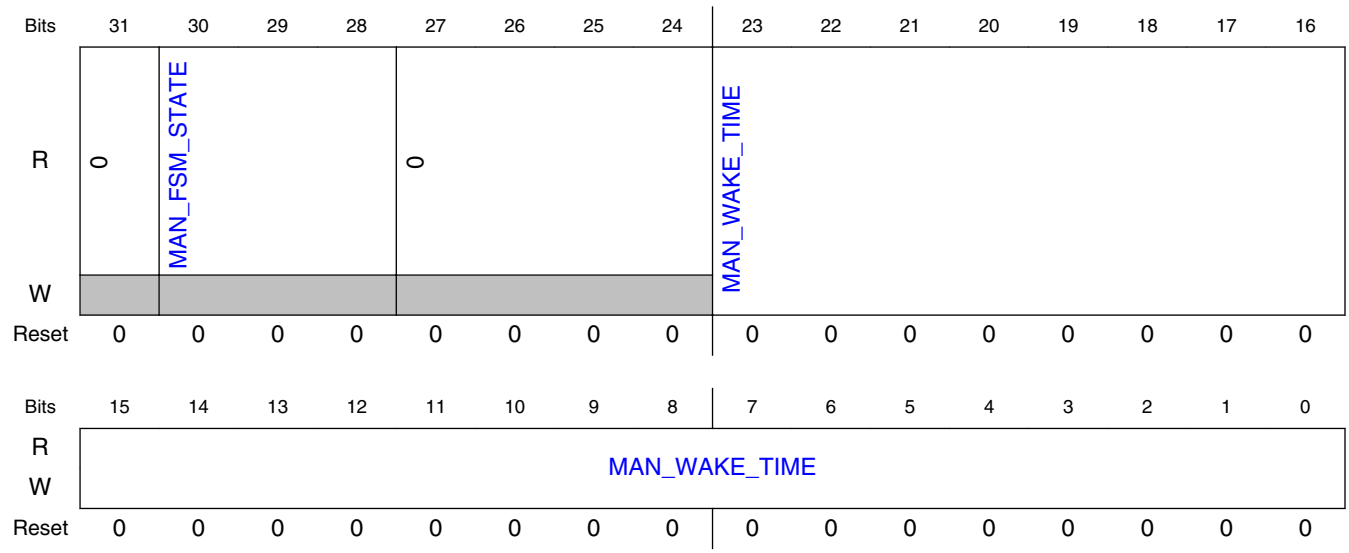
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>MAN_SLEEP_TIME | MAN Deep Sleep Module Sleep Time<br>Software determines the future time at which it would like to enter Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM start-time, and writing that value into this register. The value programmed into this register should be no fewer than 4 clocks greater (in the future) than the current time as read from DSM_TIMER. |

## 64.2.13.1.12 MAN Deep Sleep Wake Time (MAN_WAKE)

### 64.2.13.1.12.1 Offset

| Register | Offset |
|---|---|
| MAN_WAKE | 120h |

### 64.2.13.1.12.2 Diagram




### 64.2.13.1.12.3 Fields

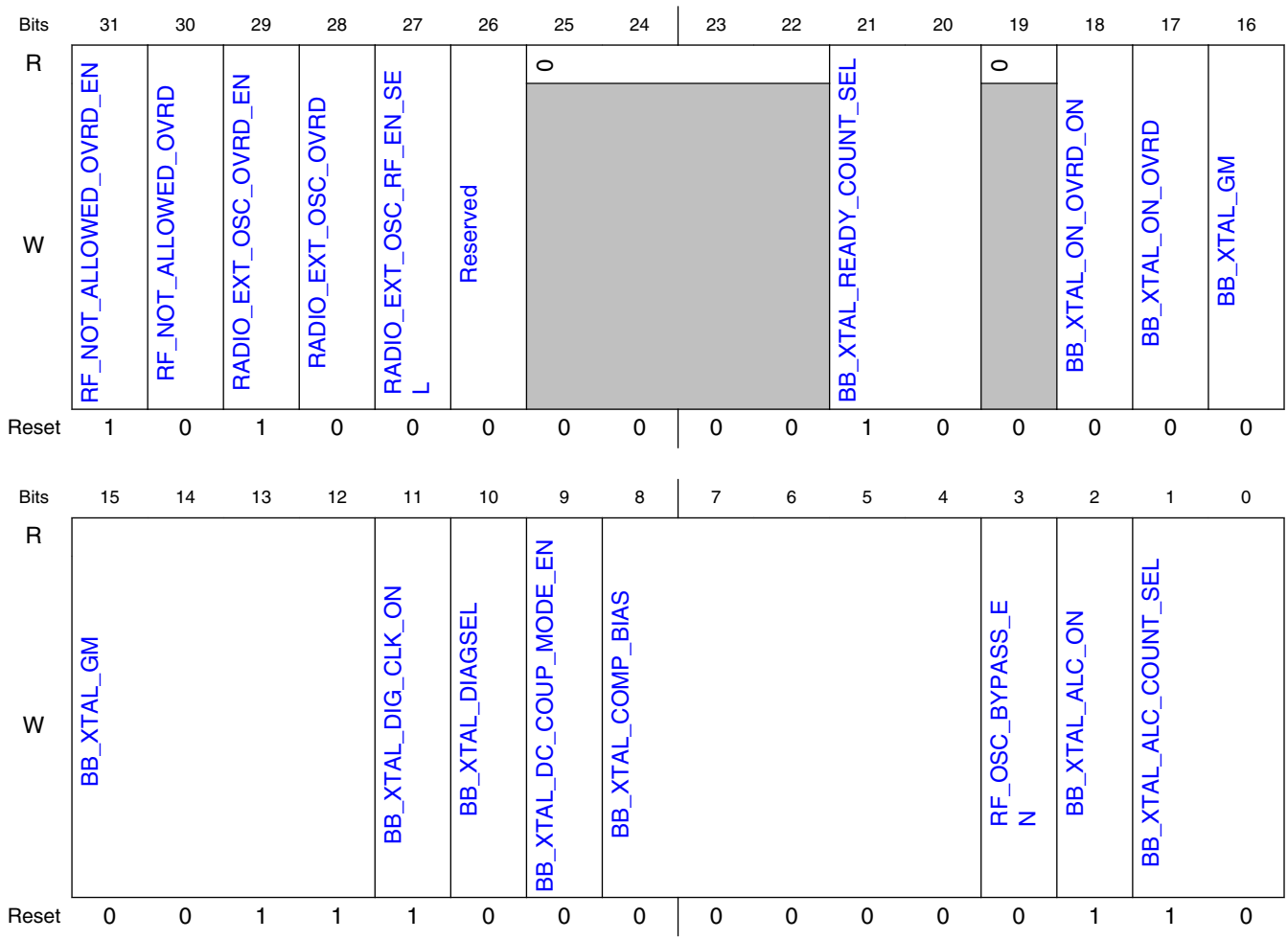
| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>MAN_FSM_STATE | MAN Deep Sleep State Machine State<br>This is the current State that the MAN Deep Sleep Finite State Machine is in. |
| 27-24<br>— | Reserved |
| 23-0<br>MAN_WAKE_TIME | MAN Deep Sleep Module Wake Time<br>Software determines the future time at which it would like to exit Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM exit time, and writing that value into this register. |

## 64.2.13.1.13 Radio Oscillator Control (RF_OSC_CTRL)

### 64.2.13.1.13.1 Offset

| Register | Offset |
|---|---|
| RF_OSC_CTRL | 124h |

## 64.2.13.1.13.2 Diagram




## 64.2.13.1.13.3 Fields

| Field | Function |
|---|---|
| 31<br>RF_NOT_ALLOWED_OVRD_EN | RF Not Allowed Override Enable<br>This bit enables the RF Not Allowed Override bit. |
| 30<br>RF_NOT_ALLOWED_OVRD | RF Not Allowed Override<br>If the RF_NOT_ALLOWED_OVRD_EN bit is set, then this bit controls the RF Not Allowed functionality instead of the SoC pin. |
| 29<br>RADIO_EXT_OSC_OVRD_EN | Radio External Request for RF OSC Override Enable<br>This bit enables the Radio External RF OSC Override bit. |
| 28<br>RADIO_EXT_OSC_OVRD | Radio External Request for RF OSC Override<br>If the RADIO_EXT_OSC_OVRD_EN bit is set, then this bit controls that External RF OSC Request functionality. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 27<br>RADIO_EXT_OSC_RF_EN_SEL | Radio External Request for RF OSC Select<br>The Radio has two pins which may be controlled by external pins in the SoC in such a way as to allow an external device, such as another radio, to enable this Radio's RF Oscillator. This bit selects which of the two pins has control of that external request. |
| 26<br>— | Reserved |
| 25-22<br>— | Reserved |
| 21-20<br>BB_XTAL_READY_COUNT_SEL | rmap_bb_xtal_ready_count_sel_hv[1:0]<br>Program counter for xtal ready signal<br>Sets up count value for XO startup time.<br>00b - 1024 counts (32 us @ 32 MHz)<br>01b - 2048 (64 us @ 32 MHz)<br>10b - 4096 (128 us @ 32 MHz)<br>11b - 8192 (256 us @ 32 MHz) |
| 19<br>— | Reserved |
| 18<br>BB_XTAL_ON_OVRD_ON | rmap_bb_xtal_on_ovrd_on_hv<br>Enable override XO enable bit<br>Enable selector:<br>0b - rfctrl_bb_xtal_on_hv is asserted<br>1b - rfctrl_bb_xtal_on_ovrd_hv is asserted |
| 17<br>BB_XTAL_ON_OVRD | rmap_bb_xtal_on_ovrd_hv<br>Override XO enable |
| 16-12<br>BB_XTAL_GM | rmap_bb_xtal_gm_hv[4:0]<br>Amplifier current bumps, bit [4] not used. Current values assume ALC is off<br>0 - Min setting: 8 current sources on (TT27:60uA)<br>...<br>15 - Max setting: 128 current sources on (TT27:960uA) |
| 11<br>BB_XTAL_DIG_CLK_ON | rmap_bb_xtal_dig_clk_on_hv<br>Enable digital clk output |
| 10<br>BB_XTAL_DIAGSEL | rmap_bb_xtal_diagsel_hv<br>Enable diagnostics for XO block |
| 9<br>BB_XTAL_DC_COUP_MODE_EN | rmap_bb_xtal_dc_coup_mode_en_hv<br>This bit enables the external dc coupled mode. This bit powers down the XO amplifier to enable DC coupled input. |
| 8-4<br>BB_XTAL_COMP_BIAS | rmap_bb_xtal_comp_bias_hv[4:0]<br>Not used. |

*Table continues on the next page...*

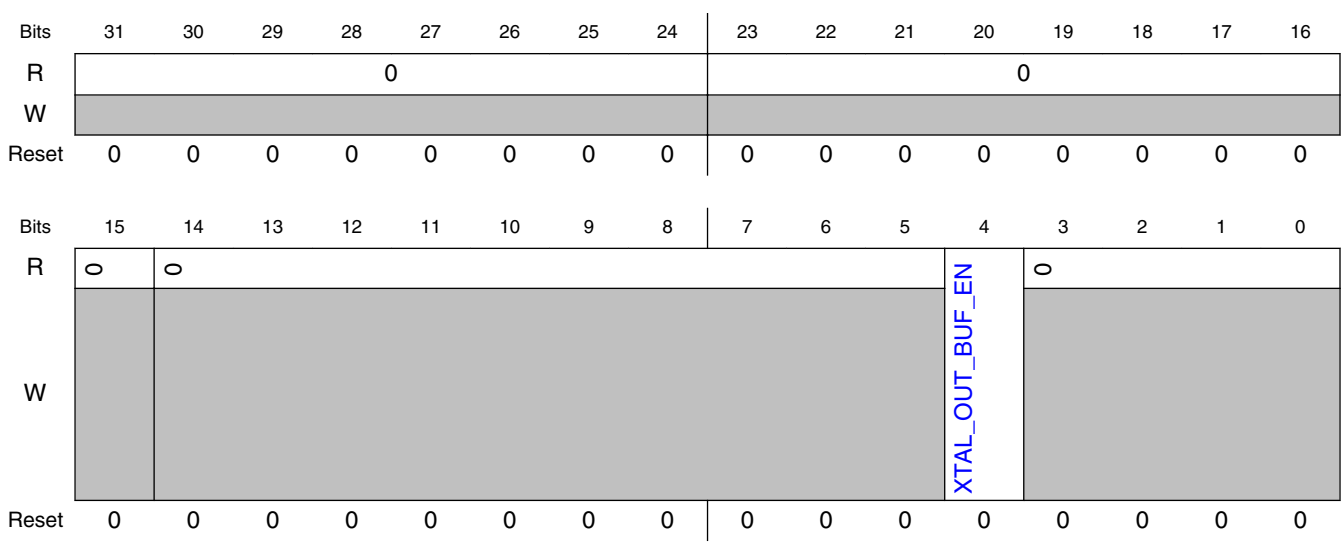
| Field | Function |
|---|---|
| 3<br>RF_OSC_BYPASS_EN | RF Ref Osc Bypass Enable<br>This bit engages the RF Ref Osc analog bypass circuit if the RF Ref Osc is enabled. When the RF Ref Osc is in bypass mode it passes the RF EXTAL clock as the RF Ref Osc clock. Note that the RF Ref Osc Ready signal functions normally in RF OSC Bypass mode, unless overridden with the RF_OSC_READY_OVRD_EN bit. |
| 2<br>BB_XTAL_ALC_ON | rmap_bb_xtal_alc_on_hv<br>Enable ALC |
| 1-0<br>BB_XTAL_ALC_COUNT_SEL | rmap_bb_xtal_alc_count_sel_hv[1:0]<br>Program counter for alc ready signal<br>Sets up count value for fastcharge to turn off:<br>00b - 2048 (64 us @ 32 MHz)<br>01b - 4096 (128 us @ 32 MHz)<br>10b - 8192 (256 us @ 32 MHz)<br>11b - 16384 (512 us @ 32 MHz) |

## 64.2.13.1.14 Radio Analog Test Registers (ANA_TEST)

### 64.2.13.1.14.1 Offset

| Register | Offset |
|---|---|
| ANA_TEST | 128h |

### 64.2.13.1.14.2 Diagram



### 64.2.13.1.14.3 Fields

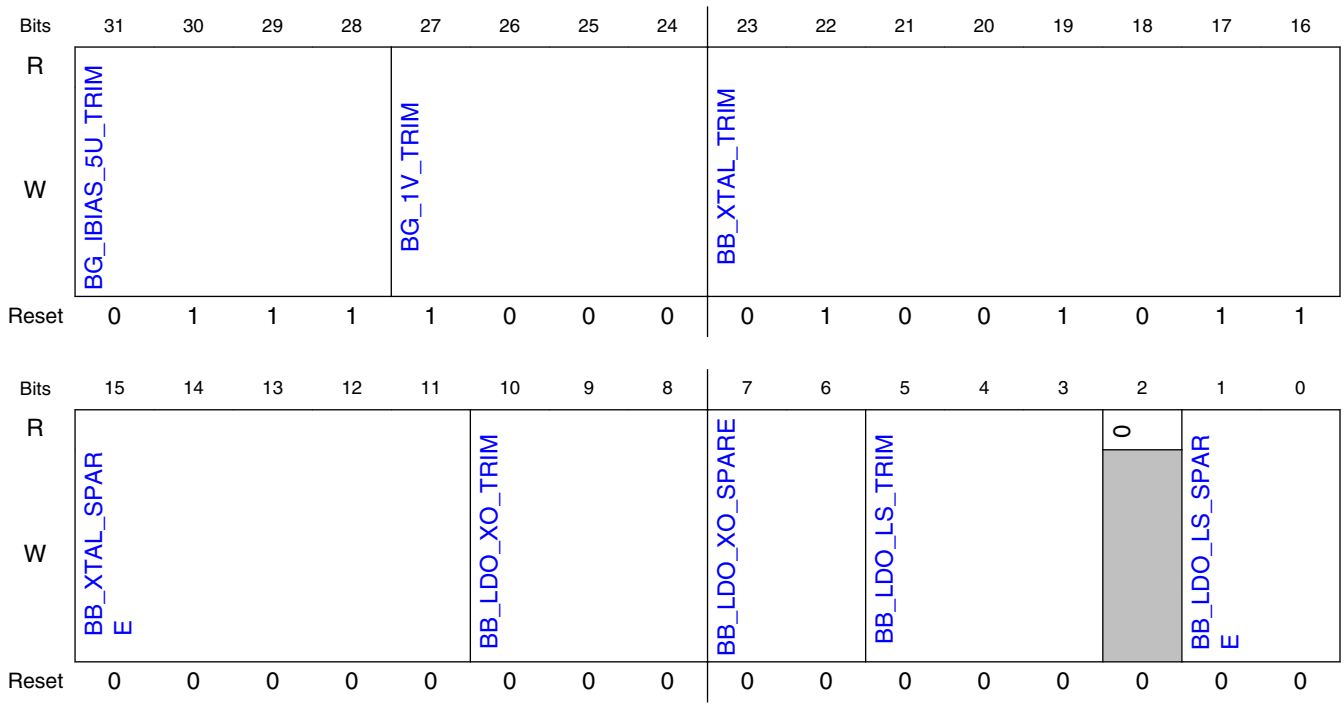
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-15<br>— | Reserved |
| 14-5<br>— | Reserved. |
| 4<br>XTAL_OUT_BUF_EN | XTAL Output Buffer Enable<br>The RF Analog Oscillator XTAL_OUT buffer for the XTAL_OUT SoC Pin is enabled when this bit is set. |
| 3-0<br>— | Reserved. |

## 64.2.13.1.15 Radio Analog Trim Registers (ANA_TRIM)

### 64.2.13.1.15.1 Offset

| Register | Offset |
|---|---|
| ANA_TRIM | 12Ch |

## 64.2.13.1.15.2 Diagram




## 64.2.13.1.15.3 Fields

| Field | Function |
|---|---|
| 31-28<br>BG_IBIAS_5U_TRIM | rmap_bg_ibias_5u_trim_hv[3:0]<br>5uA current trim bits. Default setting is 0111<br>Trim bits for VBG output current<br>0000b - 3.55 uA<br>0001b - 3.73 uA<br>0010b - 4.04 uA<br>0011b - 4.22 uA<br>0100b - 4.39 uA<br>0101b - 4.57 uA<br>0110b - 4.89 uA<br>0111b - 5.06 (Default)<br>1000b - 5.23 uA<br>1001b - 5.41 uA<br>1010b - 5.72 uA<br>1011b - 5.9 uA<br>1100b - 6.07 uA<br>1101b - 6.25 uA<br>1110b - 6.56 uA<br>1111b - 6.74 uA |
| 27-24<br>BG_1V_TRIM | rmap_bg_1v_trim_hv[3:0]<br>Trim bits for VBG output voltage<br>Trim bits for VBG output voltage |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0000b - 954.14 mV<br>0001b - 959.26 mV<br>0010b - 964.38 mV<br>0011b - 969.5 mV<br>0100b - 974.6 mV<br>0101b - 979.7 mV<br>0110b - 984.8 mV<br>0111b - 989.9 mV<br>1000b - 995 mV (Default)<br>1001b - 1 V<br>1010b - 1.005 V<br>1011b - 1.01 V<br>1100b - 1.015 V<br>1101b - 1.02 V<br>1110b - 1.025 V<br>1111b - 1.031 V |
| 23-16<br>BB_XTAL_TRIM | rmap_bb_xtal_trim_hv[7:0]<br>Bump XO load capacitor<br>Load capacitor bumps. bit [7] not used.<br>0 - Min C1: 5.7pF. Min C2: 7.1pF<br>...<br>127 - Max C1: 22.6pF. Max C2: 28.2pF |
| 15-11<br>BB_XTAL_SPARE | rmap_bb_xtal_spare_hv[4:0]<br>Bit 4: Override RF Analog XTAL Ready signal ; 0 - No override, BB_XTAL_READY_COUNT_SEL is used ; 1- Force RF Analog XTAL Ready = 1. Note that this RF Analog signal can be also be overridden using the RSIM RF_OSC_READY_OVRD bits.<br>Bit 3: XTAL_OUT output Polarity invert bit; 0 - No Polarity inversion; 1 - Polarity inverted<br>Bit 2: This bit selects the clock to be output on the XTAL_OUT SoC pin. The default is to select the XTAL as the clock source. If this bit is written to a 1 then the AuxPLL output will be selected instead.<br>Bits 1:0 are unused. |
| 10-8<br>BB_LDO_XO_TRIM | rmap_bb_ldo_xo_trim_hv[2:0]<br>Trim settings for LDO.<br>Trim settings for LDO.<br>000b - 1.20 V (Default)<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 7-6<br>BB_LDO_XO_SPARE | rmap_bb_ldo_xo_spare_hv[1:0]<br>Spare bits for LDO, not used. |
| 5-3<br>BB_LDO_LS_TRIM | rmap_bb_ldo_ls_trim_hv[2:0]<br>Trim settings for LDO.<br>Trim settings for LDO. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 000b - 1.20 V (Default)<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 2<br>— | Reserved |
| 1-0<br>BB_LDO_LS_SPARE | rmap_bb_ldo_ls_spare_hv[1:0]<br>Spare bits. Not used so far. |

# 64.3 Transceiver Digital

## 64.3.1 Transmitter Digital Module

### 64.3.1.1 About the Transmitter Digital

The Transmitter Digital processes the Transmission Data from the RF Protocol Link Layers and presents the processed data to the PLL Frequency Synthesizer as the Baseband Frequency Word.

## 64.3.1.1.1 Block diagram

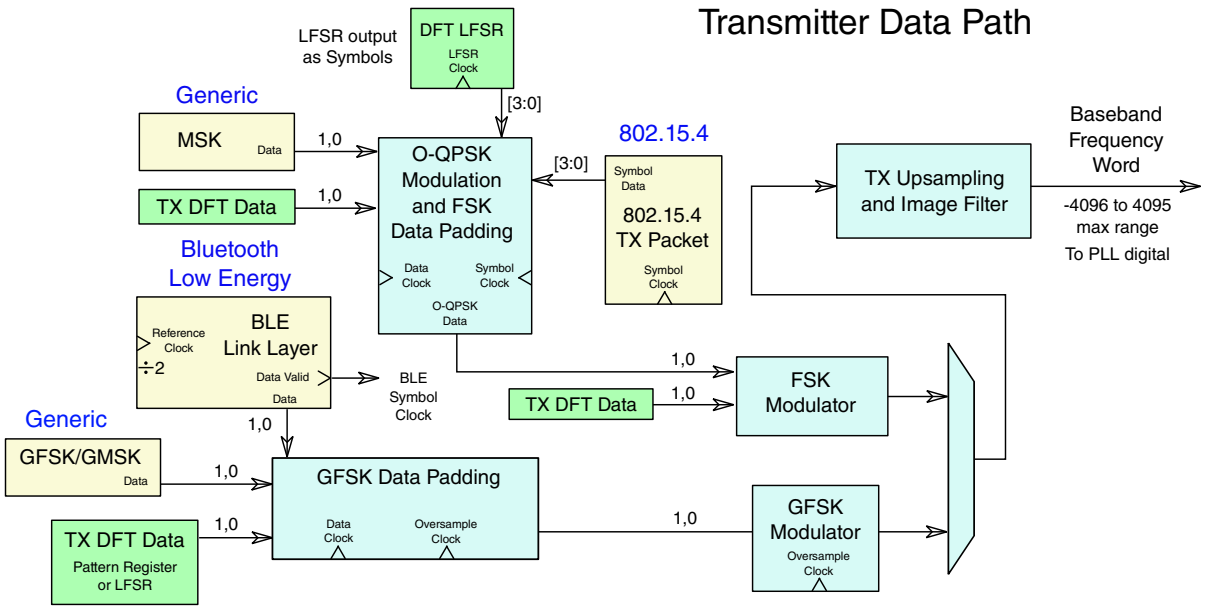



**Figure 64-12. Transmitter Digital**

## 64.3.1.1.2 Modulation Types

| Modulation Type[1] | Data/Chip Rate (kbps)[2] | BT Product[3] | Modulation Index[4] | Symbol Time (us) | Max Frequency Deviation (kHz) |
|---|---|---|---|---|---|
| **IEEE 802.15.4 OQPSK[5]** | 250/2000 | | 0.5 | 16 | 500 |
| **GMSK/ GFSK,BT=0.5, h=0.5** | 2000 | 0.5 | 0.5 | 0.5 | 500 |
| | 1000 | 0.5 | 0.5 | 1 | 250 |
| | 500 | 0.5 | 0.5 | 2 | 125 |
| | 250 | 0.5 | 0.5 | 4 | 62.5 |
| **GFSK, BT=0.5, h=0.32** | 2000 | 0.5 | 0.32 | 0.5 | 320 |
| | 1000 | 0.5 | 0.32 | 1 | 160 |
| | 500 | 0.5 | 0.32 | 2 | 80 |
| | 250 | 0.5 | 0.32 | 4 | 40 |
| **GFSK, BT=0.5, h=0.7** | 1000 | 0.5 | 0.7 | 1 | 350 |
| | 500 | 0.5 | 0.7 | 2 | 175 |
| | 250 | 0.5 | 0.7 | 4 | 87.5 |
| **GFSK, BT=0.5, h=1.0** | 1000 | 0.5 | 1 | 1 | 500 |
| | 500 | 0.5 | 1 | 2 | 250 |
| | 250 | 0.5 | 1 | 4 | 125 |
| **GMSK, BT=0.3** | 2000 | 0.3 | 0.5 | 0.5 | 300 |
| | 1000 | 0.3 | 0.5 | 1 | 150 |

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| | 500 | 0.3 | 0.5 | 2 | 75 |
| | 250 | 0.3 | 0.5 | 4 | 37.5 |
| **GMSK, BT=0.7** | 2000 | 0.7 | 0.5 | 0.5 | 700 |
| | 1000 | 0.7 | 0.5 | 1 | 350 |
| | 500 | 0.7 | 0.5 | 2 | 175 |
| | 250 | 0.7 | 0.5 | 4 | 87.5 |
| **Generic MSK** | 2000 | NA | 0.5 | 0.5 | 500 |
| | 1000 | NA | 0.5 | 1 | 250 |
| | 500 | NA | 0.5 | 2 | 125 |
| | 250 | NA | 0.5 | 4 | 62.5 |
| **Generic FSK** | 2000 | NA | 0.5 | 0.5 | 500 |
| | 1000 | NA | 0.5 | 1 | 250 |
| | 500 | NA | 0.5 | 2 | 125 |
| | 250 | NA | 0.5 | 4 | 62.5 |

1. Modulation Type is set using the PROTOCOL bits in the XCVR_CTRL register.
2. Data Rate is set using the BITRATE bits in the GENFSK register.
3. Bandwidth Time Product is changed by reprogramming the GFSK Gaussian Filter coefficients.
4. Modulation Index is set using the GFSK_MI register bits.
5. IEEE 802.15.4 uses O-QPSK modulation at a symbol rate of 62.5 ksps, which is spread to a chip rate of 2 Mbps.

#### 64.3.1.1.3 Data Rates

For other Radio protocols, the Transit Data Rate is set by the BITRATE bits in the GENFSK register.

The supported data rates are shown in the table below.

**Table 64-7. Transmitter Data Rates**

| GENFSK BITRATE[1:0] Register | TX Data Rate |
|---|---|
| 0 | 1000 kbit/sec |
| 1 | 500 kbit/sec |
| 2 | 250 kbit/sec |
| 3 | 2000 kbit/sec |

#### 64.3.1.1.4 Over-Sampling

The TX Digital uses an Oversample Clock to process the transmission data.

The GFSK, GMSK, and MSK Oversample Ratios are summarized below.

**Table 64-8. Oversample Ratios for GFSK, GMSK, and MSK Modulations**

| TX Data Rate (kb/s) | Ref Clk Freq (MHz) | OSR Ref Clk Divider | OSR Clk Rate (MHz) | Data Oversample Rate |
|---|---|---|---|---|
| 2000 | 32 | 2 | 16 | 8 |
| 1000 | 32 | 4 | 8 | 8 |
| 500 | 32 | 8 | 4 | 8 |
| 250 | 32 | 16 | 2 | 8 |
| 2000 | 26 | 1 | 26 | 13 |
| 1000 | 26 | 2 | 13 | 13 |
| 500 | 26 | 4 | 6.5 | 13 |
| 250 | 26 | 8 | 3.25 | 13 |

#### 64.3.1.1.5 Frequency Word Adjust

The Transmitter has a frequency adjustment register, FREQ_WORD_ADJ, which is used as a signed 9-bit number that is added to the TX Digital output before it is presented to the PLL as the baseband frequency word. This allows the baseband frequency word to be adjusted, or skewed, by a range of -512 to +511.

This frequency adjustment is applied to the final modulation word from any source (GFSK, FSK, DFT) and there is no protection from a math overflow. So the baseband frequency word range of -4096 to 4095 must be considered when adding this frequency word adjustment.

### 64.3.1.2 Data Pre-Processing

Before the Transmission Data is processed by the GFSK or FSK Modulators, it can be inverted, and it can also have additional preamble-like data pre-pended.

#### 64.3.1.2.1 Data Polarity

The Transmission Data that is received by the TX Digital module is mapped to a positive frequency deviation if the data is a 1, and is mapped to a negative frequency deviation if the data is a 0

This mapping can be reversed by setting the TX_CAPTURE_POL register bit.

#### 64.3.1.2.2 Data Padding

Transmission Data Padding is done to avoid abrupt steps in On-Air frequency modulation and thereby minimize spectral transients during the transition from a low Power Amplifier setting to a high Power Amplifier setting prior to protocol data packet transmission.

The abrupt step in modulation is avoided by pre-pending symbols of preamble-like modulation onto the front end of the actual preamble transmission. The Power Amplifier ramp-up is done while modulating these additional symbols, and then the modulating power ramp-up smoothly transitions into the packet transmission at the Target Power level.

Data padding is supported for all of the Radio Protocols packet preambles. For example, the BLE preamble is an alternating 1-0-1-0-1-0-1-0 or 0-1-0-1-0-1-0-1 pattern, depending on the first bit of the packet that follows. Data padding pre-pends the same 1 or 0 pattern as required to match the preamble case.

The data padding pattern is programmed into two 8-bit register fields to allow for two padding choices as seen in the BLE example shown above. If 16 bits of padding is selected, the 8-bit register fields are used twice for each case.

The DATA_PADDING_EN bits in the XCVR_TSM_CTRL register enable data padding as follows:

**Table 64-9. Data Padding Selection**

| DATA_PADDING_EN[1:0] | Result |
|---|---|
| 00 | No Data Padding |
| 01 | 8 bits or symbols of padding |
| 10 | 16 bits or symbols of padding |
| 11 | Reserved |

#### 64.3.1.2.3 Power Amplifier Ramping

The XCVR_TSM provides the TX Digital module with the ramp-up timing needed to adjust the data padding to match the time when the start of the protocol data packet transmission needs to go On-Air.

During ramp-down of the Power Amplifier the TX Digital holds the TX Modulation Data at the final value until power reaches the minimum level.

Please refer to the section Power Amplifier Ramping in the Transceiver Sequence Manager chapter for more details on the Power Ramping topic.

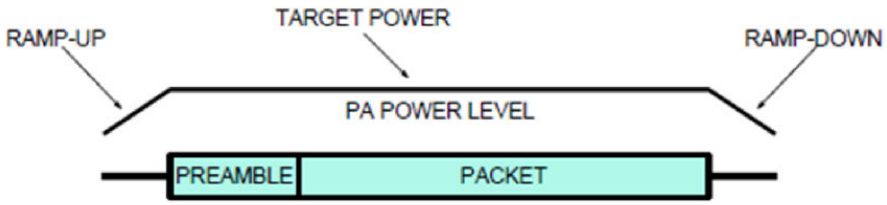



**Figure 64-13. Power Amplifier Ramping**

## 64.3.1.3 GFSK Modulator

### 64.3.1.3.1 Gaussian Filter

GFSK Modulation is implemented using a Gaussian filter and a lookup table to implement the various reference frequencies, data rates, and modulation index values.

The Gaussian filter is a symmetric 16-tap FIR filter with 11-bit quantized coefficients and supports reference frequencies of 32 and 26 MHz at oversampling ratios of 8 and 13 respectively. There are 2 sets of pre-programmed filter coefficients available as shown in the table below.

**Table 64-10. Gaussian Filter Coefficients**

| Filter OSR | 8 | 13 |
|---|---|---|
| Tap Number(s) | 32 MHz Ref | 26 MHz Ref |
| 1 | 1 | 23 |
| 2 | 4 | 41 |
| 3 | 13 | 68 |
| 4 | 41 | 103 |
| 5 | 99 | 144 |
| 6 | 192 | 186 |
| 7 | 300 | 220 |
| 8 | 374 | 239 |
| 9 | 374 | 239 |
| 10 | 300 | 220 |
| 11 | 192 | 186 |
| 12 | 99 | 144 |
| 13 | 41 | 103 |
| 14 | 13 | 68 |
| 15 | 4 | 41 |
| 16 | 1 | 23 |
| Coefficient Sum | 2048 | 2048 |

The pre-programmed filter coefficients can be manually overridden using the GFSK_FLD bit to disable the GFSK Filter Lookup Table. In this case the filter coefficients will be taken from the TX GFSK Filter Coefficients register.

#### 64.3.1.3.2 Multiplier

The GFSK Modulator has a multiplier in hardware that uses a lookup table to choose the correct value to multiply the output of the Gaussian filter by in order to get the correct frequency deviation range.

The lookup table value chosen is dependent upon the Reference Clock frequency, the Transmitter Data Rate, and the Modulation Index.

The GFSK Multiplier Lookup Table value can be overridden by setting the GFSK_MLD register bit and programming the GFSK_MULTIPLY_TABLE_MANUAL register to contain a valid integer, using bits [15:4], and a valid fraction, using bits [3:0].

#### 64.3.1.3.3 Modulation Scaling

The GFSK modulator supports nominal modulation indices of 0.32, 0.5, 0.7 and 1.0. Transmit modulation scaling adds the capability to scale the GFSK modulated output up or down by a factor 1/32, 1/16 or 1/8. The scaling coefficients have been chosen so that the scaling can be realized without a multiplication.

The TX modulation scaling is equivalent to having some programmability on the nominal modulation index, which can be useful to alter the modulation bandwidth and/or effective frequency deviation of the TX modulation signal.

Transmit modulation scaling is applied to the Gaussian Filter output.

**Table 64-11. Transmit Modulation Scaling Choices**

| GFSK_MOD_INDEX_SCALING[2:0] | Modulation Scaling | Scaling of nominal modulation Index (%) |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 + 1/32 | +3.125 |
| 2 | 1 + 1/16 | +6.25 |
| 3 | 1 + 1/8 | +12.5 |
| 4 | 1 - 1/32 | -3.125 |
| 5 | 1 - 1/16 | -6.25 |
| 6 | 1 - 1/8 | -12.5 |

#### 64.3.1.3.4 On-Air Frequency Deviation

By default, the GFSK Modulator presents the modulation word to the PLL starting with the Negative Frequency Deviation, this results in the PLL going On-Air at the Carrier Frequency minus the frequency deviation.

The GFSK Modulator can instead be configured to begin modulating at Zero Frequency Deviation, allowing the PLL to go On-Air at the Carrier Frequency. This configuration is used if the ZERO_FDEV bit in the TX Digital Control register is set.

### 64.3.1.4 FSK Modulator

#### 64.3.1.4.1 FSK Modulation Word

For 802.15.4 FSK Modulation, the FSK modulation word presented to the PLL is a direct mapping of a data 1 to the signed value of the FSK_MODULATION_SCALE_1 register, and of a data 0 to the signed value of the FSK_MODULATION_SCALE_0 register.

For Generic MSK and Generic FSK, the FSK modulation word presented to the PLL is also scaled based on the BITRATE bits in the GENFSK register as shown in the table below.

This scaling can be disabled by setting the FSK_BITRATE_SCALE_DISABLE bit in the FSK_SCALE register.

**Table 64-12. FSK Modulation Scaling**

| GENFSK BITRATE[1:0] Register | FSK Modulation Data Rate | FSK Modulation Scaling |
|---|---|---|
| 0 | 1000 kbit/sec | Divide by 2 |
| 1 | 500 kbit/sec | Divide by 4 |
| 2 | 250 kbit/sec | Divide by 8 |
| 3 | 2000 kbit/sec | No Scaling |

Note that a frequency inversion is allowed by inverting the values in the FSK_MODULATION_SCALE registers.

The default FSK Modulation mapping translates the chips/bits to a frequency modulation of +Frequency Deviation/PLL Minimum Modulation Step for a data 1, and -Frequency Deviation/PLL Minimum Modulation Step for a data 0.

For a 32 MHz reference clock with a PLL Minimum Modulation Step Size of 244.14 Hz, the default FSK Modulation mapping translates to a +/- 500 kHz peak Frequency Deviation, which is a PLL Baseband Frequency Word of +2047/-2047.

#### 64.3.1.4.1.1 Half-Sine Pulse Shaping

The half-sine pulse shaping results from the inherent VCO reaction to the digital FSK modulation and the resulting modulation is as specified in IEEE 802.15.4 section 10.2.5, Figure 71, shown below.

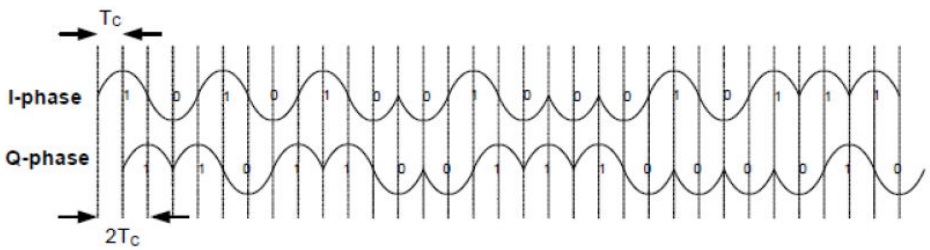



Figure 71—Sample baseband chip sequences with pulse shaping

**Figure 64-14. Half-Sine Pulse Shaping**

### 64.3.1.4.2 MSK Modes

Minimum-shift keying (MSK) is a type of continuous-phase frequency-shift keying with a frequency separation of one-half the bit rate. MSK modulation construction is quite similar to O-QPSK; MSK is encoded with bits alternating between quadrature components, with the Q component delayed by half the symbol period.

However, instead of square pulses as used by ordinary O-QPSK, MSK encodes each bit as a half sinusoid, which is similar to IEEE 802.15.4 O-QPSK. This results in a constant-modulus signal (constant envelope signal), which reduces problems caused by non-linear distortion.

In MSK the difference between the higher and lower frequency is identical to half the bit rate. Consequently, the waveforms used to represent a 0 and a 1 bit differ by exactly half a carrier period.

Thus, the maximum frequency deviation is $\delta = 0.25$ fm where fm is the maximum modulating frequency. As a result, the modulation index M is 0.5. This is the smallest FSK modulation index that can be chosen such that the waveforms for 0 and 1 are orthogonal.

As an example, 1 Mbps MSK would result in a frequency deviation of +/-250 kHz.

MSK data comes to the O-QPSK Modulator at the chip/data rate (no symbol to chip mapping is done), and the O-QPSK conversion is done on the data in the same manner as it is in IEEE 802.15.4-2011 section 10.2.5, as shown above.

### 64.3.1.5 Data Post-Processing

After the Transmission Data is processed by the GFSK or FSK Modulators, it can be post-processed to remove TX modulation images.

#### 64.3.1.5.1 Sample/Hold Images

Power efficient constant envelope transmit modulation is done at a multiple of the symbol clock (Over-Sample Rate), which is chosen as a fraction of the RF Osc reference clock frequency. The transmit modulation signal is then converted to the reference clock rate, at which it is presented to the PLL as the modulation word. This sample/hold (or zero-order hold, ZOH) operation results in TX modulation images that repeat at the fractional reference clock rate at used.

These modulation images are automatically filtered by the sinc ZOH transfer function and the integration (i.e., 1/s) operation of the VCO. These inherent filtering operations result in these modulation images not violating any of the transmit modulation frequency mask. However, to provide additional spectral margin to support using an external power amplifier to boost the transmit power, an image suppression set of FIR filters has been included in the transmit path to suppress the transmit sample/hold images by at least an additional 20 dB.

The transmit image filters are enabled by default in GFSK and GMSK modes and not enabled in IEEE 802.15.4 and MSK modulation modes, but their usage can be overridden using the TX_IMAGE_FILTER_OVRD_EN bit.

#### 64.3.1.5.2 Image Filters

The Transmitter uses up to three image filters, by default they are enabled based upon the modulation type and data rate as shown below:

**Table 64-13. Default Transmit Image Filter Usage**

| Modulation Type; Data Rate | TX Image Filter 2 | TX Image Filter 1 | TX Image Filter 0 |
|---|---|---|---|
| OQPSK/FSK/MSK Modes; All data rates | 0 | 0 | 0 |
| GFSK/GMSK; 1000 kbps | 0 | 0 | 1 |
| GFSK/GMSK; 500 kbps | 0 | 1 | 1 |
| GFSK/GMSK; 250 kbps | 1 | 1 | 1 |

### 64.3.1.5.3 Image Filters Frequency Response

The frequency response of the TX image suppression filters and the reduction in the level of TX images for the 1 Mbps GMSK (BLE) with the FIR filter are demonstrated in the plots below.

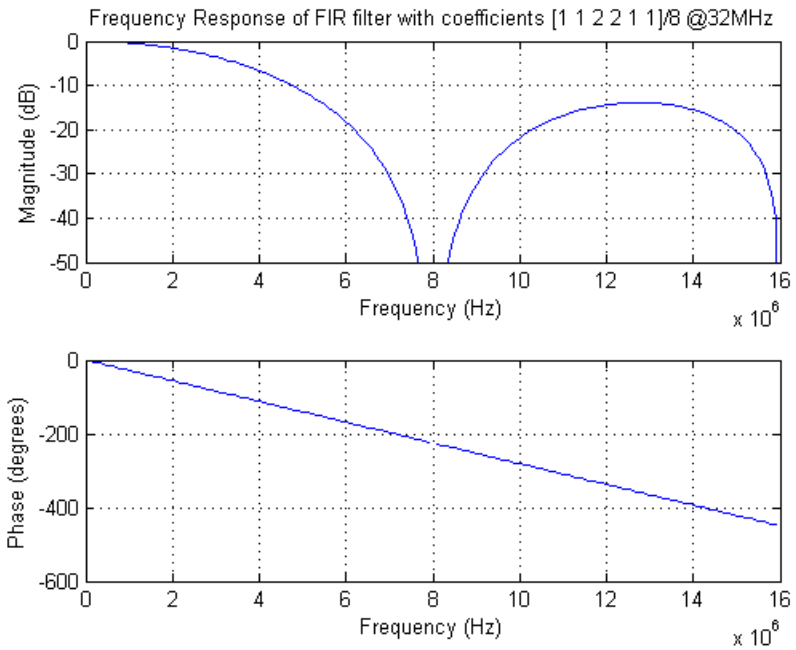



**Figure 64-15. Frequency response of the 6-Tap FIR Transmit Image Suppression filter**

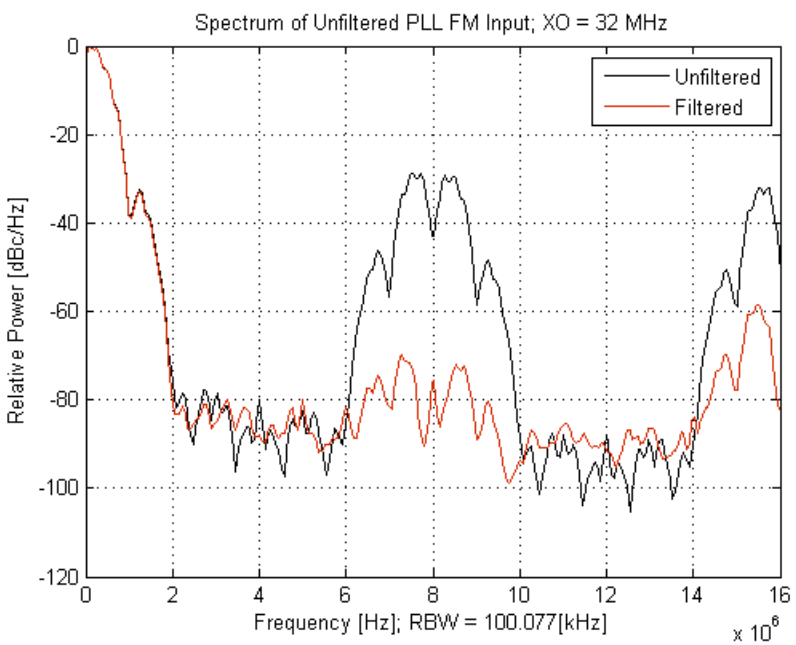



**Figure 64-16. Comparison of Transmit Modulation Images at the PLL Input for BLE**

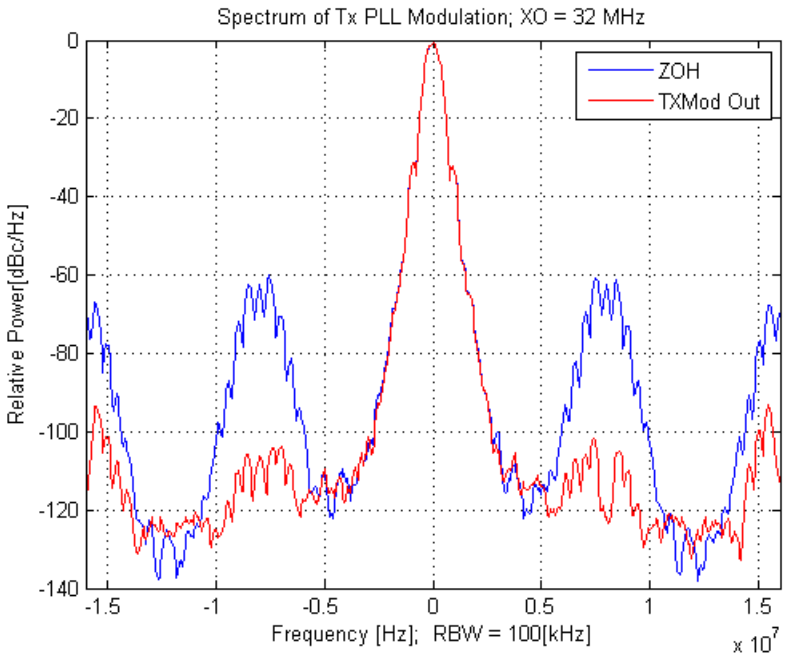



**Figure 64-17. Comparison of Transmit Modulation Images at the PLL Output for BLE (1 Mbps GMSK)**

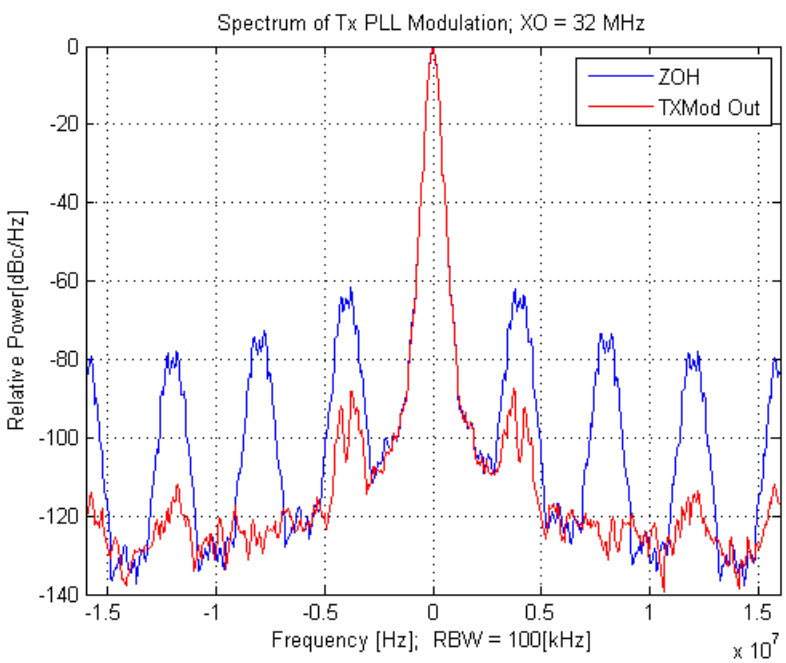



**Figure 64-18. Comparison of Transmit Modulation Images at the PLL Output for 500 kbps GMSK**

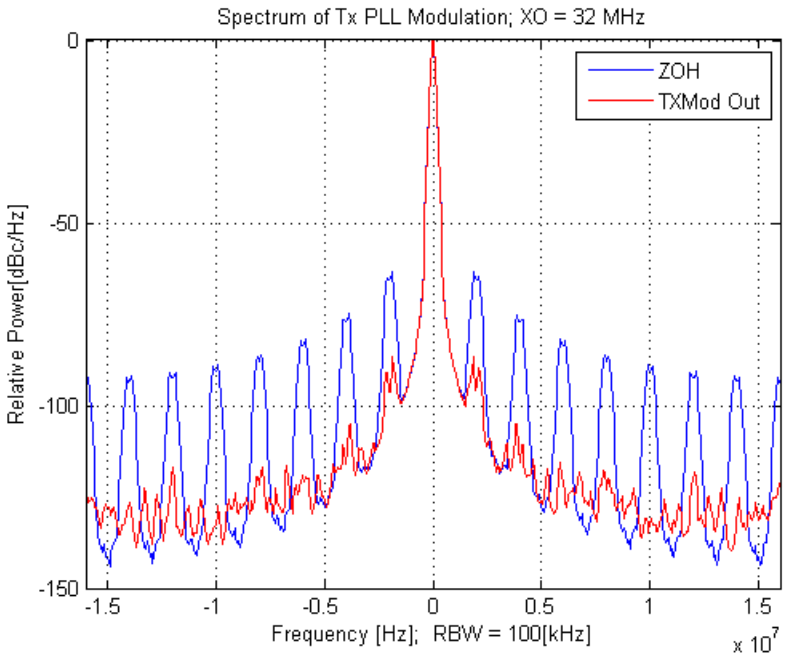



**Figure 64-19. Comparison of Transmit Modulation Images at the PLL Output for 250 kbps GMSK**

## 64.3.1.6 Memory Map and Register Definition

The Transmitter Digital memory map and detailed descriptions of all its registers are as follows.

### 64.3.1.6.1 XCVR_TX_DIG register descriptions

#### 64.3.1.6.1.1 XCVR_TX_DIG_ADDR Memory map

XCVR_TX_DIG base address: 4103_0200h

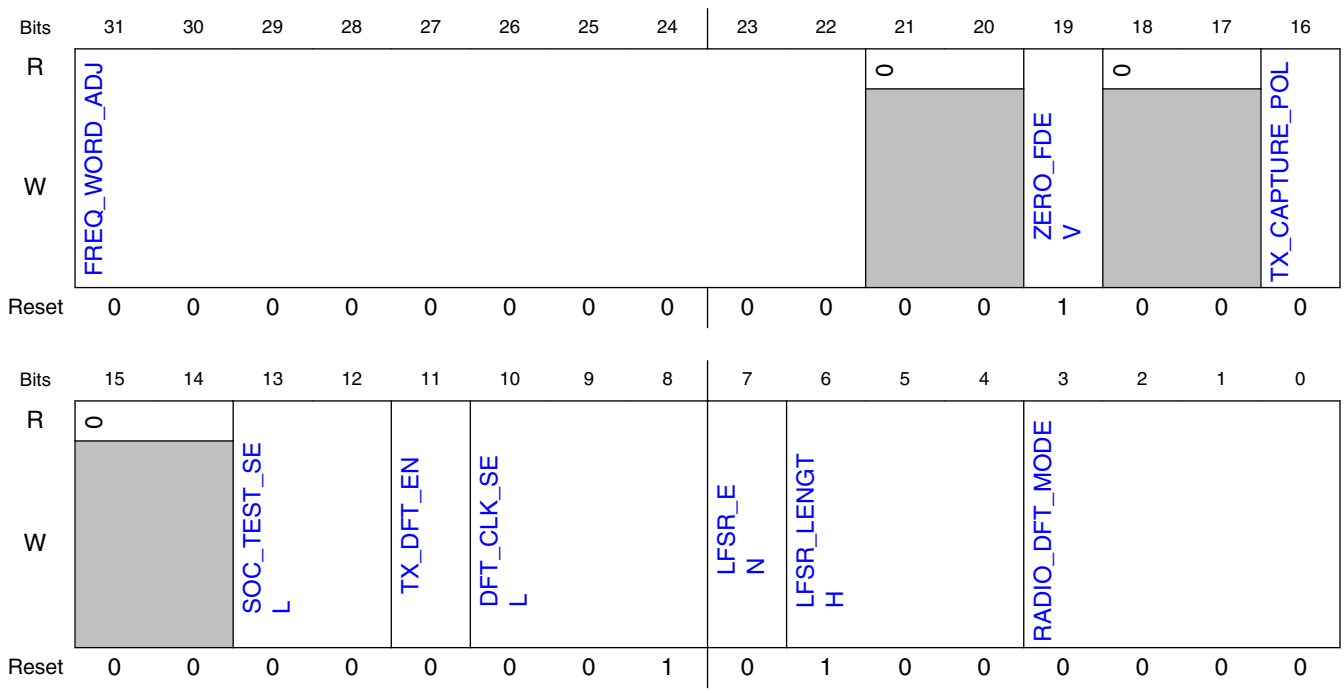
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TX Digital Control (CTRL) | 32 | RW | 0008_0140h |
| 4h | TX Data Padding (DATA_PADDING) | 32 | RW | 7FFF_55AAh |
| 8h | TX GFSK Modulator Control (GFSK_CTRL) | 32 | RW | 0001_4000h |
| Ch | TX GFSK Filter Coefficients 2 (GFSK_COEFF2) | 32 | RW | C063_0401h |
| 10h | TX GFSK Filter Coefficients 1 (GFSK_COEFF1) | 32 | RW | BB29_960Dh |
| 14h | TX FSK Modulation Levels (FSK_SCALE) | 32 | RW | 0800_1800h |
| 18h | TX DFT Modulation Pattern (DFT_PATTERN) | 32 | RW | 0000_0000h |

## 64.3.1.6.1.2 TX Digital Control (CTRL)

### *64.3.1.6.1.2.1 Offset*

| Register | Offset |
|---|---|
| CTRL | 0h |

### *64.3.1.6.1.2.2 Diagram*




### *64.3.1.6.1.2.3 Fields*

| Field | Function |
|---|---|
| 31-22<br>FREQ_WORD_ADJ | Frequency Word Adjustment<br>This register is a signed 9 bit number that is added to the TX Digital output before it is presented to the PLL as the baseband frequency word. This allows the baseband frequency word to be adjusted, or skewed, by a range of -512 to +511.<br>This frequency adjustment is applied to the final modulation word from any source (GFSK, FSK, DFT) and there is no protection from a math overflow. So the baseband frequency word range of -4096 to 4095 must be considered when adding this frequency word adjustment. |
| 21-20<br>— | Reserved |
| 19 | On-Air at Zero FDev |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ZERO_FDEV | If this bit is set, then in Gaussian Modulation modes the modulation will start at the Carrier Frequency instead of at the negative modulation value. |
| 18-17<br>— | Reserved |
| 16<br>TX_CAPTURE_POL | Polarity of the Input Data for the Transmitter<br>If this bit is set, the TX data presented to the Transmitter will be inverted before it is processed. |
| 15-14<br>— | Reserved |
| 13-12<br>SOC_TEST_SEL | Radio Clock Selector for SoC RF Clock Tests<br>This register selects the Radio clock source for the SoC Clock Tests Frequency Measurement.<br>00b - No Clock Selected<br>01b - PLL Sigma Delta Clock, divided by 2<br>10b - Auxiliary PLL Clock, divided by 2<br>11b - RF Ref Osc clock, divided by 2 |
| 11<br>TX_DFT_EN | DFT Modulation Enable<br>If the Radio is in a DFT Pattern Register mode, then this bit is used to turn on and off the modulation that is shifted out from the pattern register. |
| 10-8<br>DFT_CLK_SEL | DFT Clock Selection<br>This register selects the frequency of the DFT clock that is used to shift out the DFT Modulation Pattern in DFT Pattern Register modes, and the same frequency is also used to clock the LFSR and generate the pseudo-random modulation in DFT LFSR modes.<br>000b - 62.5 kHz<br>001b - 125 kHz<br>010b - 250 kHz<br>011b - 500 kHz<br>100b - 1 MHz<br>101b - 2 MHz<br>110b - 4 MHz<br>111b - RF OSC Clock |
| 7<br>LFSR_EN | LFSR Enable<br>If the Radio is in a DFT LFSR mode, then this bit is used to turn on and off the LFSR that is used to generate the modulation.<br>Note that the LFSR is clocked at the DFT Clock frequency. |
| 6-4<br>LFSR_LENGTH | LFSR Length<br>This register selects the length of the DFT LFSR and the associated LFSR Tap Mask.<br>The Mask is in the form of [MSB...LSB]<br>000b - LFSR 9, tap mask 100010000<br>001b - LFSR 10, tap mask 1001000000<br>010b - LFSR 11, tap mask 11101000000<br>011b - LFSR 13, tap mask 1101100000000<br>100b - LFSR 15, tap mask 111010000000000<br>101b - LFSR 17, tap mask 11110000000000000<br>110b - Reserved<br>111b - Reserved |
| 3-0 | Radio DFT Modes |

| Field | Function |
|---|---|
| RADIO_DFT_MODE | This register selects the Radio DFT mode as described below.<br>In addition to setting the Radio DFT mode, the DFT LFSR needs to be configured, and the Radio Protocol needs to be chosen.<br>For LFSR modes the LFSR_EN needs to be set to turn on the LFSR.<br>0000b - Normal Radio Operation, DFT not engaged.<br>0001b - Carrier Frequency Only<br>0010b - Pattern Register GFSK<br>0011b - LFSR GFSK<br>0100b - Pattern Register FSK<br>0101b - LFSR FSK<br>0110b - Pattern Register O-QPSK<br>0111b - LFSR O-QPSK<br>1000b - LFSR 802.15.4 Symbols<br>1001b - PLL Modulation from RAM<br>1010b - PLL Coarse Tune BIST<br>1011b - PLL Frequency Synthesizer BIST<br>1100b - High Port DAC BIST<br>1101b - VCO Frequency Meter<br>1110b - Reserved<br>1111b - Reserved |

## 64.3.1.6.1.3 TX Data Padding (DATA_PADDING)

### *64.3.1.6.1.3.1 Offset*

| Register | Offset |
|---|---|
| DATA_PADDING | 4h |

### *64.3.1.6.1.3.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LRM | DFT_LFSR_OUT | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_PADDING_PAT_1 | | | | | | | | DATA_PADDING_PAT_0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

#### 64.3.1.6.1.3.3 Fields

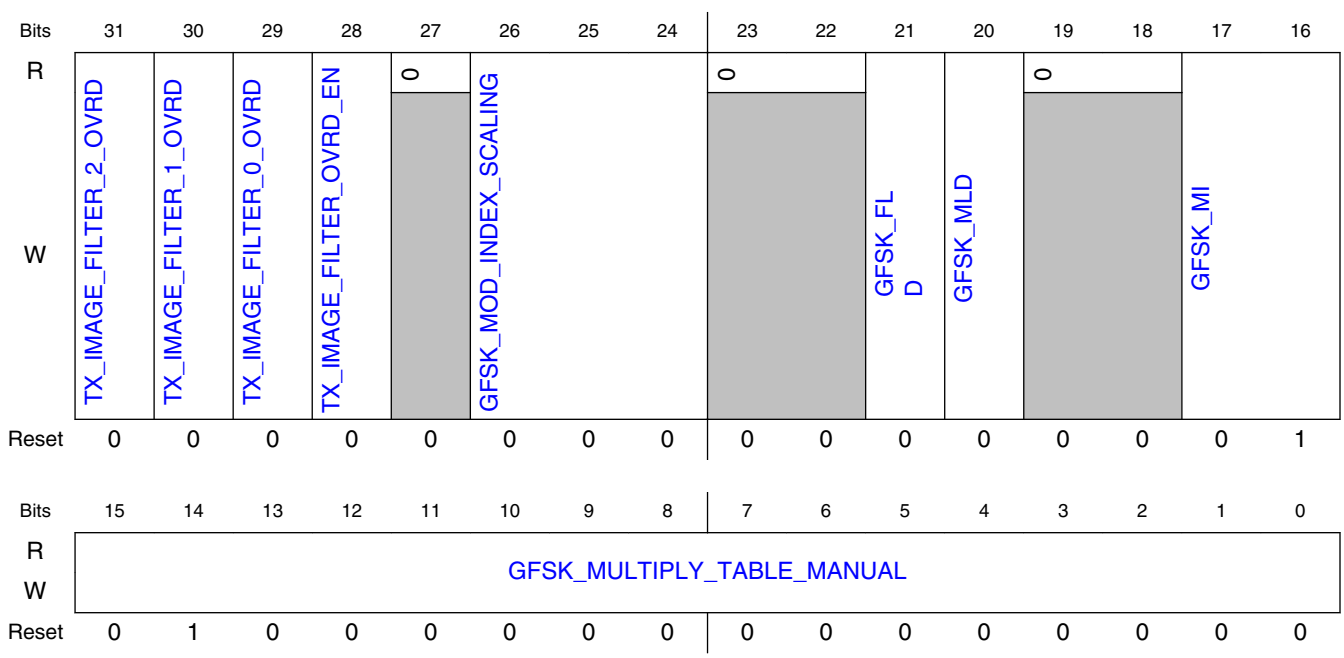
| Field | Function |
|---|---|
| 31<br>LRM | LFSR Reset Mask<br>When this bit is set the DFT LFSR will not be reset when LFSR_EN is cleared and will instead continue to repeat its sequence as defined by the DFT_LFSR_LEN bits when LFSR_EN is next set. When this bit is cleared the DFT LFSR will reset every time LFSR_EN is cleared. |
| 30-16<br>DFT_LFSR_OUT | LFSR Output<br>This register can be read to observe the current value of the DFT LFSR, only bits [14:0] are available. |
| 15-8<br>DATA_PADDING_PAT_1 | Data Padding Pattern 1<br>These bits are used for Data Padding when the first bit of the Preamble is 1; the LSB is the first bit shifted out as padding. |
| 7-0<br>DATA_PADDING_PAT_0 | Data Padding Pattern 0<br>These bits are used for Data Padding when the first bit of the Preamble is 0; the LSB is the first bit shifted out as padding. |

## 64.3.1.6.1.4 TX GFSK Modulator Control (GFSK_CTRL)

#### 64.3.1.6.1.4.1 Offset

| Register | Offset |
|---|---|
| GFSK_CTRL | 8h |

#### 64.3.1.6.1.4.2 Diagram



### *64.3.1.6.1.4.3 Fields*

| Field | Function |
|---|---|
| 31<br>TX_IMAGE_FILTER_2_OVRD | TX Image Filter 2 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 2 |
| 30<br>TX_IMAGE_FILTER_1_OVRD | TX Image Filter 1 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 1 |
| 29<br>TX_IMAGE_FILTER_0_OVRD | TX Image Filter 0 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 0 |
| 28<br>TX_IMAGE_FILTER_OVRD_EN | TX Image Filter Override Enable<br>This bit enables the TX Image Filter Override Control bits. |
| 27<br>— | Reserved |
| 26-24<br>GFSK_MOD_INDEX_SCALING | GFSK Modulation Index Scaling Factor<br>This register selects the amount to scale the Transmitter Modulation Word in GFSK Protocols.<br>Transmit modulation scaling adds the capability to scale the GFSK modulated output up or down by a factor of 1/32, 1/16 or 1/8. This is equivalent to having some additional programmability of the modulation index.<br>000b - 1<br>001b - 1 + 1/32<br>010b - 1 + 1/16<br>011b - 1 + 1/8<br>100b - 1 - 1/32<br>101b - 1 - 1/16<br>110b - 1 - 1/8<br>111b - Reserved |
| 23-22<br>— | Reserved |
| 21<br>GFSK_FLD | Disable GFSK Filter Lookup Table<br>If this bit is set, the internal GFSK filter coefficients that are normally derived from a lookup table based on the reference clock frequency, are disabled, and the coefficients are instead derived from the GFSK_FILTER_COEFF_MANUAL1 and GFSK_FILTER_COEFF_MANUAL2 registers. |
| 20<br>GFSK_MLD | Disable GFSK Multiply Lookup Table<br>If this bit is set, the GFSK Multiply Lookup table is disabled and GFSK_MULTIPLY_TABLE_MANUAL is used instead. |
| 19-18<br>— | Reserved |
| 17-16<br>GFSK_MI | GFSK Modulation Index<br>This register selects the GFSK Modulation Index which, together with the GFSK Symbol Rate, determines the Peak Modulation frequency. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The formula used for the Peak Modulation is { Symbol Rate / (2 x 1/Modulation Index) }<br>00b - 0.32<br>01b - 0.50<br>10b - 0.70<br>11b - 1.00 |
| 15-0<br>GFSK_MULTIPLY_TABLE_MANUAL | Manual GFSK Multiply Lookup Table Value<br>The GFSK Modulator Multiplier uses a lookup table to select the multiplicand representing the { Frequency Deviation divided by the Low Port Sigma Delta LSB resolution in Hz } for the Modulation requested based on the Modulation Index, the Symbol Rate, and the Reference Clock Frequency.<br>The lookup table value is overridden by this register if GFSK_MLD is set, and these bits should then contain a number that represents { FDev/SD_LSB integer[11:0] + FDev/SD_LSB fraction[3:0] } |

## 64.3.1.6.1.5 TX GFSK Filter Coefficients 2 (GFSK_COEFF2)

### *64.3.1.6.1.5.1 Offset*

| Register | Offset |
|---|---|
| GFSK_COEFF2 | Ch |

### *64.3.1.6.1.5.2 Function*

The two registers TX_GFSK_COEFF1 and TX_GFSK_COEFF2 form a 64-bit little endian register that is memory mapped internally as shown to override the GFSK Filter Coefficients.

In 64-bit they combine as {TX_GFSK_COEFF2[31:0],TX_GFSK_COEFF1[31:0]}

The resulting 64-bit value is the GFSK Manual Filter Coefficient [63:0]

### *64.3.1.6.1.5.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL2 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### *64.3.1.6.1.5.4 Fields*

| Field | Function |
|---|---|
| 31-0<br>GFSK_FILTER_COEFF_MANUAL2 | GFSK Manual Filter Coefficients[63:32]<br>If GFSK_FLD is set, the GFSK Filter Coefficients are overridden using the bits as described below, and since the filter is symmetrical each coefficient is used twice. |

| Bits | Filter Coefficients |
|---|---|
| [36:32] | Filter coeff 0 and 15 |
| [45:40] | Filter coeff 1 and 14 |
| [55:48] | Filter coeff 4 and 11 |
| [63:56] | Filter coeff 5 and 10 |

### 64.3.1.6.1.6 TX GFSK Filter Coefficients 1 (GFSK_COEFF1)

#### *64.3.1.6.1.6.1 Offset*

| Register | Offset |
|---|---|
| GFSK_COEFF1 | 10h |

#### *64.3.1.6.1.6.2 Function*

The two registers TX_GFSK_COEFF1 and TX_GFSK_COEFF2 form a 64-bit little endian register that is memory mapped internally as shown to override the GFSK Filter Coefficients.

In 64-bit they combine as {TX_GFSK_COEFF2[31:0],TX_GFSK_COEFF1[31:0]}

The resulting 64-bit value is the GFSK Manual Filter Coefficient [63:0]

#### 64.3.1.6.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL1 | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL1 | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

#### 64.3.1.6.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>GFSK_FILTER_COEFF_MANUAL1 | GFSK Manual Filter Coefficient [31:0]<br>If GFSK_FLD is set, the GFSK Filter Coefficients are overridden using the bits as described below, and since the filter is symmetrical each coefficient is used twice. |

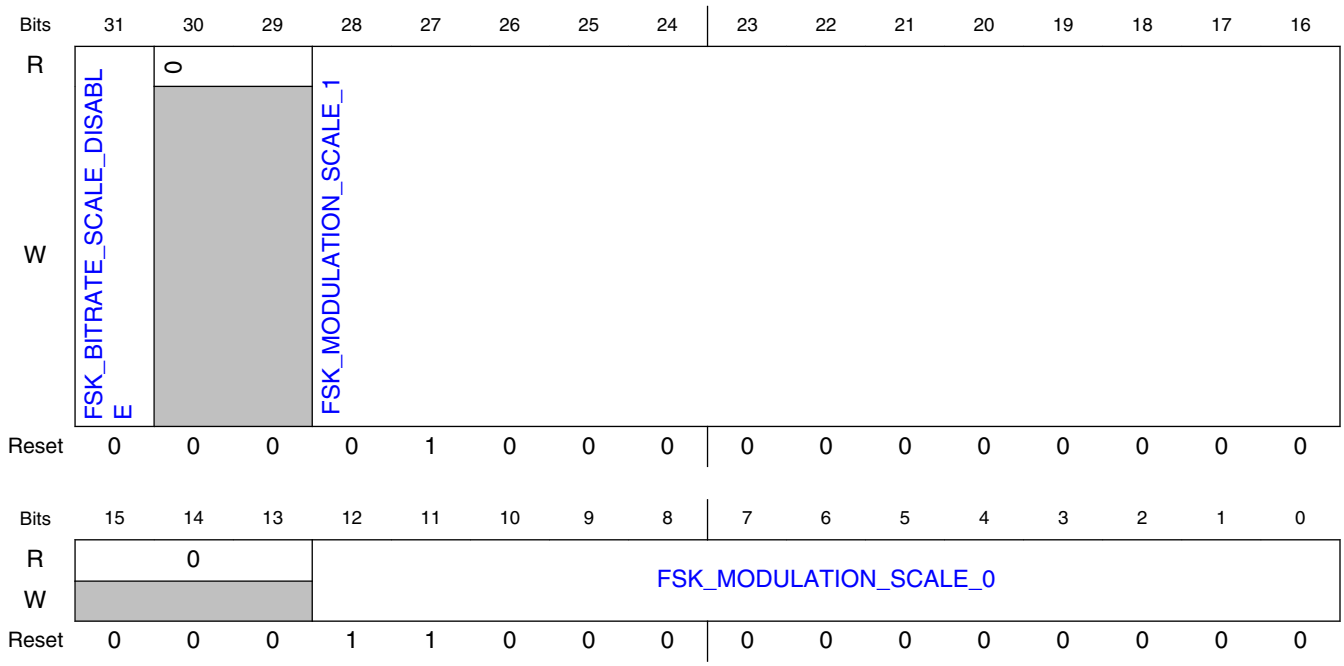
| Bits | 000 Filter Coefficients |
|---|---|
| [6:0] | Filter coeff 2 and 13 |
| [15:7] | Filter coeff 6 and 9 |
| [22:16] | Filter coeff 3 and 12 |
| [31:23] | Filter coeff 7 and 8 |

## 64.3.1.6.1.7 TX FSK Modulation Levels (FSK_SCALE)

#### 64.3.1.6.1.7.1 Offset

| Register | Offset |
|---|---|
| FSK_SCALE | 14h |

### 64.3.1.6.1.7.2 Diagram




### 64.3.1.6.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>FSK_BITRATE_SCALE_DISABLE | FSK Bitrate Scaling Disable<br>If this bit is set, the FSK Modulation values will not be scaled according to the bitrate setting. |
| 30-29<br>— | Reserved |
| 28-16<br>FSK_MODULATION_SCALE_1 | FSK Modulation Scale for a data 1<br>This register is used to provide the modulation level for a data 1 in FSK Protocols. The signed 12 bit value that is represented by this register is presented to the PLL as the Baseband Frequency Word. The frequency result for this Modulation Scale will be the value of this register times the PLL Low Port Modulator Frequency Resolution; a typical value for the Frequency Resolution is 244.14 Hz.<br>The range of signed values that this register supports is -4096 to 4095 |
| 15-13<br>— | Reserved |
| 12-0<br>FSK_MODULATION_SCALE_0 | FSK Modulation Scale for a data 0<br>This register is used to provide the modulation level for a data 0 in FSK Protocols. The signed 12 bit value that is represented by this register is presented to the PLL as the Baseband Frequency Word. The frequency result for this Modulation Scale will be the value of this register times the PLL Low Port Modulator Frequency Resolution; a typical value for the Frequency Resolution is 244.14 Hz.<br>The range of signed values that this register supports is -4096 to 4095 |

#### 64.3.1.6.1.8 TX DFT Modulation Pattern (DFT_PATTERN)

##### *64.3.1.6.1.8.1 Offset*

| Register | Offset |
|---|---|
| DFT_PATTERN | 18h |

##### *64.3.1.6.1.8.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFT_MOD_PATTERN | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFT_MOD_PATTERN | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

##### *64.3.1.6.1.8.3 Fields*

| Field | Function |
|---|---|
| 31-0<br>DFT_MOD_PATTERN | DFT Modulation Pattern<br>In TX DFT Pattern Register modes, if TX_DFT_EN is set, the bits in this register will be shifted out as the DFT Modulation Data in a repeating loop starting with bit [0]. |

## 64.3.2 PLL Frequency Synthesizer

### 64.3.2.1 Introduction

#### 64.3.2.1.1 About the PLL Frequency Synthesizer

The PLL Frequency Synthesizer is a Digital Module that -

1. Selects the Radio Carrier Frequency
2. Coarse Tunes the VCO and Calibrates the High Port DAC
3. Controls the PLL Loop Divider value to lock the VCO at the Carrier Frequency

4. Applies the Baseband Frequency Word as High Port Modulation (HPM) to the VCO High Port DAC
5. Applies the Baseband Frequency Word as Low Port Modulation (LPM) to the PLL Loop Divider
6. Monitors the PLL lock status and flags unlocked conditions
7. Provides DFT features for evaluating the PLL Frequency Synthesizer during factory validation and testing

### 64.3.2.1.2 Block diagram

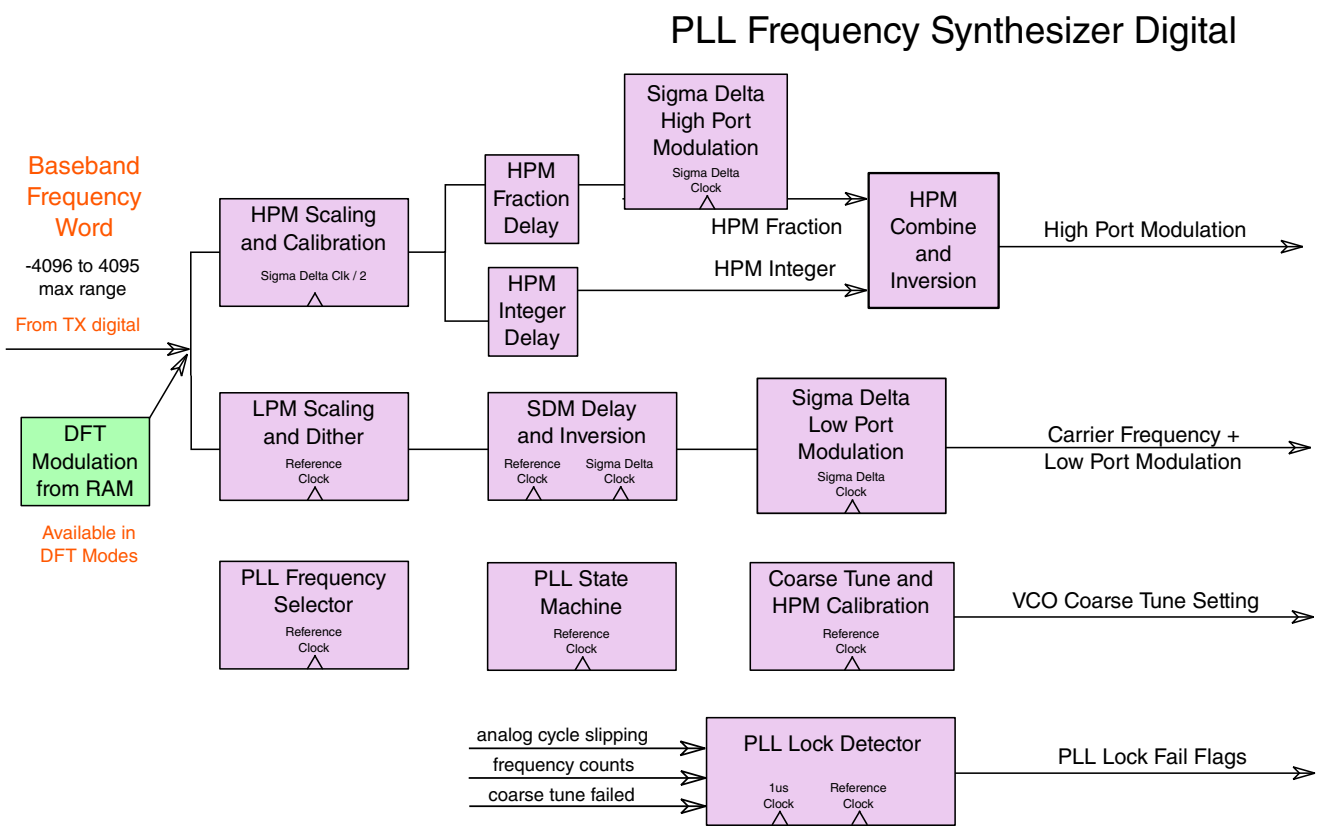



**Figure 64-20. PLL Frequency Synthesizer**

### 64.3.2.1.3 Baseband Frequency Word

The Transmitter Digital presents the PLL Frequency Synthesizer with a signed 13-bit Baseband Frequency word to modulate onto the RF Carrier Frequency. The Baseband Frequency Word range is -4096 to 4095, which provides a modulation range of +/- 1000 kHz when using a 32 MHz reference frequency with a PLL Minimum Frequency Step Size of 244.14 Hz.

#### 64.3.2.1.3.1 Manual Frequency Word

The Baseband Frequency word can be overridden by software if the MOD_DISABLE bit is set. In this case the PLL Digital will use the MODULATION_WORD_MANUAL register as the source of the modulation.

Note that due to the speeds at which the High Port and Low Port react to changes in the modulation word, and the speed at which software is able to effect such changes, for all practical purposes this can only support low frequency modulation.

## 64.3.2.2 Carrier Frequency Tuning

### 64.3.2.2.1 Carrier Frequency Control

#### 64.3.2.2.1.1 Radio Frequency Selection

The PLL Carrier Frequency is selected in these ways, depending on the Hop Configuration selected :

| HOP_TBL_CFG | PLL Carrier Frequency Selection |
|---|---|
| 0 or 1 | Channel Mode<br>The channel number, from the link layer or the Hop Frequency Word, is used by the Radio Frequency Selector to tune to a channel on the 1 MHz raster. |
| 2 | Frequency Offset Mode<br>The link layer channel number, from the Hop Frequency Word, is used by the Radio Frequency Selector to tune to a channel on the 1 MHz raster. A signed 9-bit Frequency Offset from the Hop Frequency Word is added to the channel selected. This provides approximately a +/- 1 MHz adjustment to the 1 MHz raster channels. The step size of the adjustment is 3.906 kHz for a 32 MHz clock and 3.174 kHz for a 26 MHz clock. |
| 3 | Direct Carrier Frequency Selection<br>An Integer is selected and the Hop Frequency Word is applied directly as a signed 15-bit value for the Numerator. This provides tuning with approximately a 2 kHz frequency resolution that covers the full 2400 - 2480 MHz range with a 32 MHz clock, and covers a range of 2400 - 2470 MHz with a 26 MHz clock. |
| 4,5,6,7 | Reserved |

The Radio mapping of the 128 RF channels on the 1 MHz raster, spanning the range from 2.360 GHz to 2.487 GHz, is shown in the figure below along with the Hop Frequency Word options.

There is also an option to allow an override of the Protocol Channel Frequency Mapping in order to allow any Protocol to choose to use any of the available Channel Frequency Mapping tables.

Note that there is also a full manual mode where the Integer and Fraction can be selected by software and thereby choose any RF Carrier Frequency in increments of the PLL SDM LSB (as long as the requested frequency is within the VCO specified range)

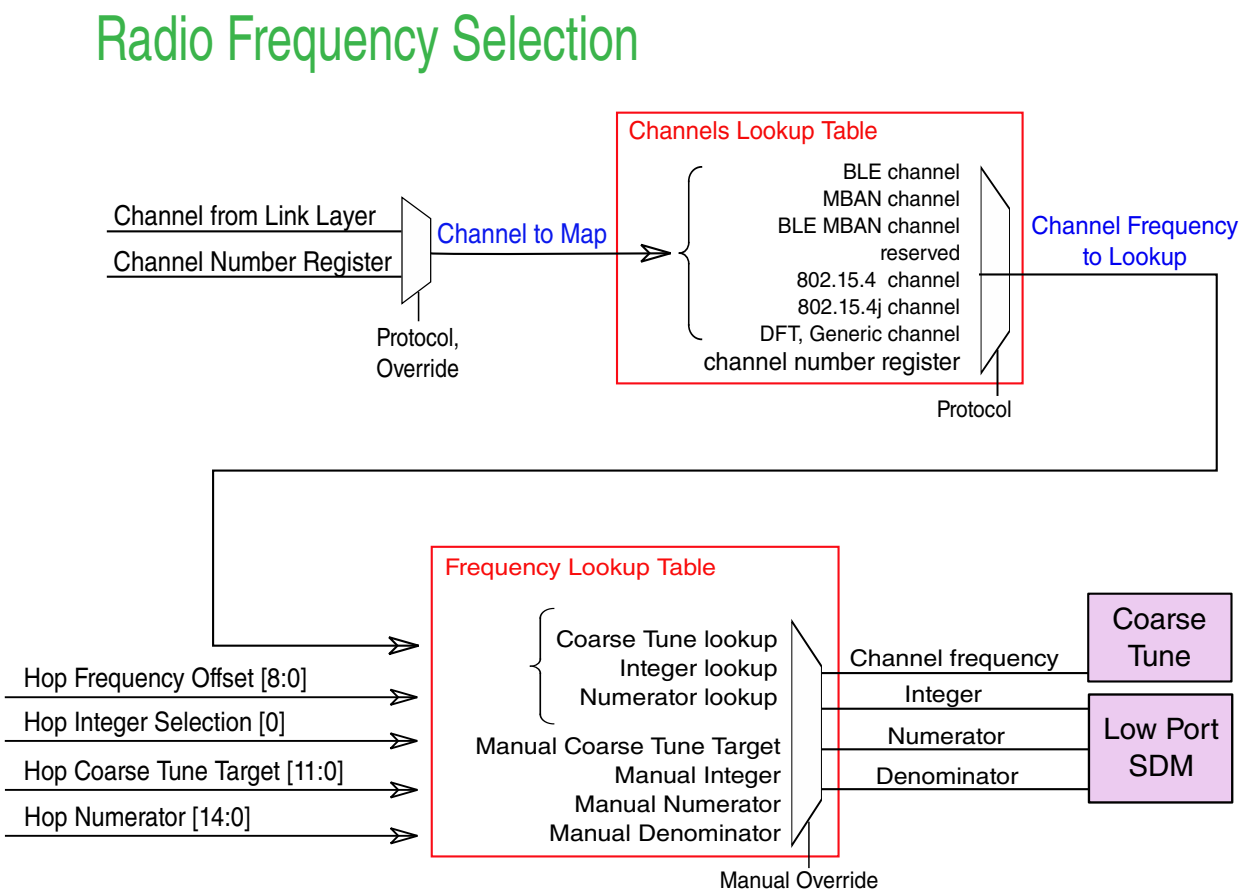



**Figure 64-21. Radio Frequency Selection**

### 64.3.2.2.1.2 RF Channels Supported

The channel frequencies for each protocol are shown in the tables below.

**Table 64-14. Radio Protocols (0 to 5) Channel Frequencies**

| Radio Protocols Supported, Channel Frequency in MHz | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BLE | | MBAN | | BLE/MBAN | | Reserved | | | | 802.15.4 | | 802.15.4j MBAN | |
| 0 | | 1 | | 2 | | 3 | | | | 4 | | 5 | |
| Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz |
| 0 | 2,402 | 0 | 2,360 | 0 | 2,402 | 0 | 2,400 | 43 | 2,443 | 11 | 2,405 | 0 | 2,363 |

*Table continues on the next page...*

## Table 64-14. Radio Protocols (0 to 5) Channel Frequencies (continued)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2,404 | 1 | 2,361 | 1 | 2,404 | 1 | 2,401 | 44 | 2,444 | 12 | 2,410 | 1 | 2,368 |
| 2 | 2,406 | 2 | 2,362 | 2 | 2,406 | 2 | 2,402 | 45 | 2,445 | 13 | 2,415 | 2 | 2,373 |
| 3 | 2,408 | 3 | 2,363 | 3 | 2,408 | 3 | 2,403 | 46 | 2,446 | 14 | 2,420 | 3 | 2,378 |
| 4 | 2,410 | 4 | 2,364 | 4 | 2,410 | 4 | 2,404 | 47 | 2,447 | 15 | 2,425 | 4 | 2,383 |
| 5 | 2,412 | 5 | 2,365 | 5 | 2,412 | 5 | 2,405 | 48 | 2,448 | 16 | 2,430 | 5 | 2,388 |
| 6 | 2,414 | 6 | 2,366 | 6 | 2,414 | 6 | 2,406 | 49 | 2,449 | 17 | 2,435 | 6 | 2,393 |
| 7 | 2,416 | 7 | 2,367 | 7 | 2,416 | 7 | 2,407 | 50 | 2,450 | 18 | 2,440 | 7 | 2,367 |
| 8 | 2,418 | 8 | 2,368 | 8 | 2,418 | 8 | 2,408 | 51 | 2,451 | 19 | 2,445 | 8 | 2,372 |
| 9 | 2,420 | 9 | 2,369 | 9 | 2,420 | 9 | 2,409 | 52 | 2,452 | 20 | 2,450 | 9 | 2,377 |
| 10 | 2,422 | 10 | 2,370 | 10 | 2,422 | 10 | 2,410 | 53 | 2,453 | 21 | 2,455 | 10 | 2,382 |
| 11 | 2,424 | 11 | 2,371 | 11 | 2,424 | 11 | 2,411 | 54 | 2,454 | 22 | 2,460 | 11 | 2,387 |
| 12 | 2,426 | 12 | 2,372 | 12 | 2,426 | 12 | 2,412 | 55 | 2,455 | 23 | 2,465 | 12 | 2,392 |
| 13 | 2,428 | 13 | 2,373 | 13 | 2,428 | 13 | 2,413 | 56 | 2,456 | 24 | 2,470 | 13 | 2,397 |
| 14 | 2,430 | 14 | 2,374 | 14 | 2,430 | 14 | 2,414 | 57 | 2,457 | 25 | 2,475 | 14 | 2,395 |
| 15 | 2,432 | 15 | 2,375 | 15 | 2,432 | 15 | 2,415 | 58 | 2,458 | 26 | 2,480 | | |
| 16 | 2,434 | 16 | 2,376 | 16 | 2,434 | 16 | 2,416 | 59 | 2,459 | | | | |
| 17 | 2,436 | 17 | 2,377 | 17 | 2,436 | 17 | 2,417 | 60 | 2,460 | | | | |
| 18 | 2,438 | 18 | 2,378 | 18 | 2,438 | 18 | 2,418 | 61 | 2,461 | | | | |
| 19 | 2,440 | 19 | 2,379 | 19 | 2,440 | 19 | 2,419 | 62 | 2,462 | | | | |
| 20 | 2,442 | 20 | 2,380 | 20 | 2,442 | 20 | 2,420 | 63 | 2,463 | | | | |
| 21 | 2,444 | 21 | 2,381 | 21 | 2,444 | 21 | 2,421 | 64 | 2,464 | | | | |
| 22 | 2,446 | 22 | 2,382 | 22 | 2,446 | 22 | 2,422 | 65 | 2,465 | | | | |
| 23 | 2,448 | 23 | 2,383 | 23 | 2,448 | 23 | 2,423 | 66 | 2,466 | | | | |
| 24 | 2,450 | 24 | 2,384 | 24 | 2,450 | 24 | 2,424 | 67 | 2,467 | | | | |
| 25 | 2,452 | 25 | 2,385 | 25 | 2,452 | 25 | 2,425 | 68 | 2,468 | | | | |
| 26 | 2,454 | 26 | 2,386 | 26 | 2,454 | 26 | 2,426 | 69 | 2,469 | | | | |
| 27 | 2,456 | 27 | 2,387 | 27 | 2,456 | 27 | 2,427 | 70 | 2,470 | | | | |
| 28 | 2,458 | 28 | 2,388 | 28 | 2,458 | 28 | 2,428 | 71 | 2,471 | | | | |
| 29 | 2,460 | 29 | 2,389 | 29 | 2,460 | 29 | 2,429 | 72 | 2,472 | | | | |
| 30 | 2,462 | 30 | 2,390 | 30 | 2,390 | 30 | 2,430 | 73 | 2,473 | | | | |
| 31 | 2,464 | 31 | 2,391 | 31 | 2,391 | 31 | 2,431 | 74 | 2,474 | | | | |
| 32 | 2,466 | 32 | 2,392 | 32 | 2,392 | 32 | 2,432 | 75 | 2,475 | | | | |
| 33 | 2,468 | 33 | 2,393 | 33 | 2,393 | 33 | 2,433 | 76 | 2,476 | | | | |
| 34 | 2,470 | 34 | 2,394 | 34 | 2,394 | 34 | 2,434 | 77 | 2,477 | | | | |
| 35 | 2,472 | 35 | 2,395 | 35 | 2,395 | 35 | 2,435 | 78 | 2,478 | | | | |
| 36 | 2,474 | 36 | 2,396 | 36 | 2,396 | 36 | 2,436 | 79 | 2,479 | | | | |
| 37 | 2,476 | 37 | 2,397 | 37 | 2,397 | 37 | 2,437 | 80 | 2,480 | | | | |
| 38 | 2,478 | 38 | 2,398 | 38 | 2,398 | 38 | 2,438 | 81 | 2,481 | | | | |
| 39 | 2,480 | 39 | 2,399 | 39* | 2,480 | 39 | 2,439 | 82 | 2,482 | | | | |
| | | | | | | 40 | 2,440 | 83 | 2,483 | | | | |

*Table continues on the next page...*

**Table 64-14. Radio Protocols (0 to 5) Channel Frequencies (continued)**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 41 | 2,441 | | | | | | |
| | | | | | | 42 | 2,442 | | | | | | |

* The BLE MBAN Channel Remap bit (BMR) controls the frequency mapping for channel 39 in BLE/MBAN, 2480 MHz is the default, but this channel can be remapped to 2399 MHz by setting BMR to a 1.

**Table 64-15. Radio Protocols (6 to 9) Channel Frequencies**

| Radio Protocols Supported, Channel Frequency in MHz | | | | | |
|---|---|---|---|---|---|
| DFT, Generic | | | | | |
| 6,7,8,9 | | | | | |
| Channel | MHz | Channel | MHz | Channel | MHz |
| 0 | 2,360 | 43 | 2,403 | 86 | 2,446 |
| 1 | 2,361 | 44 | 2,404 | 87 | 2,447 |
| 2 | 2,362 | 45 | 2,405 | 88 | 2,448 |
| 3 | 2,363 | 46 | 2,406 | 89 | 2,449 |
| 4 | 2,364 | 47 | 2,407 | 90 | 2,450 |
| 5 | 2,365 | 48 | 2,408 | 91 | 2,451 |
| 6 | 2,366 | 49 | 2,409 | 92 | 2,452 |
| 7 | 2,367 | 50 | 2,410 | 93 | 2,453 |
| 8 | 2,368 | 51 | 2,411 | 94 | 2,454 |
| 9 | 2,369 | 52 | 2,412 | 95 | 2,455 |
| 10 | 2,370 | 53 | 2,413 | 96 | 2,456 |
| 11 | 2,371 | 54 | 2,414 | 97 | 2,457 |
| 12 | 2,372 | 55 | 2,415 | 98 | 2,458 |
| 13 | 2,373 | 56 | 2,416 | 99 | 2,459 |
| 14 | 2,374 | 57 | 2,417 | 100 | 2,460 |
| 15 | 2,375 | 58 | 2,418 | 101 | 2,461 |
| 16 | 2,376 | 59 | 2,419 | 102 | 2,462 |
| 17 | 2,377 | 60 | 2,420 | 103 | 2,463 |
| 18 | 2,378 | 61 | 2,421 | 104 | 2,464 |
| 19 | 2,379 | 62 | 2,422 | 105 | 2,465 |
| 20 | 2,380 | 63 | 2,423 | 106 | 2,466 |
| 21 | 2,381 | 64 | 2,424 | 107 | 2,467 |
| 22 | 2,382 | 65 | 2,425 | 108 | 2,468 |
| 23 | 2,383 | 66 | 2,426 | 109 | 2,469 |
| 24 | 2,384 | 67 | 2,427 | 110 | 2,470 |
| 25 | 2,385 | 68 | 2,428 | 111 | 2,471 |
| 26 | 2,386 | 69 | 2,429 | 112 | 2,472 |
| 27 | 2,387 | 70 | 2,430 | 113 | 2,473 |

*Table continues on the next page...*

**Table 64-15. Radio Protocols (6 to 9) Channel Frequencies (continued)**

| | | | | | |
|---|---|---|---|---|---|
| 28 | 2,388 | 71 | 2,431 | 114 | 2,474 |
| 29 | 2,389 | 72 | 2,432 | 115 | 2,475 |
| 30 | 2,390 | 73 | 2,433 | 116 | 2,476 |
| 31 | 2,391 | 74 | 2,434 | 117 | 2,477 |
| 32 | 2,392 | 75 | 2,435 | 118 | 2,478 |
| 33 | 2,393 | 76 | 2,436 | 119 | 2,479 |
| 34 | 2,394 | 77 | 2,437 | 120 | 2,480 |
| 35 | 2,395 | 78 | 2,438 | 121 | 2,481 |
| 36 | 2,396 | 79 | 2,439 | 122 | 2,482 |
| 37 | 2,397 | 80 | 2,440 | 123 | 2,483 |
| 38 | 2,398 | 81 | 2,441 | 124 | 2,484 |
| 39 | 2,399 | 82 | 2,442 | 125 | 2,485 |
| 40 | 2,400 | 83 | 2,443 | 126 | 2,486 |
| 41 | 2,401 | 84 | 2,444 | 127 | 2,487 |
| 42 | 2,402 | 85 | 2,445 | | |

#### 64.3.2.2.1.3 Channel Selector

During typical Radio usage, the channel selection is done automatically by the Link Layer.

Channel selection can be overridden using the PLL_CHAN_MAP register bits BOC (BLE protocols 0 and 2) and ZOC (802.15.4 protocols 4 and 5). If these are set then the CHANNEL_NUM register selects the Radio Channel Number instead of the Link Layer.

#### 64.3.2.2.1.4 Manual Carrier Frequency

The internal frequency mapping can be bypassed and software can directly control the Low Port Sigma Delta inputs to select any channel frequency in the VCO's range (in increments of the Low Port Minimum Frequency Step Size).

The Radio Channel Frequency override is manually selected by setting the SDM_MAP_DISABLE bit along with the LPM_INTG, LPM_NUM, and LPM_DENOM register values.

Any modulation from the TX Digital will be added to this manual frequency selection.

The Manual carrier frequency selected can be calculated using the formula below:

Radio Carrier Frequency = ((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM))

WARNING : The fraction (LPM_NUM / LPM_DENOM) must be in the range of -0.55 to +0.55 for valid Sigma Delta Modulator operation.

#### 64.3.2.2.2 Hop Frequency Word

The Hop Frequency Word is used to provide the PLL with Channel Selection, Frequency Offset, Integer Selection, and Direct Numerator Control of the Low Port Sigma Delta Modulator.

The figure below shows an overview of how the Hop Table interfaces to the PLL.

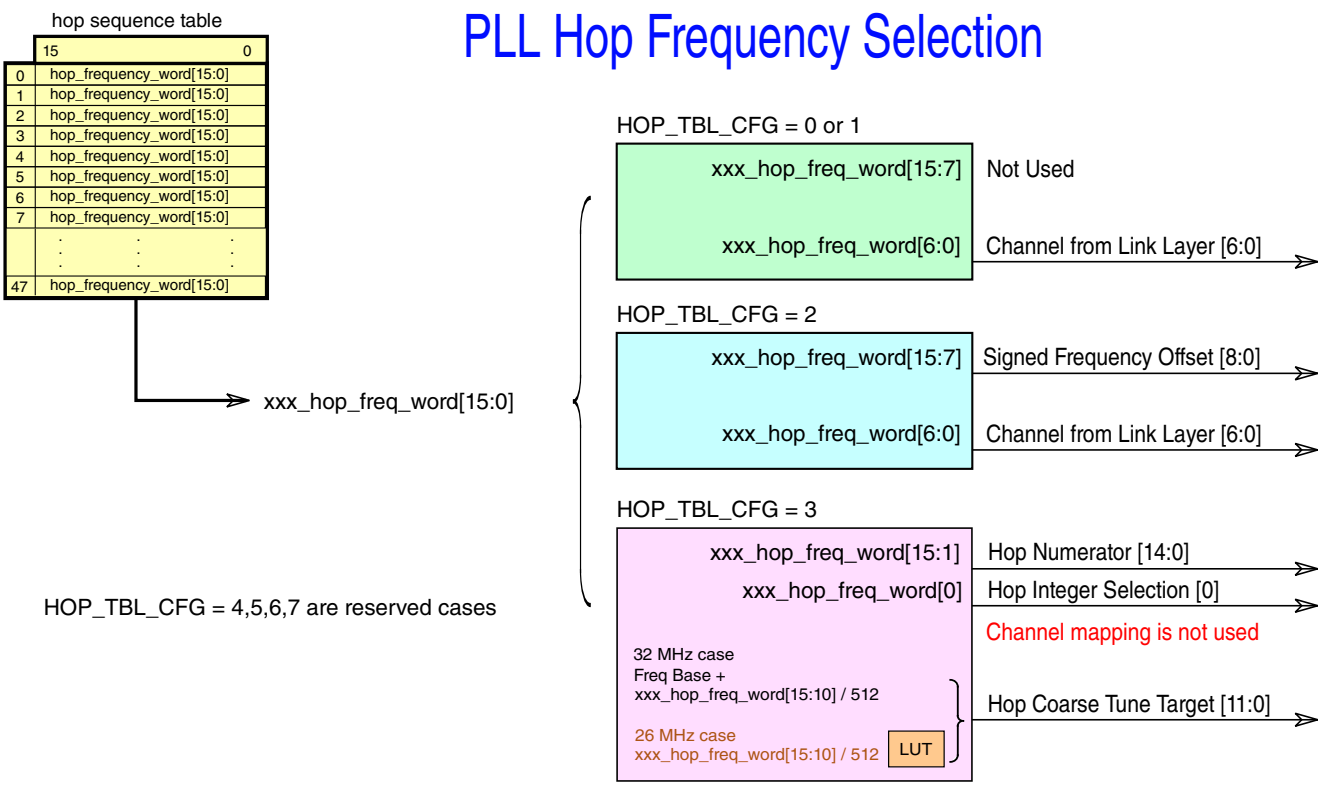



**Figure 64-22. PLL Hop Frequency Selection**

For much of the Radio usage the mapping of the 128 RF channels on a 1 MHz raster is sufficient. However, there are additional cases where Carrier Frequency Tuning at a finer resolution is desired and the Hop Frequency Word contains the information needed to get these finer resolutions.

The table below shows some of these desired Channel Spacings.

| Modulation Type | Data Rate | Sensitivity Estimate (dBm) | Frequency Deviation (kHz) | 99.5% Signal BW (kHz) | Typical Channel Spacing (kHz) | Notes | Desired Offsets from 1 MHz Channel Map |
|---|---|---|---|---|---|---|---|
| GFSK BT=0.5, h=0.5 | 1000 | -96 | 250 | 1105 | 2000 | Bluetooth-LE | 1000 |
| | 500 | -98 | 125 | 552.5 | 1000 | | 0 |
| | 250 | -100 | 62.5 | 276.3 | 500 | Gen FSK 4 | 500 |
| GFSK, BT=0.5, h= 0.32 | 1000 | -92 | 160 | 926 | 1000 | ANT | 0 |
| | 500 | -94 | 80 | 463 | 600 | | 400 |
| | 250 | -96 | 40 | 231.5 | 400 | | 600 |
| GFSK, BT=0.5, h=0.7 | 1000 | -96 | 350 | 1278 | 2000 | | 1000 |
| | 500 | -99 | 175 | 639 | 1000 | Gen FSK 1 | 0 |
| | 250 | -101 | 87.5 | 319.5 | 500 | | 500 |
| GFSK, BT=0.5, h=1.0 | 1000 | -93 | 500 | 1765 | 3000 | | 2000 |
| | 500 | -96 | 250 | 882.5 | 1500 | | 500 |
| | 250 | -98 | 125 | 441.3 | 750 | Gen FSK 5 | 250 |
| GMSK BT=0.3 | 1000 | -93 | 250 | 975 | 1200 | Gen FSK 2 | 200 |
| | 500 | -95 | 125 | 487.5 | 600 | | 400 |
| | 250 | -97 | 62.5 | 243.8 | 400 | | 600 |
| GMSK BT=0.7 | 1000 | -96 | 250 | 1167 | 2000 | | 1000 |
| | 500 | -98 | 125 | 583.5 | 1000 | | 0 |
| | 250 | -100 | 62.5 | 291.8 | 500 | | 500 |
| MSK/FSK (h=0.5, half-sin) | 1000 | -93 | 250 | 1605 | 2500 | | 1500 |
| | 500 | -95 | 125 | 802.5 | 1250 | Gen FSK 3 | 250 |
| | 500 | -95 | 125 | 802.5 | 1300 | Gen FSK 4 | 300 |
| | 250 | -98 | 62.5 | 401.3 | 625 | | 375 |
| | 250 | -98 | 62.5 | 401.3 | 650 | | 350 |

**Figure 64-23. PLL Typical Channel Frequency Spacings**

#### 64.3.2.2.2.1 Frequency Offset Mode

In order to support the desired Channel Spacings, a Frequency Offset Mode is provided that allows the Carrier Frequency to be offset from the selected channel in approximately 4 kHz steps. The tables below shows the resulting Frequency Error for some of the channel spacings.

The Hop Frequency Word contains a 1 MHz channel number and a 9-bit signed offset value. The offset is left shifted and then applied to the Numerator from the channel lookup table to create the Numerator applied to the Low Port SDM.

The Frequency Offset Range is approximately +/- 1 MHz

The Coarse Tune Target is not effected in this mode.

| Integer | Min Fraction | Max Fraction | SDM Denominator | Min Numerator | Max Numerator | Min Frequency | Max Frequency | Frequency Range |
|---|---|---|---|---|---|---|---|---|
| 38 | -0.5 | 0.5 | 1,048,576 | -16,320 | 16,320 | 2,431,003,906 | 2,432,996,094 | 1,992,188 |

| | | |
|---|---|---|
| Fref | 32 | MHz |
| Denom_resolution | 20 | bits |
| Denominator | 1,048,576 | |
| Channel bits | 7 | |
| Integer bits | 0 | |
| Hop numerator width, plus sign bit | 8 | |
| Left shift Hop Num | 6 | |
| min numerator value | 1 | |
| min/max signed numerator value | 255 | |

| | Numerator Value | Numerator Shifted Value | Frequency Result | Desired Frequency | Frequency Error |
|---|---|---|---|---|---|
| | 1 | 64 | 3,906 | | |
| | 8 | 512 | 31,250 | | |
| | 16 | 1,024 | 62,500 | | |
| | 32 | 2,048 | 125,000 | | |
| Desired Offsets from 1 MHz Channel Map | 51 | 3,264 | 199,219 | 200,000 | -0.3906% |
| | 64 | 4,096 | 250,000 | 250,000 | 0.0000% |
| | 77 | 4,928 | 300,781 | 300,000 | 0.2604% |
| | 90 | 5,760 | 351,563 | 350,000 | 0.4464% |
| | 96 | 6,144 | 375,000 | 375,000 | 0.0000% |
| | 102 | 6,528 | 398,438 | 400,000 | -0.3906% |
| | 128 | 8,192 | 500,000 | 500,000 | 0.0000% |
| | 154 | 9,856 | 601,563 | 600,000 | 0.2604% |
| | 160 | 10,240 | 625,000 | | |
| | 192 | 12,288 | 750,000 | | |
| | 224 | 14,336 | 875,000 | | |
| | 240 | 15,360 | 937,500 | | |
| | 255 | 16,320 | 996,094 | | |

**Figure 64-24. PLL Hop Frequency Offset Mode, 32 MHz**

| Integer | Min Fraction | Max Fraction | SDM Denominator | Min Numerator | Max Numerator | Min Frequency | Max Frequency | Frequency Range |
|---|---|---|---|---|---|---|---|---|
| 46 | -0.5 | 0.5 | 1,048,576 | -16,320 | 16,320 | 2,391,190,674 | 2,392,809,326 | 1,618,652 |

| | | |
|---|---|---|
| Fref | 26 | MHz |
| Denom_resolution | 20 | bits |
| Denominator | 1,048,576 | |
| Channel bits | 7 | |
| Integer bits | 0 | |
| Hop numerator width, plus sign bit | 8 | |
| Left shift Hop Num | 6 | |
| min numerator value | 1 | |
| min/max signed numerator value | 255 | |

| | Numerator Value | Numerator Shifted Value | Frequency Result | Desired Frequency | Frequency Error |
|---|---|---|---|---|---|
| | 1 | 64 | 3,174 | | |
| | 8 | 512 | 25,391 | | |
| | 16 | 1,024 | 50,781 | | |
| | 32 | 2,048 | 101,563 | | |
| Desired Offsets from 1 MHz Channel Map | 63 | 4,032 | 199,951 | 200,000 | -0.0244% |
| | 79 | 5,056 | 250,732 | 250,000 | 0.2930% |
| | 95 | 6,080 | 301,514 | 300,000 | 0.5046% |
| | 110 | 7,040 | 349,121 | 350,000 | -0.2511% |
| | 118 | 7,552 | 374,512 | 375,000 | -0.1302% |
| | 126 | 8,064 | 399,902 | 400,000 | -0.0244% |
| | 158 | 10,112 | 501,465 | 500,000 | 0.2930% |
| | 189 | 12,096 | 599,854 | 600,000 | -0.0244% |
| | 160 | 10,240 | 507,813 | | |
| | 192 | 12,288 | 609,375 | | |
| | 224 | 14,336 | 710,938 | | |
| | 240 | 15,360 | 761,719 | | |
| | 255 | 16,320 | 809,326 | | |

**Figure 64-25. PLL Hop Frequency Offset Mode, 26 MHz**

### 64.3.2.2.2.2 Direct Carrier Frequency Mode

For a finer Carrier Frequency resolution, the Direct Carrier Frequency mode is provided which allows approximately a 2 kHz step size. This method allows for complete coverage of the 2400 - 2480 MHz range while using a 32 MHz reference clock, and a range of 2400 - 2470 MHz while using a 26 MHz reference clock. The Frequency Errors from a 2 kHz raster are shown in the tables below.

The Hop Frequency Word contains the Integer choice and a 15-bit signed value to be left shifted and then applied as the Numerator to the Low Port SDM.

The Frequency Offset Range is +/- 32 MHz for a 32 MHz clock, and +/- 26 MHz for a 26 MHz clock.

In this mode the Coarse Tune Target is specially calculated as follows :

- For 32 MHz clock operations, the signed 15-bit Hop Frequency Word is divided by 512 and added to the selected Frequency Base.
  - Frequency Base for integer 38 is 2432, for integer 39 it is 2496
- For 26 MHz clock operations, the signed 15-bit Hop Frequency Word is divided by 512 and then a Look Up Table is used to select the best Coarse Tune Target.

| Integer | Min Fraction | Max Fraction | SDM Denominator | Min Numerator | Max Numerator | Min Frequency | Max Frequency | Frequency Range |
|---|---|---|---|---|---|---|---|---|
| 38 | -0.5 | 0.5 | 1,048,576 | -524,288 | 524,256 | 2,400,000,000 | 2,463,998,047 | 63,998,047 |
| 39 | -0.5 | 0.5 | 1,048,576 | -524,288 | 524,256 | 2,464,000,000 | 2,527,998,047 | 63,998,047 |

| | | |
|---|---|---|
| Fref | 32 | MHz |
| Denom_resolution | 20 | bits |
| Denominator | 1,048,576 | |
| Channel bits | 0 | |
| Integer bits | 1 | (38,39) or (46, 47) |
| Hop numerator width, plus sign bit | 14 | |

| Left shift Hop Num by 5 | 2 kHz Freq Steps | Numerator Value | Numerator Shifted Value | Frequency Result | Frequency Error |
|---|---|---|---|---|---|
| | 2,000 | 1 | 32 | 1,953 | -2.3438% |
| | 4,000 | 2 | 64 | 3,906 | -2.3438% |
| | 6,000 | 3 | 96 | 5,859 | -2.3438% |
| | 8,000 | 4 | 128 | 7,813 | -2.3438% |
| | 42,000 | 22 | 704 | 42,969 | 2.3065% |
| | 44,000 | 23 | 736 | 44,922 | 2.0952% |
| | 46,000 | 24 | 768 | 46,875 | 1.9022% |
| | 48,000 | 25 | 800 | 48,828 | 1.7253% |
| | 62,000 | 32 | 1024 | 62,500 | 0.8065% |
| | 64,000 | 33 | 1056 | 64,453 | 0.7080% |
| | 66,000 | 34 | 1088 | 66,406 | 0.6155% |
| | 68,000 | 35 | 1120 | 68,359 | 0.5285% |
| | 31,992,000 | 16380 | 524160 | 31,992,188 | 0.0006% |
| | 31,994,000 | 16381 | 524192 | 31,994,141 | 0.0004% |
| | 31,996,000 | 16382 | 524224 | 31,996,094 | 0.0003% |
| | 31,998,000 | 16383 | 524256 | 31,998,047 | 0.0001% |
| | 32,000,000 | 16384 | 524288 | 32,000,000 | 0.0000% |
| Max Positive Value | | 16383 | | | |

**Figure 64-26. PLL Hop Direct Carrier Frequency Selection, 32 MHz**

| Integer | Min Fraction | Max Fraction | SDM Denominator | Min Numerator | Max Numerator | Min Frequency | Max Frequency | Frequency Range |
|---|---|---|---|---|---|---|---|---|
| 46 | -0.5 | 0.5 | 1,048,576 | -524,288 | 524,256 | 2,366,000,000 | 2,417,998,413 | 51,998,413 |
| 47 | -0.5 | 0.5 | 1,048,576 | -524,288 | 524,256 | 2,418,000,000 | 2,469,998,413 | 51,998,413 |

| | | |
|---|---|---|
| Fref | 26 | MHz |
| Denom_resolution | 20 | bits |
| Denominator | 1,048,576 | |
| Channel bits | 0 | |
| Integer bits | 1 | (38,39) or (46, 47) |
| Hop numerator width, plus sign bit | 14 | |

Note that we would need to push the fraction on the LP SDM to cover the range above 2470, or add a bit somewhere

| Left shift Hop Num by 5 | 2 kHz Freq Steps | Numerator Value | Numerator Shifted Value | Frequency Result | Frequency Error |
|---|---|---|---|---|---|
| | 2,000 | 1 | 32 | 1,587 | -20.6543% |
| | 4,000 | 3 | 96 | 4,761 | 19.0186% |
| | 6,000 | 4 | 128 | 6,348 | 5.7943% |
| | 8,000 | 5 | 160 | 7,935 | -0.8179% |
| | 42,000 | 26 | 832 | 41,260 | -1.7625% |
| | 44,000 | 28 | 896 | 44,434 | 0.9854% |
| | 46,000 | 29 | 928 | 46,021 | 0.0446% |
| | 48,000 | 30 | 960 | 47,607 | -0.8179% |
| | 62,000 | 39 | 1248 | 61,890 | -0.1780% |
| | 64,000 | 40 | 1280 | 63,477 | -0.8179% |
| | 66,000 | 42 | 1344 | 66,650 | 0.9854% |
| | 68,000 | 43 | 1376 | 68,237 | 0.3490% |
| | 25,992,000 | 16379 | 524128 | 25,992,065 | 0.0003% |
| | 25,994,000 | 16380 | 524160 | 25,993,652 | -0.0013% |
| | 25,996,000 | 16381 | 524192 | 25,995,239 | -0.0029% |
| | 25,998,000 | 16383 | 524256 | 25,998,413 | 0.0016% |
| | 26,000,000 | 16384 | 524288 | 26,000,000 | 0.0000% |
| Max Positive Value | | 16383 | | | |

**Figure 64-27. PLL Hop Direct Carrier Frequency Selection, 26 MHz**

## 64.3.2.3 VCO Calibration

### 64.3.2.3.1 Coarse Tune

The VCO Coarse Tune is carried out by a Successive Approximation Register method that computes the best coarse tune setting for the VCO in successive steps from MSB to LSB.

This SAR method uses a Frequency Meter count value that is successively compared to expected counts to determine the best coarse tune setting for the VCO.

The coarse tune setting can be overridden for test and validation purposes, and the individual frequency counts used at each successive step for the coarse tune setting are available for viewing in DTEST mode and also as register values in the PLL register space.

#### 64.3.2.3.2 High Port DAC Calibration

The objective of the HPM DAC Calibration is to determine the DAC Step Size and estimate the ratio of this DAC unit step to the Low Port Minimum Frequency Step Size.

Once the step size of the High Port DAC is known, a scaling factor is applied to correctly map the baseband frequency word to the High Port modulation.

The length of the calibration time is set using the HPM_CAL_TIME register and the size of the DAC array used is selected using the HPM_CAL_ARRAY_SIZE register.

The register HPM_COUNT_ADJUST[3:0] provides a (-8/+7) range of adjustment to the HPM Calibration Count Difference for the HPM Calibration lookup table to allow for any count error bias that may be present. This adjusted Count Difference is used with a Lookup Table to choose the HPM Calibration Factor

##### 64.3.2.3.2.1 HPM CAL Factor

The HPM_CAL_FACTOR register shows the value that is currently being used by High Port Modulation Multiplier. This value is from the most recent High Port calibration unless it is being overridden using the HP_CAL_DISABLE and HPM_CAL_FACTOR_MANUAL registers.

### 64.3.2.4 High Port Modulation

The High Port Modulator multiplies the HPM Calibration Factor and the PLL Baseband Frequency Word to get the High Port Modulation Word.

The High Port Modulation Word is represented by both an Integer and a Fractional portion. The Fraction is processed by the High Port Sigma Delta Modulator and the result is combined with the Integer and used to control the VCO High Port DAC.

#### 64.3.2.4.1 High Port DAC

The High Port Digital to Analog Converter is designed to take the High Port Modulation Word and convert it to a control voltage that is applied as modulation to the VCO High Port.

#### 64.3.2.4.1.1 HPM DAC Settings

The HPM DAC settings during calibration and normal operation are shown in the table below.

Warning : TX operation with On-Air modulation ranges greater than +/- 250 kHz (+/- 500 kHz at the VCO) require that the VCO Kvm be adjusted using the SY_VCO_KVM[2:0] register.

**Table 64-16. Nominal HPM DAC parameters in various modes**

| Radio TX Mode | | DAC max swing (mV) | DAC output common mode (mV) | DAC swing during operation | Vcm (mV) / HPM_VCM | OPAMP gain setting (R2/R1) / HPM_FDB_RES | Common mode at varactor (mV) | Swing at Varactor input (mV) | Kvm (MHz/Kv) | Deviation at VCO (KHz)[1] |
|---|---|---|---|---|---|---|---|---|---|---|
| 802.15.4/ 2000kHz | HPM calibration | +/-82 | 82 | +/-82 | 288 / 3'b001 | 2/ 2'b01 | 700 | +/-164 | 20 | +/-3280 |
| | TX operation | +/-82 | 82 | +/-51.2 | 391 / 3'b000 | 1/ 2'b00 | 700 | +/51.2 | 20 | +/-1024 |
| BLE/ 1000kHz | HPM calibration | +/-82 | 82 | +/-82 | 288 / 3'b001 | 2 / 2'b01 | 700 | +/-164 | 10 | +/-1640 |
| | TX operation | +/-82 | 82 | +/-51.2 | 391 / 3'b000 | 1/ 2'b00 | 700 | +/51.2 | 10 | +/-512 |
| 500kHz | HPM calibration | +/-82 | 82 | +/-82 | 288 / 3'b001 | 2/ 2'b01 | 700 | +/-164 | 10 | +/-1640 |
| | TX operation | +/-82 | 82 | +/-25.6 | 391 / 3'b000 | 1/ 2'b00 | 700 | +/25.6 | 10 | +/-256 |
| 250kHz | HPM calibration | +/-82 | 82 | +/-82 | 288 / 3'b001 | 2/ 2'b01 | 700 | +/-164 | 10 | +/-1640 |
| | TX operation | +/-82 | 82 | +/-12.8 | 391 / 3'b000 | 1/ 2'b00 | 700 | +/12.8 | 10 | +/-128 |

1. On-Air modulation frequency is a divide by 2 of the VCO frequency.

#### 64.3.2.4.1.2 HPM DAC Bump

The HPM DAC Calibration is normally done at 2X the On-Air Modulation Range, this is accomplished by writing the PLL HPM Analog Bump Control (PLL_HPM_BUMP) register bits:

- HPM_VCM_CAL, value applied during HPM DAC Calibration
- HPM_FDB_RES_CAL, value applied during HPM DAC Calibration
- HPM_VCM_TX, value applied during Radio Transmissions
- HPM_FDB_RES_TX, value applied during Radio Transmissions

If the registers are written in such a way that the DAC range is not doubled, then the HPM_CAL_NOT_BUMPED bit must be set to change the math in the HPM Calibration.

#### 64.3.2.4.1.3 HPM DAC 1 MHz Range

TX operation with On-Air modulation ranges greater than +/- 250 kHz (+/- 500 kHz at the VCO) require that the VCO Kvm be adjusted using the SY_VCO_KVM[2:0] register.

This VCO Kvm setting should be made prior to HPM DAC calibration.

### 64.3.2.4.2 High Port Sigma Delta

The High Port fractional modulation is implemented using a 2nd-order MASH Sigma Delta Modulator (SDM), which converts the 6-bit fractional portion of the high port modulation word to a 3-state thermometric code {-1,0,+1}. The resulting code is then added to the integer portion of the high port modulation word and the final result is applied to the HPM DAC.

#### 64.3.2.4.2.1 HPM SDM Dither

In addition to the nominal SDM functionality, the HPM SDM design includes a zero-mean dithering to mitigate idle tones that can occur when modulating a string of ones or zeros.

The Dither is implemented using a Linear-Feedback Shift Register (LFSR) which provides a 1-bit pseudo-randomization that adds or subtracts 1-bit to the 6-bit fractional portion before it is processed by the Sigma Delta Modulator.

The length of the LFSR can be selected using the HPM_LFSR_SIZE register, and the scale can be changed using the HPM_DTH_SCL bit. The Dither can also be disabled by clearing the HPM_DTH_EN bit.

### 64.3.2.4.3 High Port Delay, Disable, Invert

The HPM Integer and Fraction portions each have an array of programmable delays available to allow the HPM modulations to be matched to each other and to the LPM modulation and the delay through the PLL Loop Divider and onto the VCO feedback loop.

The HPM delay steps are in increments of the PLL Sigma Delta Clock divided by 2, and the LPM SDM delay steps are in increments of the PLL Sigma Delta Clock. All three delay step settings are in the range of {0 to 15}.

There are several inversion bits that can be used if needed:

- HPM_CAL_INVERT, Invert the High Port Modulator during Calibration
- HPM_MOD_IN_INVERT, Invert the input word to the High Port Modulator

- HPM_INTEGER_INVERT, Invert the High Port Modulation Integer portion
- HPM_SDM_OUT_INVERT, Invert the HPM SDM Fractional Output

High Port Modulation can be disabled by setting the HPM_MOD_DISABLE bit, in this case the HPM_MOD_MANUAL register value is applied to the VCO High Port.

### 64.3.2.5 Low Port Modulation

The Low Port Modulator uses a 3rd order Sigma-Delta to produce a fractional division (Fractional-N Frequency Synthesis) to control the PLL Loop Divider and tune the VCO to the Radio Carrier Frequency.

The Low Port Baseband Frequency Word is added as modulation to the Sigma-Delta input.

#### 64.3.2.5.1 Low Port Resolution

The table below shows the PLL Minimum Frequency Step Sizes for Modulation and Carrier Frequency tuning.

Note that the minimum step size for carrier frequency tuning is a divide by 512 of the modulation step sizes.

**Table 64-17. PLL Minimum Frequency Step Size**

| Ref Clk Freq (MHz) | 32 | 26 |
|---|---|---|
| PLL Minimum Modulation Step (Hz) | 244.14 | 198.36 |
| PLL Minimum Carrier Frequency Step (Hz) | 0.9537 | 0.7749 |

#### 64.3.2.5.2 Low Port Sigma-Delta

The Integer, Numerator, and Denominator are presented at the input to the Sigma-Delta Modulator.

A 3-level quantized Sigma-Delta output signal {-1, 0, +1} is generated and combined with the Integer divide value to produce the Modulus Divide Ratio for the PLL Loop Divider. The resulting RF Frequency can be calculated using the formula below:

RF Frequency = ((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM))

WARNING : The fraction (LPM_NUM / LPM_DENOM) must be in the range of -0.55 to +0.55 for valid Sigma Delta Modulator operation.

#### 64.3.2.5.2.1 LPM SDM Dither

While in Transmit mode the input to the SDM is naturally dithered by the Low Port modulation, which keeps the SDM from generating a pseudo-periodic output. However, in Receive mode, the SDM can potentially generate idle tones (dominant frequency modes at its output) when the input to the SDM is a constant. And in either Transmit or Receive mode the LPM SDM can generate idle tones if the Fraction is near zero.

To mitigate these impairments, the Numerator input to the SDM is dithered whenever the Numerator is in the near zero range { -64 to 63 }, or whenever the Radio is in Receive Mode.

This dither operation can also be overridden using the LPM_D_OVRD and LPM_D_CTRL bits.

### 64.3.2.5.3 Low Port Delay, Disable, Invert

The Low Port Modulator has an array of programmable delays available to allow the HPM modulation to be matched to the LPM modulation and the delay through the PLL Loop Divider and onto the VCO feedback loop.

The Low Port Modulation can be delayed at the PLL Loop Divider by delaying the input to the LPM SDM (LPM_SDM_DELAY).

There is an option to invert the Numerator (LPM_SDM_INV) before it is applied to the SDM in order to invert the sigma-delta modulation to the PLL.

The Low Port Modulation of the Loop Divider can be disabled by setting the LPM_DISABLE bit. When the modulation is disabled the LPM SDM will continue to tune the VCO to the Radio Carrier Frequency.

## 64.3.2.6 Lock Detector

The PLL Lock Detector has three methods to measure the stability of the PLL and check that it is tuned to the selected Radio Frequency.

The real-time results of the PLL Lock Detect module can be read as register bits, and flags are set and can be cleared to provide software long-term access to any PLL unlock events.

The TSM can be configured to choose any or all of the PLL Lock Detect module methods to generate an Abort and avoid violating any FCC On-Air or other radio requirements. This abort option is also a power saving feature and avoids continuing a radio operation that will be unsuccessful.

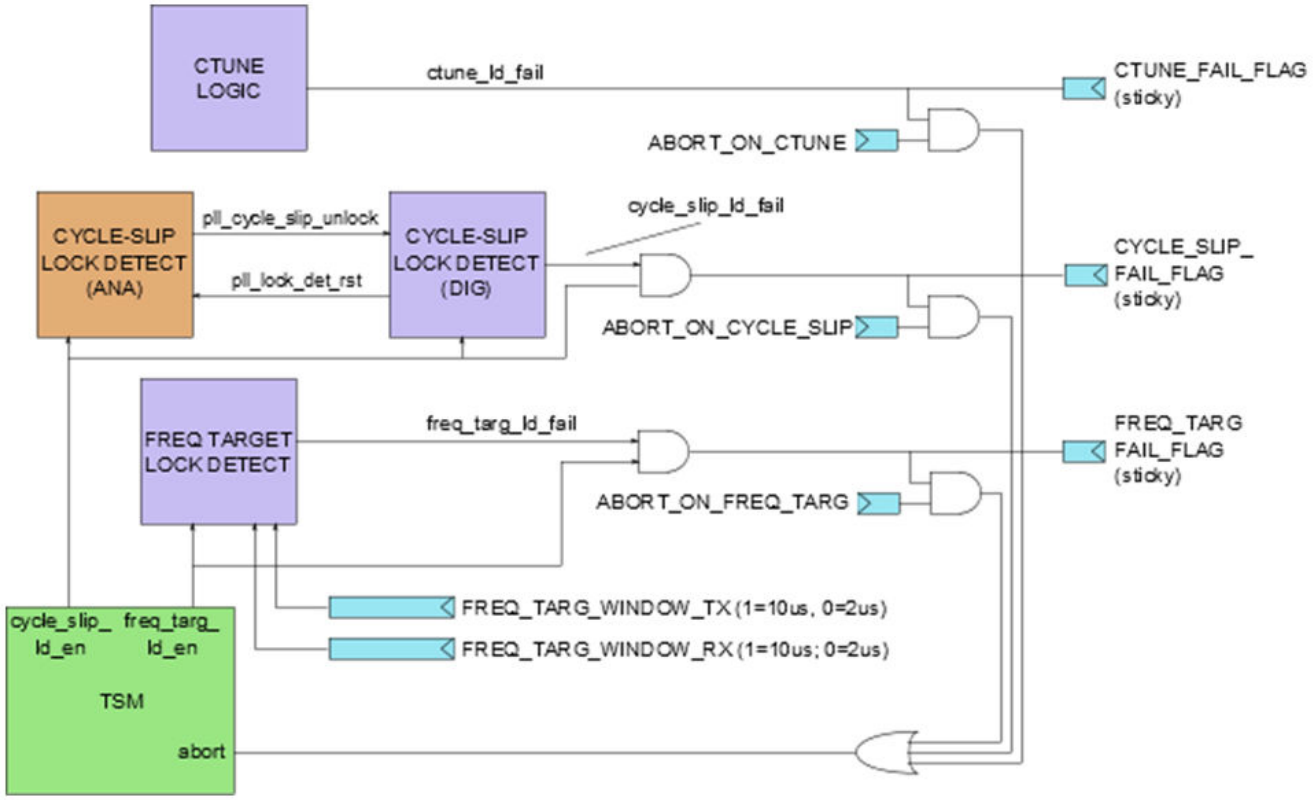



**Figure 64-28. PLL Lock Detect Block Diagram**

### 64.3.2.6.1 Coarse Tune Fail

During every Radio start-up sequence the PLL does a Coarse Tune Calibration to select a VCO setting that will allow the PLL to correctly tune and lock the VCO when the PLL/VCO loop is closed.

During the coarse tune calibration a count is maintained that indicates how close the coarse tune is to the targeted Radio Frequency, if this count is outside of the threshold selected by the Coarse Tune Lock Detect Fail Level register (CTUNE_LDF_LEV), then the Coarse Tune Failure Flag bit (CTFF) will be set and it can be used to abort the TSM sequence.

The CTFF bit is cleared by writing a 1 to it.

The CT_FAIL bit always shows the real time status of the latest Coarse Tune Calibration. If the absolute count difference was out of the range selected by CTUNE_LDF_LEV, then this bit will be set.

#### 64.3.2.6.2 Fine Tune Fail

At the end of every Radio start-up sequence the PLL turns on the PLL Frequency Meter for a time that is requested by the Frequency Target Window time select register bits, separately selected for RX (FTW_RX) and TX (FTW_TX).

These bits select either a 4us or an 8us count period and if the frequency counter is outside of the threshold selected by the Frequency Target Fail Threshold registers, FTF_RX_THRSH and FTF_TX_THRSH, then the Frequency Target Fail Flag bit (FTFF) will be set and it can be used to abort the TSM sequence.

The FTFF bit is cleared by writing a 1 to it

The FT_FAIL bit always shows the real time status of the latest Fine Tune Frequency Measurement. If the absolute count difference was out of the range selected by Frequency Target Fail Threshold registers, then this bit will be set.

#### 64.3.2.6.3 Cycle Slip Fail

During any radio sequence, the PLL can turn on an analog cycle slip detection circuit that measures the phase difference between the VCO and the reference clock. If the phase is not properly aligned the circuit will send a signal to the PLL Lock Detect digital circuit.

The PLL Lock Detect digital circuit evaluates the number and timing of the analog cycle slip events and can generate a Cycle Slip Fail Flag (CSFF).

There is a programmable state machine that controls the monitoring of Cycle Slip Events as shown in the figure below. The CS_FAIL bit always shows the real time status of the cycle slip state machine.

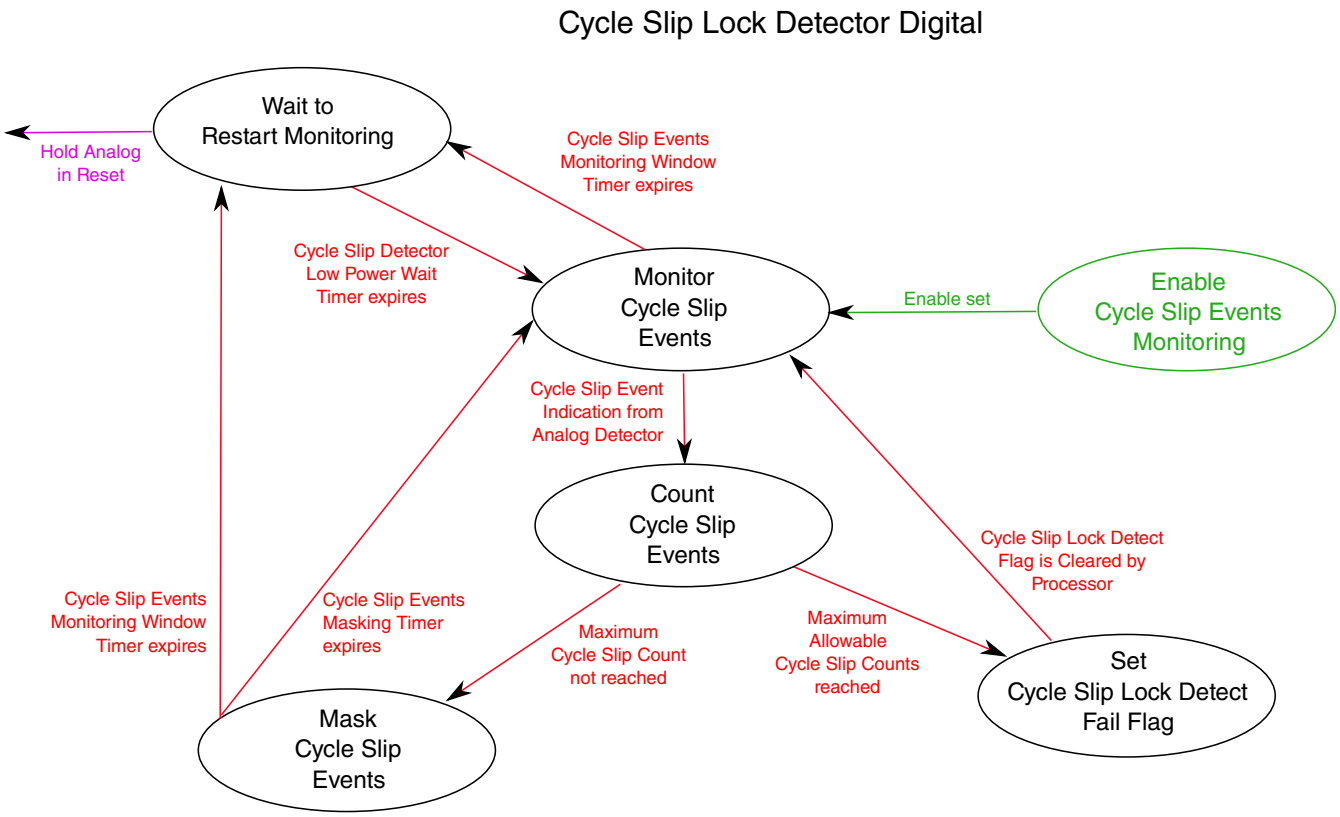



**Figure 64-29. PLL Lock Detect State Machine**

The following table below shows an example calculation of the time needed for the Cycle Slip Lock Detector to fail as a function of the number of cycle slip events and the VCO frequency error.

| Crystal frequency | 32 | MHz | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Reference Time Period | 31.25 | ns | | | | | | | |
| | | | | 3 | 4 | 5 | 8 | 10 | 15 |
| Frequency Error (kHz) | Loop Divider Time Period(ns) | Num cycles in 32MHz Time Period | Cycle-Slip Time (us) | Time for 3 cycle-slips (us) | Time for 4 cycle-slips (us) | Time for 5 cycle-slips (us) | Time for 8 cycle-slips (us) | Time for 10 cycle-slips (us) | Time for 15 cycle-slips (us) |
| -5000 | 37.04 | 5.400 | **0.169** | 0.506 | 1.519 | 4.556 | 13.669 | 41.006 | 123.019 |
| -2000 | 33.33 | 15.000 | **0.469** | 1.406 | 4.219 | 12.656 | 37.969 | 113.906 | 341.719 |
| -1000 | 32.26 | 31.000 | **0.969** | 2.906 | 8.719 | 26.156 | 78.469 | 235.406 | 706.219 |
| -500 | 31.75 | 63.000 | **1.969** | 5.906 | 17.719 | 53.156 | 159.469 | 478.406 | 1435.219 |
| -200 | 31.45 | 159.000 | **4.969** | 14.906 | 44.719 | 134.156 | 402.469 | 1207.406 | 3622.219 |
| 150 | 31.10 | 214.333 | **6.698** | 20.094 | 60.281 | 180.844 | 542.531 | 1627.594 | 4882.781 |
| -100 | 31.35 | 319.000 | **9.969** | 29.906 | 89.719 | 269.156 | 807.469 | 2422.406 | 7267.219 |
| 0 | 31.25 | | | | | | | | |
| 100 | 31.15 | 321.000 | **10.031** | 30.094 | 90.281 | 270.844 | 812.531 | 2437.594 | 7312.781 |
| 150 | 31.10 | 214.333 | **6.698** | 20.094 | 60.281 | 180.844 | 542.531 | 1627.594 | 4882.781 |
| 200 | 31.06 | 161.000 | **5.031** | 15.094 | 45.281 | 135.844 | 407.531 | 1222.594 | 3667.781 |
| 500 | 30.77 | 65.000 | **2.031** | 6.094 | 18.281 | 54.844 | 164.531 | 493.594 | 1480.781 |
| 1000 | 30.30 | 33.000 | **1.031** | 3.094 | 9.281 | 27.844 | 83.531 | 250.594 | 751.781 |
| 2000 | 29.41 | 17.000 | **0.531** | 1.594 | 4.781 | 14.344 | 43.031 | 129.094 | 387.281 |
| 5000 | 27.03 | 7.400 | **0.231** | 0.694 | 2.081 | 6.244 | 18.731 | 56.194 | 168.581 |
| 10000 | 23.81 | 4.200 | **0.131** | 0.394 | 1.181 | 3.544 | 10.631 | 31.894 | 95.681 |
| 20000 | 19.23 | 2.600 | **0.081** | 0.244 | 0.731 | 2.194 | 6.581 | 19.744 | 59.231 |
| 50000 | 12.20 | 1.640 | **0.051** | 0.154 | 0.461 | 1.384 | 4.151 | 12.454 | 37.361 |

**Figure 64-30. Cycle Slip Monitoring Window as a function of frequency error and number of cycle slip events**

A PLL Cycle Slip Fail example is shown in the figure below.

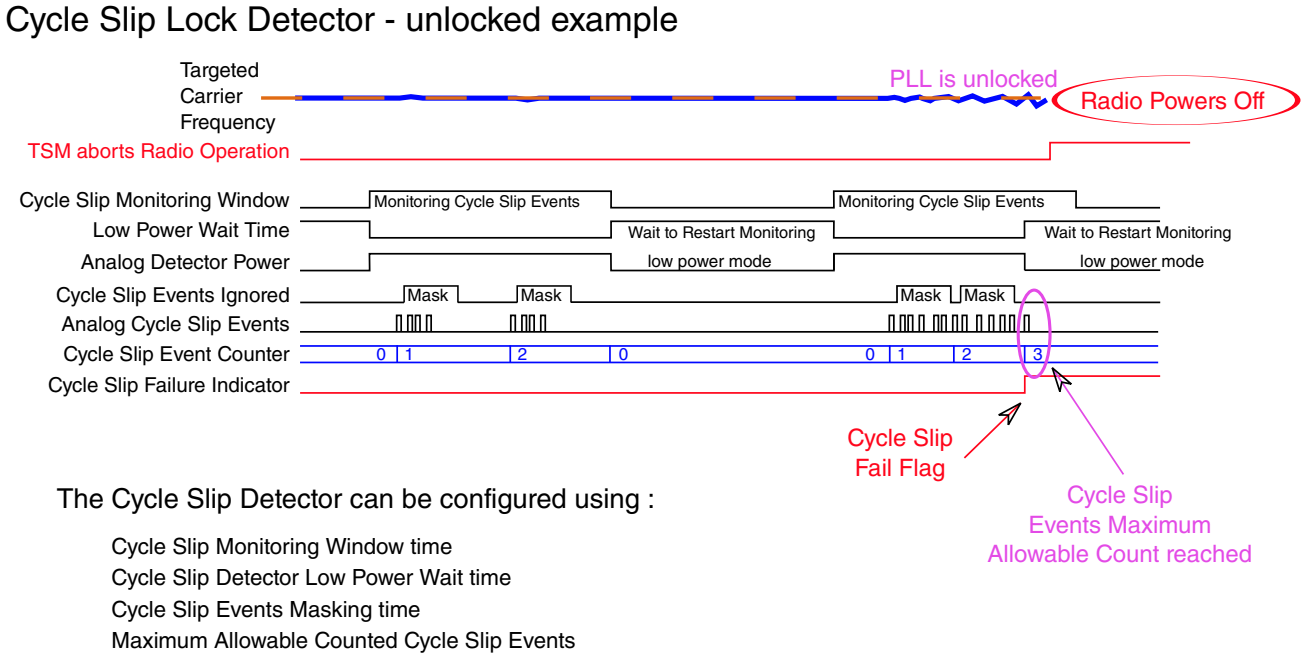



**Figure 64-31. PLL Unlock Example**

### 64.3.2.6.3.1 Cycle Slip Events Monitor Window

The Cycle Slip Events Monitoring Window Timer selects the time in microseconds that the Cycle Slip Lock Detector will monitor Cycle Slip Events.

If the Cycle Slip Events Monitoring Window Timer expires before the Cycle Slip Events Count is reached, then the Events Counter will be reset and the Events Counting sequence can restart after the Cycle Slip Detector Low Power Wait Time if the Cycle Slip Recycle bit is set.

The following selections are to be supported for the Cycle Slip Events Monitoring Window Time:

**Table 64-18. Cycle Slip Events Monitoring Window Times**

| CS_FW[2:0], Cycle Slip Monitor Time | Monitor Time (us) |
|---|---|
| 0 | 8 |
| 1 | 16 |
| 2 | 24 |
| 3 | 32 |
| 4 | 64 |

*Table continues on the next page...*

**Table 64-18. Cycle Slip Events Monitoring Window Times (continued)**

| | |
|---|---|
| 5 | 96 |
| 6 | 128 |
| 7 | 256 |

### 64.3.2.6.3.2 Cycle Slip Detector Low Power Wait Timer

The Cycle Slip Detector Low Power Wait Timer is a method for saving power by turning off the analog cycle slip detector circuit for a programmable period of time. If the Cycle Slip Recycle bit is set when the Cycle Slip Events Monitoring Window Timer expires, then the Events Counter will be reset and the Events Counting sequence will restart after the Cycle Slip Detector Low Power Wait Time expires.

The following selections are to be supported for the Low Power Wait Time:

**Table 64-19. Cycle Slip Detector Low Power Wait Times**

| CS_WT[2:0], Cycle Slip Wait Time | Wait Time (us) |
|---|---|
| 0 | 128 |
| 1 | 256 |
| 2 | 384 |
| 3 | 512 |
| 4 | 640 |
| 5 | 768 |
| 6 | 896 |
| 7 | 1024 |

### 64.3.2.6.3.3 Cycle Slip Events Masking Timer

The Cycle Slip Events Masking Timer selects the number of nanoseconds that the Cycle Slip Lock Detector will wait before counting the next Cycle Slip Event. This allows time for the PLL to try to regain lock on the VCO.

The following selections are to be supported for the Cycle Slip Events Masking time after an analog cycle slip event:

**Table 64-20. Cycle Slip Events Masking Times**

| CS_FT[4:0], Cycle Slips Masking Time | Wait Time (ns), 26 MHz | Wait Time (ns), 32 MHz |
|---|---|---|
| 0 | 38 | 31 |

*Table continues on the next page...*

**Table 64-20. Cycle Slip Events Masking Times (continued)**

| | | |
|---|---|---|
| 1 | 77 | 63 |
| 2 | 115 | 94 |
| 3 | 154 | 125 |
| 4 | 192 | 156 |
| 5 | 231 | 188 |
| 6 | 269 | 219 |
| 7 | 308 | 250 |
| 8 | 346 | 281 |
| 9 | 385 | 313 |
| 10 | 423 | 344 |
| 11 | 462 | 375 |
| 12 | 500 | 406 |
| 13 | 538 | 438 |
| 14 | 577 | 469 |
| 15 | 615 | 500 |
| 16 | 654 | 531 |
| 17 | 692 | 563 |
| 18 | 731 | 594 |
| 19 | 769 | 625 |
| 20 | 808 | 656 |
| 21 | 846 | 688 |
| 22 | 885 | 719 |
| 23 | 923 | 750 |
| 24 | 962 | 781 |
| 25 | 1000 | 813 |
| 26 | 1038 | 844 |
| 27 | 1077 | 875 |
| 28 | 1115 | 906 |
| 29 | 1154 | 938 |
| 30 | 1192 | 969 |
| 31 | 1231 | 1000 |

#### 64.3.2.6.3.4 Cycle Slip Events Count

The Cycle Slip Events Count (CS_FCNT) determines the number of Cycle Slip Events that will cause a Cycle Slip Fail Flag to be set.

The range of Cycle Slip Events Counted that will cause a Fail Flag shall be from 1 to 15.

## 64.3.2.7 Memory Map and Register Definition

The PLL Digital memory map and detailed descriptions of all its registers are as follows.

### 64.3.2.7.1 XCVR_PLL_DIG register descriptions

#### 64.3.2.7.1.1 XCVR_PLL_DIG_ADDR Memory map

XCVR_PLL_DIG base address: 4103_0224h

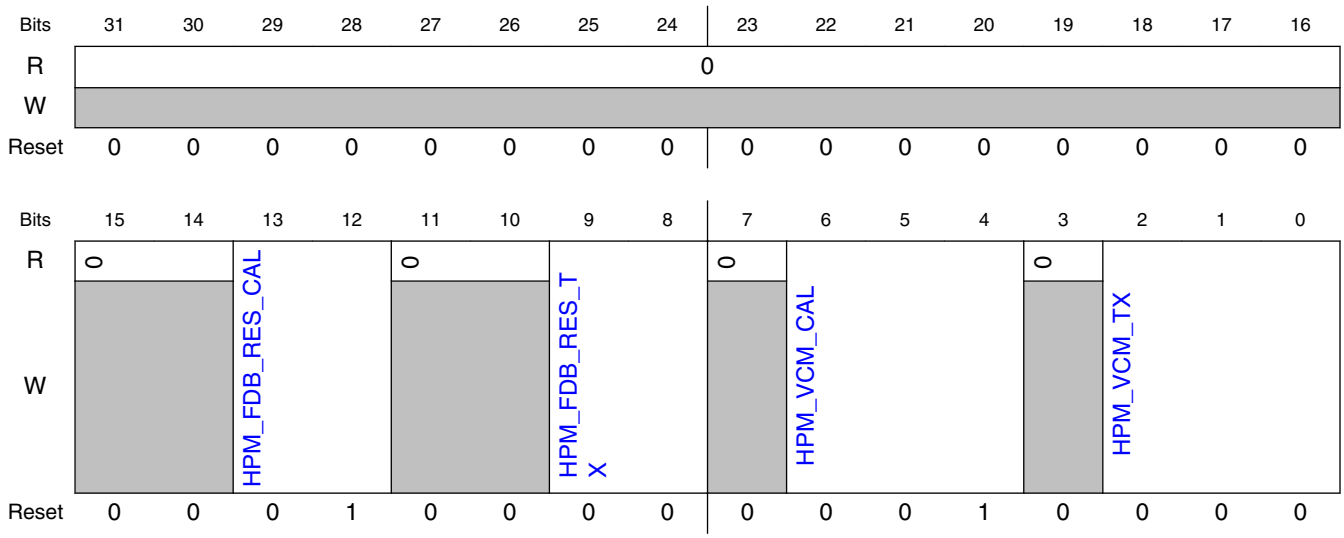
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | PLL HPM Analog Bump Control (HPM_BUMP) | 32 | RW | 0000_1010h |
| 4h | PLL Modulation Control (MOD_CTRL) | 32 | RW | 0000_0000h |
| 8h | PLL Channel Mapping (CHAN_MAP) | 32 | RW | 0000_0200h |
| Ch | PLL Lock Detect Control (LOCK_DETECT) | 32 | RW | 0060_6800h |
| 10h | PLL High Port Modulator Control (HPM_CTRL) | 32 | RW | 9084_0000h |
| 20h | PLL High Port Sigma Delta Results (HPM_SDM_RES) | 32 | RW | 0100_0000h |
| 24h | PLL Low Port Modulator Control (LPM_CTRL) | 32 | RW | 0202_0000h |
| 28h | PLL Low Port Sigma Delta Control 1 (LPM_SDM_CTRL1) | 32 | RW | 0026_0026h |
| 2Ch | PLL Low Port Sigma Delta Control 2 (LPM_SDM_CTRL2) | 32 | RW | 0008_0000h |
| 30h | PLL Low Port Sigma Delta Control 3 (LPM_SDM_CTRL3) | 32 | RW | 0010_0000h |
| 34h | PLL Low Port Sigma Delta Result 1 (LPM_SDM_RES1) | 32 | RO | 0000_0000h |
| 38h | PLL Low Port Sigma Delta Result 2 (LPM_SDM_RES2) | 32 | RO | 0400_0000h |
| 3Ch | PLL Delay Matching (DELAY_MATCH) | 32 | RW | 0000_0004h |
| 40h | PLL Coarse Tune Control (CTUNE_CTRL) | 32 | RW | 0000_0000h |
| 54h | PLL Coarse Tune Results (CTUNE_RES) | 32 | RO | 0962_0040h |

#### 64.3.2.7.1.2 PLL HPM Analog Bump Control (HPM_BUMP)

##### *64.3.2.7.1.2.1 Offset*

| Register | Offset |
|---|---|
| HPM_BUMP | 0h |

### 64.3.2.7.1.2.2 Diagram




### 64.3.2.7.1.2.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-12<br>HPM_FDB_RES_CAL | rfctrl_tx_dac_bump_fdb_res[1:0] during Calibration<br>This is the value applied to the rfctrl_tx_dac_bump_fdb_res[1:0] port during High Port Calibration.<br>This register sets the HPM DAC feedback resistor to increase the modulation gain during calibration.<br>00b - 29 kohms<br>01b - 58 kohms(gain of 2)<br>10b - 13 kohms<br>11b - 23.7 kohms |
| 11-10<br>— | Reserved |
| 9-8<br>HPM_FDB_RES_TX | rfctrl_tx_dac_bump_fdb_res[1:0] during Transmission<br>This is the value applied to the rfctrl_tx_dac_bump_fdb_res[1:0] port during Radio Transmissions.<br>This register sets the HPM DAC feedback resistor during transmissions.<br>00b - 29 kohms<br>01b - 58 kohms(gain of 2)<br>10b - 13 kohms<br>11b - 23.7 kohms |
| 7<br>— | Reserved |
| 6-4<br>HPM_VCM_CAL | rfctrl_tx_dac_bump_vcm[2:0] during Calibration<br>This is the value applied to the rfctrl_tx_dac_bump_vcm[2:0] port during High Port Calibration.<br>This register sets the HPM DAC Op-Amp reference voltage during calibration. |

*Table continues on the next page...*

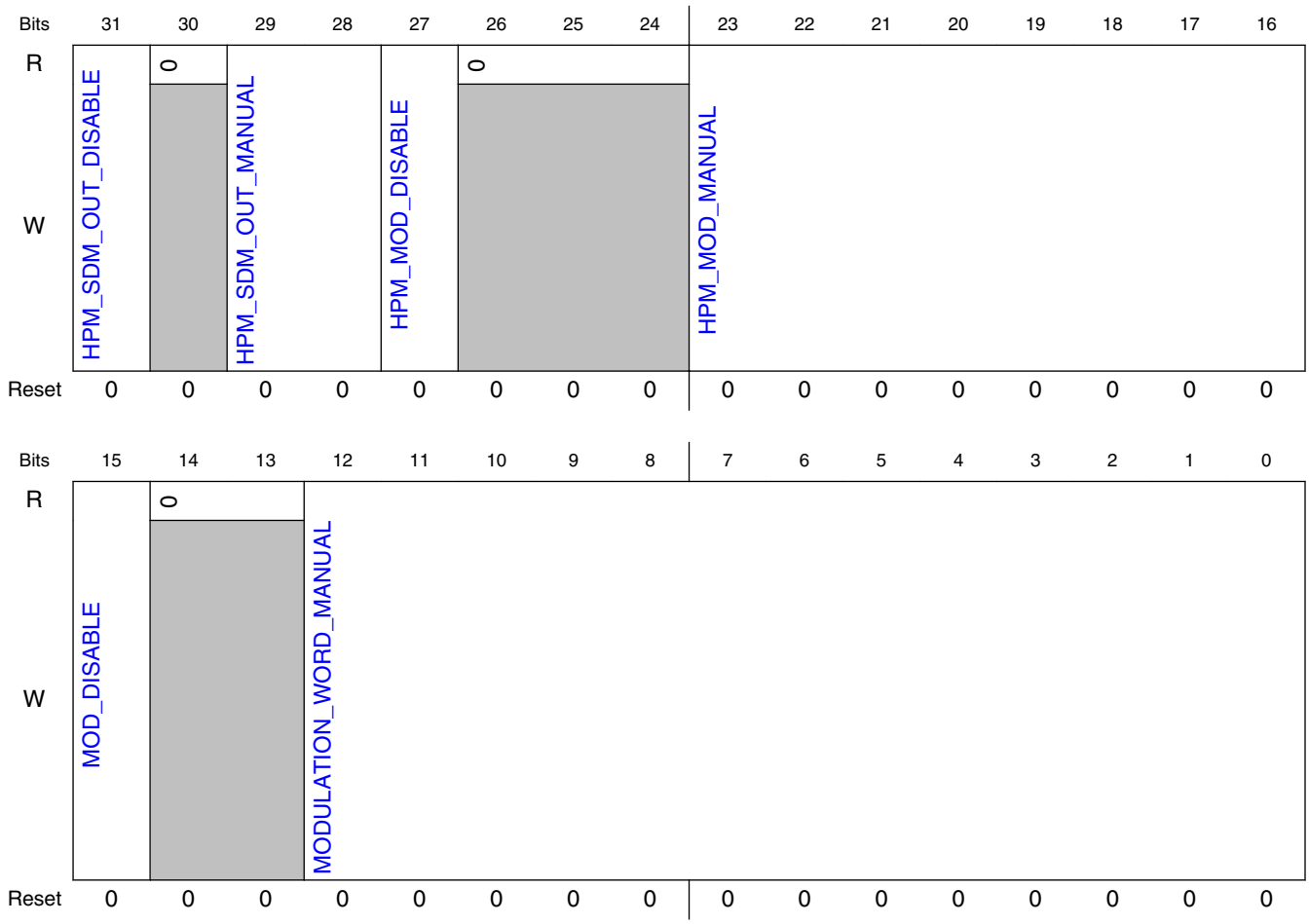
| Field | Function |
|---|---|
| | 000b - 432 mV<br>001b - 328 mV<br>010b - 456 mV<br>011b - 473 mV<br>100b - 488 mV<br>101b - 408 mV<br>110b - 392 mV<br>111b - 376 mV |
| 3<br>— | Reserved |
| 2-0<br>HPM_VCM_TX | rfctrl_tx_dac_bump_vcm[2:0] during Transmission<br>This is the value applied to the rfctrl_tx_dac_bump_vcm[2:0] port during Radio Transmissions.<br>This register sets the HPM DAC Op-Amp reference voltage during transmissions.<br>000b - 432 mV<br>001b - 328 mV<br>010b - 456 mV<br>011b - 473 mV<br>100b - 488 mV<br>101b - 408 mV<br>110b - 392 mV<br>111b - 376 mV |

## 64.3.2.7.1.3 PLL Modulation Control (MOD_CTRL)

### *64.3.2.7.1.3.1 Offset*

| Register | Offset |
|---|---|
| MOD_CTRL | 4h |

### 64.3.2.7.1.3.2 Diagram




### 64.3.2.7.1.3.3 Fields

| Field | Function |
|---|---|
| 31<br>HPM_SDM_OUT_DISABLE | Disable HPM SDM out<br>If this bit is set, the High Port Sigma Delta Modulator output is disabled, and the High Port Fractional value applied to the VCO comes from the HPM_SDM_OUT_MANUAL register. |
| 30<br>— | Reserved |
| 29-28<br>HPM_SDM_OUT_MANUAL | Manual HPM SDM out<br>If HPM_SDM_OUT_DISABLE is set, this register is the Fractional value that is applied to the VCO High Port. |
| 27<br>HPM_MOD_DISABLE | Disable HPM Modulation<br>If this bit is set, the High Port Modulation is disabled, and the High Port Modulation value applied to the VCO comes from the HPM_MOD_MANUAL register. |
| 26-24<br>— | Reserved |

*Table continues on the next page...*

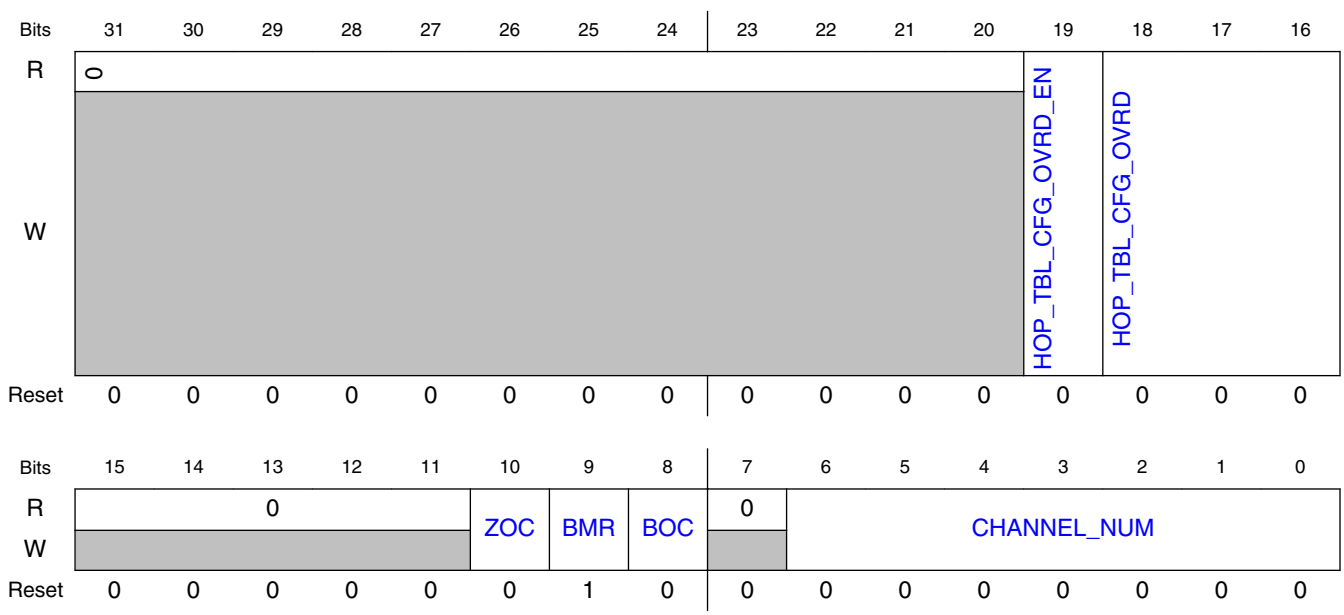
| Field | Function |
|---|---|
| 23-16<br>HPM_MOD_MANUAL | Manual HPM Modulation<br>If HPM_MOD_DISABLE is set, this register is the modulation value that is applied to the VCO High Port. |
| 15<br>MOD_DISABLE | Disable Modulation Word<br>If this bit is set, any modulation from the TX Digital is disabled and the source of the Baseband Frequency Word is the MODULATION_WORD_MANUAL register. |
| 14-13<br>— | Reserved |
| 12-0<br>MODULATION_WORD_MANUAL | Manual Modulation Word<br>If MOD_DISABLE is set, the signed 12 bit value that is represented by this register is the Baseband Frequency Word. |

## 64.3.2.7.1.4 PLL Channel Mapping (CHAN_MAP)

### *64.3.2.7.1.4.1 Offset*

| Register | Offset |
|---|---|
| CHAN_MAP | 8h |

### *64.3.2.7.1.4.2 Diagram*



### 64.3.2.7.1.4.3 Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>HOP_TBL_CFG_OVRD_EN | Hop Table Configuration Override Enable<br>If this bit is set then the HOP_TBL_CFG_OVRD and DFT_PATTERN Registers will be used to control the Coarse Tune, Channel Number, and Hop Table for the PLL Carrier Frequency Selection. |
| 18-16<br>HOP_TBL_CFG_OVRD | Hop Table Configuration Override<br>If the HOP_TBL_CFG_OVRD_EN bit is set, then the DFT_PATTERN Register will be used to control the Hop Table for the PLL Carrier Frequency Selection.<br>010b - DFT_PATTERN[15:7] is signed offset to DFT_PATTERN[6:0] mapped channel number<br>011b - DFT_PATTERN[15:1] is signed Numerator, DFT_PATTERN[0] is integer selection |
| 15-11<br>— | Reserved |
| 10<br>ZOC | 802.15.4 Channel Number Override<br>This bit controls the source of the 802.15.4 channel selection.<br>0b - 802.15.4 channel number comes from the 802.15.4 Link Layer.<br>1b - 802.15.4 channel number comes from the CHANNEL_NUM register (802.15.4 protocols 4 and 5) |
| 9<br>BMR | BLE MBAN Channel Remap<br>This bit controls the mapping of BLE channel 39 in Radio Protocol 2, BLE overlap MBAN mode.<br>0b - BLE channel 39 is mapped to BLE channel 39, 2.480 GHz<br>1b - BLE channel 39 is mapped to MBAN channel 39, 2.399 GHz |
| 8<br>BOC | BLE Channel Number Override<br>This bit controls the source of the BLE channel selection.<br>0b - BLE channel number comes from the BLE Link Layer<br>1b - BLE channel number comes from the CHANNEL_NUM register (BLE protocols 0 and 2) |
| 7<br>— | Reserved |
| 6-0<br>CHANNEL_NUM | Protocol specific Channel Number for PLL Frequency Mapping<br>When this register is active, it can be used to directly select a Protocol specific Channel Number, which is mapped internally to the correct Radio Carrier Frequency for PLL tuning. The internal mapping is detailed in the table below.<br>This register is active when BOC or ZOC are set along with their corresponding Radio Protocols, and this register is also active in protocols 1, 6 and 7.<br>The Radio Channel Frequency can also be selected by setting the SDM_MAP_DISABLE bit in the PLL_LPM_SDM_CTRL1 register along with the LPM_INTG, LPM_NUM, and LPM_DENOM registers to get a frequency that equals ((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM))<br>This table shows the internal mapping by Protocol of the Channel Numbers to the Radio Carrier Frequency (MHz). |

| Radio Protocols Supported, Channel Frequency in MHz | | | | |
|---|---|---|---|---|
| BLE | MBAN | BLE Overlap MBAN | 802.15.4 | 802.15.4j MBAN |
| 0 | 1 | 2 | 3 | 4 |

| Field | Function | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz |
| | 0 | 2,402 | 0 | 2,360 | 0 | 2,402 | 11 | 2,405 | 0 | 2,363 |
| | 1 | 2,404 | 1 | 2,361 | 1 | 2,404 | 12 | 2,410 | 1 | 2,368 |
| | 2 | 2,406 | 2 | 2,362 | 2 | 2,406 | 13 | 2,415 | 2 | 2,373 |
| | 3 | 2,408 | 3 | 2,363 | 3 | 2,408 | 14 | 2,420 | 3 | 2,378 |
| | 4 | 2,410 | 4 | 2,364 | 4 | 2,410 | 15 | 2,425 | 4 | 2,383 |
| | 5 | 2,412 | 5 | 2,365 | 5 | 2,412 | 16 | 2,430 | 5 | 2,388 |
| | 6 | 2,414 | 6 | 2,366 | 6 | 2,414 | 17 | 2,435 | 6 | 2,393 |
| | 7 | 2,416 | 7 | 2,367 | 7 | 2,416 | 18 | 2,440 | 7 | 2,367 |
| | 8 | 2,418 | 8 | 2,368 | 8 | 2,418 | 19 | 2,445 | 8 | 2,372 |
| | 9 | 2,420 | 9 | 2,369 | 9 | 2,420 | 20 | 2,450 | 9 | 2,377 |
| | 10 | 2,422 | 10 | 2,370 | 10 | 2,422 | 21 | 2,455 | 10 | 2,382 |
| | 11 | 2,424 | 11 | 2,371 | 11 | 2,424 | 22 | 2,460 | 11 | 2,387 |
| | 12 | 2,426 | 12 | 2,372 | 12 | 2,426 | 23 | 2,465 | 12 | 2,392 |
| | 13 | 2,428 | 13 | 2,373 | 13 | 2,428 | 24 | 2,470 | 13 | 2,397 |
| | 14 | 2,430 | 14 | 2,374 | 14 | 2,430 | 25 | 2,475 | 14 | 2,395 |
| | 15 | 2,432 | 15 | 2,375 | 15 | 2,432 | 26 | 2,480 | | |
| | 16 | 2,434 | 16 | 2,376 | 16 | 2,434 | | | | |
| | 17 | 2,436 | 17 | 2,377 | 17 | 2,436 | | | | |
| | 18 | 2,438 | 18 | 2,378 | 18 | 2,438 | | | | |
| | 19 | 2,440 | 19 | 2,379 | 19 | 2,440 | | | | |
| | 20 | 2,442 | 20 | 2,380 | 20 | 2,442 | | | | |
| | 21 | 2,444 | 21 | 2,381 | 21 | 2,444 | | | | |
| | 22 | 2,446 | 22 | 2,382 | 22 | 2,446 | | | | |
| | 23 | 2,448 | 23 | 2,383 | 23 | 2,448 | | | | |
| | 24 | 2,450 | 24 | 2,384 | 24 | 2,450 | | | | |
| | 25 | 2,452 | 25 | 2,385 | 25 | 2,452 | | | | |
| | 26 | 2,454 | 26 | 2,386 | 26 | 2,454 | | | | |
| | 27 | 2,456 | 27 | 2,387 | 27 | 2,456 | | | | |
| | 28 | 2,458 | 28 | 2,388 | 28 | 2,458 | | | | |
| | 29 | 2,460 | 29 | 2,389 | 29 | 2,460 | | | | |
| | 30 | 2,462 | 30 | 2,390 | 30 | 2,390 | | | | |
| | 31 | 2,464 | 31 | 2,391 | 31 | 2,391 | | | | |
| | 32 | 2,466 | 32 | 2,392 | 32 | 2,392 | | | | |
| | 33 | 2,468 | 33 | 2,393 | 33 | 2,393 | | | | |
| | 34 | 2,470 | 34 | 2,394 | 34 | 2,394 | | | | |
| | 35 | 2,472 | 35 | 2,395 | 35 | 2,395 | | | | |
| | 36 | 2,474 | 36 | 2,396 | 36 | 2,396 | | | | |
| | 37 | 2,476 | 37 | 2,397 | 37 | 2,397 | | | | |
| | 38 | 2,478 | 38 | 2,398 | 38 | 2,398 | | | | |
| | 39 | 2,480 | 39 | 2,399 | 39* | 2,480 | | | | |

| Field | Function |
|---|---|

* The BLE MBAN Channel Remap bit, BMR, controls the frequency mapping in this case.

| Radio Protocols Supported, Channel Frequency in MHz | | | | | |
|---|---|---|---|---|---|
| DFT, Generic | | | | | |
| 6,7,8,9 | | | | | |
| Channel | MHz | Channel | MHz | Channel | MHz |
| 0 | 2,360 | 43 | 2,403 | 86 | 2,446 |
| 1 | 2,361 | 44 | 2,404 | 87 | 2,447 |
| 2 | 2,362 | 45 | 2,405 | 88 | 2,448 |
| 3 | 2,363 | 46 | 2,406 | 89 | 2,449 |
| 4 | 2,364 | 47 | 2,407 | 90 | 2,450 |
| 5 | 2,365 | 48 | 2,408 | 91 | 2,451 |
| 6 | 2,366 | 49 | 2,409 | 92 | 2,452 |
| 7 | 2,367 | 50 | 2,410 | 93 | 2,453 |
| 8 | 2,368 | 51 | 2,411 | 94 | 2,454 |
| 9 | 2,369 | 52 | 2,412 | 95 | 2,455 |
| 10 | 2,370 | 53 | 2,413 | 96 | 2,456 |
| 11 | 2,371 | 54 | 2,414 | 97 | 2,457 |
| 12 | 2,372 | 55 | 2,415 | 98 | 2,458 |
| 13 | 2,373 | 56 | 2,416 | 99 | 2,459 |
| 14 | 2,374 | 57 | 2,417 | 100 | 2,460 |
| 15 | 2,375 | 58 | 2,418 | 101 | 2,461 |
| 16 | 2,376 | 59 | 2,419 | 102 | 2,462 |
| 17 | 2,377 | 60 | 2,420 | 103 | 2,463 |
| 18 | 2,378 | 61 | 2,421 | 104 | 2,464 |
| 19 | 2,379 | 62 | 2,422 | 105 | 2,465 |
| 20 | 2,380 | 63 | 2,423 | 106 | 2,466 |
| 21 | 2,381 | 64 | 2,424 | 107 | 2,467 |
| 22 | 2,382 | 65 | 2,425 | 108 | 2,468 |
| 23 | 2,383 | 66 | 2,426 | 109 | 2,469 |
| 24 | 2,384 | 67 | 2,427 | 110 | 2,470 |
| 25 | 2,385 | 68 | 2,428 | 111 | 2,471 |
| 26 | 2,386 | 69 | 2,429 | 112 | 2,472 |
| 27 | 2,387 | 70 | 2,430 | 113 | 2,473 |
| 28 | 2,388 | 71 | 2,431 | 114 | 2,474 |
| 29 | 2,389 | 72 | 2,432 | 115 | 2,475 |
| 30 | 2,390 | 73 | 2,433 | 116 | 2,476 |
| 31 | 2,391 | 74 | 2,434 | 117 | 2,477 |
| 32 | 2,392 | 75 | 2,435 | 118 | 2,478 |
| 33 | 2,393 | 76 | 2,436 | 119 | 2,479 |

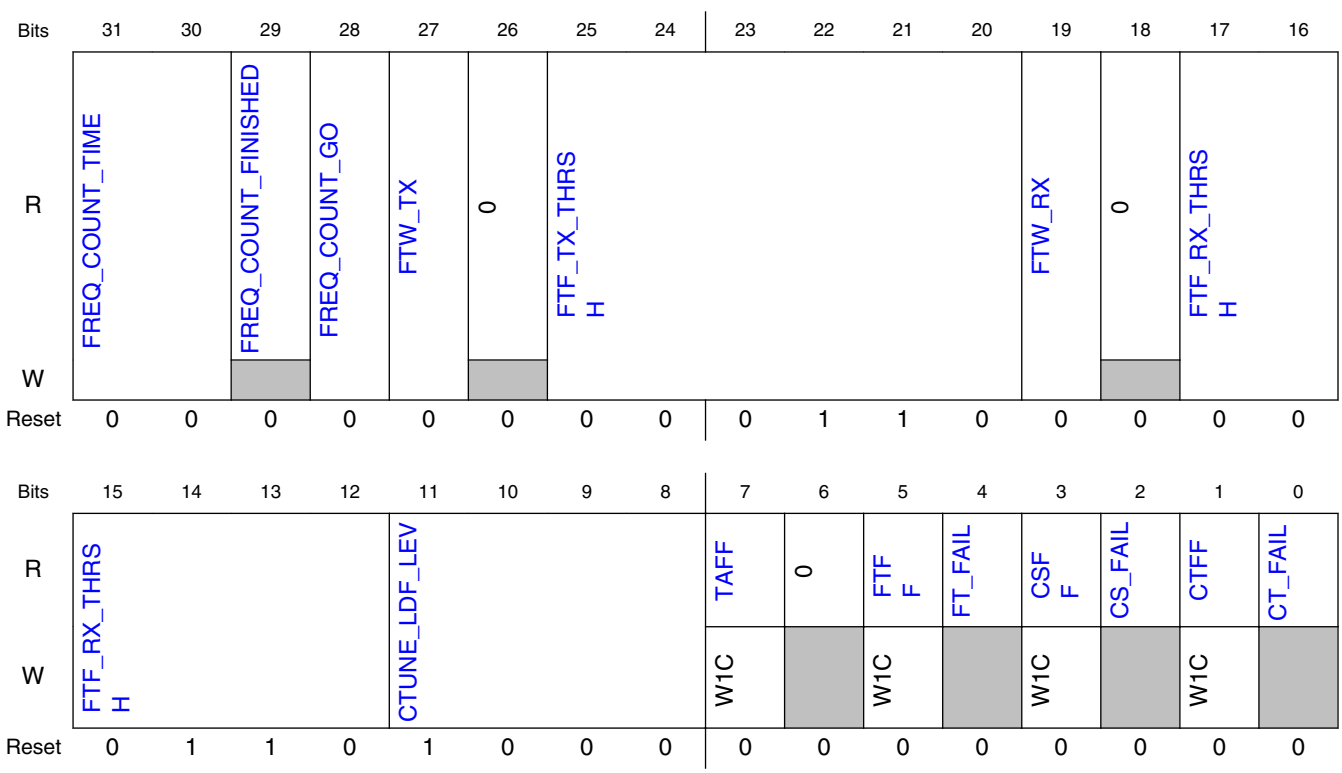
| Field | Function | | | | | |
|---|---|---|---|---|---|---|
| | 34 | 2,394 | 77 | 2,437 | 120 | 2,480 |
| | 35 | 2,395 | 78 | 2,438 | 121 | 2,481 |
| | 36 | 2,396 | 79 | 2,439 | 122 | 2,482 |
| | 37 | 2,397 | 80 | 2,440 | 123 | 2,483 |
| | 38 | 2,398 | 81 | 2,441 | 124 | 2,484 |
| | 39 | 2,399 | 82 | 2,442 | 125 | 2,485 |
| | 40 | 2,400 | 83 | 2,443 | 126 | 2,486 |
| | 41 | 2,401 | 84 | 2,444 | 127 | 2,487 |
| | 42 | 2,402 | 85 | 2,445 | | |

## 64.3.2.7.1.5 PLL Lock Detect Control (LOCK_DETECT)

### *64.3.2.7.1.5.1 Offset*

| Register | Offset |
|---|---|
| LOCK_DETECT | Ch |

### *64.3.2.7.1.5.2 Diagram*



### *64.3.2.7.1.5.3 Fields*

| Field | Function |
|---|---|
| 31-30<br>FREQ_COUNT_TIME | Frequency Meter Count Time<br>This is the length of time that the Frequency Meter will count VCO clocks.<br>00b - 800 us<br>01b - 25 us<br>10b - 50 us<br>11b - 100 us |
| 29<br>FREQ_COUNT_FINISHED | Frequency Meter has finished the Count Time<br>This bit is set when the FREQ_COUNT_TIME value is reached, it is cleared when FREQ_COUNT_GO is cleared. |
| 28<br>FREQ_COUNT_GO | Start the Frequency Meter<br>The Frequency Meter starts when this bit is set and runs until the FREQ_COUNT_TIME value is reached. The bit should not be cleared until after the counting is finished.<br>After the Counting Time finishes, the FREQ_COUNT_FINISHED bit will read as a one. Then the measured frequency can be calculated by reading the frequency counts in the DFT_FREQ_COUNTER register (in the XCVR_ANALOG register space) and dividing by the FREQ_COUNT_TIME.<br>Note that the counting is done at the VCO frequency which is 2X the Carrier Frequency |
| 27<br>FTW_TX | TX Frequency Target Window time select<br>In Radio Transmit Mode, this bit selects the length of time to count for the estimation of PLL lock frequency, which is compared with the Frequency Target Window, set by FTF_TX.<br>0b - 4 us<br>1b - 8 us |
| 26<br>— | Reserved |
| 25-20<br>FTF_TX_THRSH | TX Frequency Target Fail Threshold<br>In Radio Transmit Mode, the FT_FAIL and FTFF bits will be set after the Frequency Target Count completes if the absolute value of the count difference from the frequency target count is greater than this register value. |
| 19<br>FTW_RX | RX Frequency Target Window time select<br>In Radio Receive Mode, this bit selects the length of time to count for the estimation of PLL lock frequency, which is compared with the Frequency Target Window, set by FTF_RX.<br>0b - 4 us<br>1b - 8 us |
| 18<br>— | Reserved |
| 17-12<br>FTF_RX_THRSH | RX Frequency Target Fail Threshold<br>In Radio Receive Mode, the FT_FAIL and FTFF bits will be set after the Frequency Target Count completes if the absolute value of the count difference from the frequency target count is greater than this register value. |
| 11-8<br>CTUNE_LDF_LEV | CTUNE Lock Detect Fail Level<br>The CT_FAIL and CTFF bits will be set after Coarse Tune Calibration completes if the absolute value of the Coarse Tune best count difference (CTUNE_BEST_DIFF in the PLL_CTUNE_RESULTS register) is greater than this register value. |

*Table continues on the next page...*

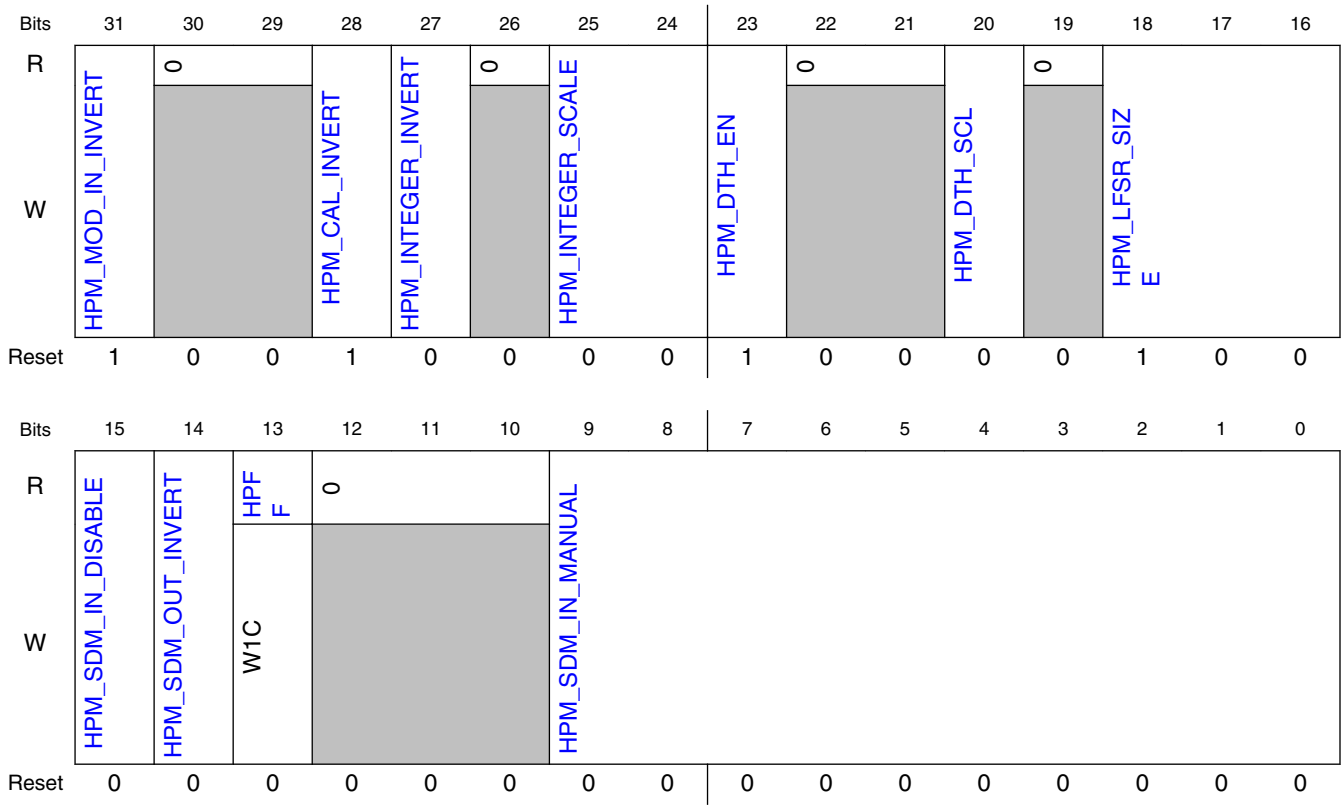
| Field | Function |
|---|---|
| 7<br>TAFF | TSM Abort Failure Flag<br>This bit is set if the TSM Sequence Aborts, and this bit is cleared by writing a 1 to it. |
| 6<br>— | Reserved |
| 5<br>FTFF | Frequency Target Failure Flag<br>This bit is set when FT_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 4<br>FT_FAIL | Real time status of Frequency Target Failure<br>If the Frequency Target Count has completed and the count was out of the range selected by FTW_TX or FTW_RX, then this bit will be set. |
| 3<br>CSFF | Cycle Slip Failure Flag, held until cleared<br>This bit is set when CS_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 2<br>CS_FAIL | Real time status of Cycle Slip circuit<br>This bit shows the real-time status of the Cycle Slip State Machine Interrupt which is configured using the control bits in the PLL_HPM_CAL1 and PLL_HPM_CAL2 registers. |
| 1<br>CTFF | CTUNE Failure Flag, held until cleared<br>This bit is set when CT_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 0<br>CT_FAIL | Real time status of Coarse Tune Fail signal<br>If the Coarse Tune Calibration has completed and the best count difference is out of the range selected by CTUNE_LDF_LEV, then this bit will be set. |

## 64.3.2.7.1.6 PLL High Port Modulator Control (HPM_CTRL)

### *64.3.2.7.1.6.1 Offset*

| Register | Offset |
|---|---|
| HPM_CTRL | 10h |

### 64.3.2.7.1.6.2 Diagram




### 64.3.2.7.1.6.3 Fields

| Field | Function |
|---|---|
| 31<br>HPM_MOD_IN_INVERT | Invert High Port Modulation<br>If this bit is set then the High Port Modulation Word is inverted before it is multiplied and split into Integer and Fractional values. |
| 30-29<br>— | Reserved |
| 28<br>HPM_CAL_INVERT | Invert High Port Modulator Calibration<br>If this bit is set then the order of the High Port Calibration is reversed in order to get a positive count difference between the Maximum and Minimum settings of the HPM DAC. |
| 27<br>HPM_INTEGER_INVERT | Invert High Port Modulation Integer<br>If this bit is set then the High Port Modulation Integer value, after the multiply and split into Integer and Fractional values, will be inverted before it is applied to the VCO High Port DAC Array. |
| 26<br>— | Reserved |
| 25-24<br>HPM_INTEGER_SCALE | High Port Modulation Integer Scale<br>This register controls the scaling of the High Port Modulation Integer Value applied to the VCO High Port DAC Array.<br>00b - No Scaling |

*Table continues on the next page...*

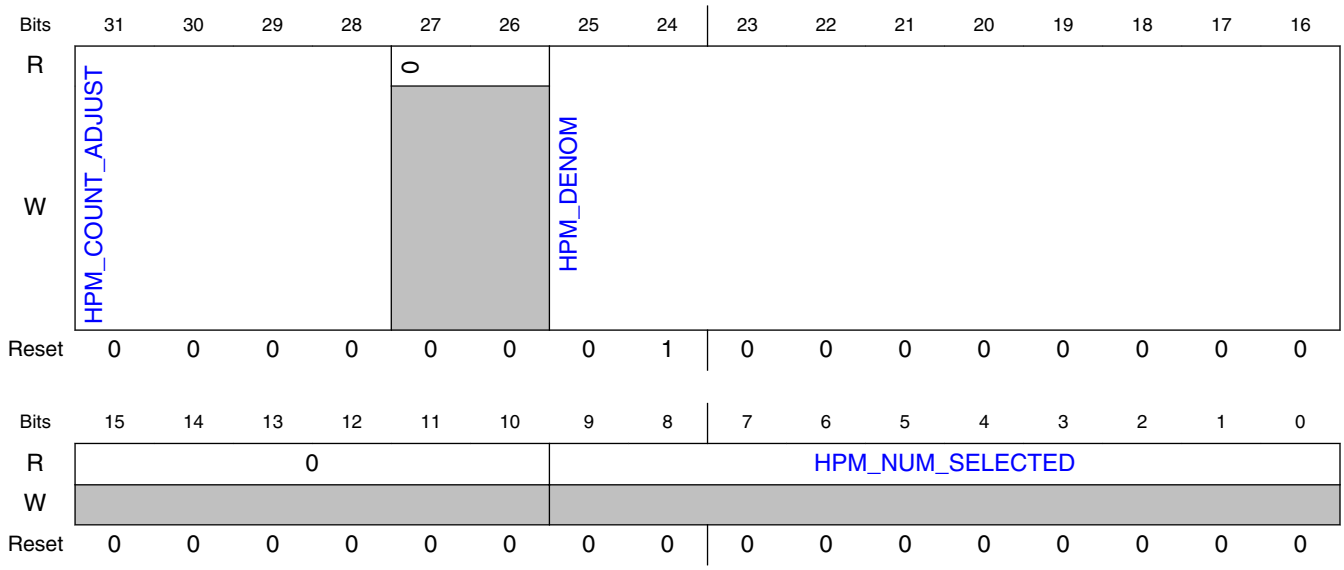
| Field | Function |
|---|---|
| | 01b - Multiply by 2<br>10b - Divide by 2<br>11b - Reserved |
| 23<br>HPM_DTH_EN | Dither Enable for HPM LFSR<br>If this bit is set, the High Port Fraction will be Dithered by the High Port LFSR before it is applied to the High Port SDM. |
| 22-21<br>— | Reserved |
| 20<br>HPM_DTH_SCL | HPM Dither Scale<br>If this bit is set, the LFSR dithering of the High Port Fraction will be multiplied by 2. |
| 19<br>— | Reserved |
| 18-16<br>HPM_LFSR_SIZE | HPM LFSR Length<br>This register selects the length of the HPM LFSR and the associated LFSR Tap Mask<br>000b - LFSR 9, tap mask 100010000<br>001b - LFSR 10, tap mask 1001000000<br>010b - LFSR 11, tap mask 11101000000<br>011b - LFSR 13, tap mask 1101100000000<br>100b - LFSR 15, tap mask 111010000000000<br>101b - LFSR 17, tap mask 111100000000000000<br>110b - Reserved<br>111b - Reserved |
| 15<br>HPM_SDM_IN_DISABLE | Disable HPM SDM Input<br>If this bit is set, the Fractional portion of the High Port Modulation to the High Port SDM is disabled, and the High Port SDM input comes from the HPM_SDM_IN_MANUAL register. |
| 14<br>HPM_SDM_OUT_INVERT | Invert HPM SDM Output<br>If this bit is set the High Port SDM result will be Inverted before it is applied to the VCO High Port DAC Array. |
| 13<br>HPFF | HPM SDM Invalid Flag<br>This bit is set if the High Port Sigma Delta Modulator output is invalid due to an error in the fraction applied, and this bit is cleared by writing a 1 to it. |
| 12-10<br>— | Reserved |
| 9-0<br>HPM_SDM_IN_MANUAL | Manual High Port SDM Fractional value<br>If HPM_SDM_IN_DISABLE is set, this register is the value that is applied as the Fractional value to the input of the High Port SDM. |

## 64.3.2.7.1.7 PLL High Port Sigma Delta Results (HPM_SDM_RES)

### *64.3.2.7.1.7.1 Offset*

| Register | Offset |
|---|---|
| HPM_SDM_RES | 20h |

### 64.3.2.7.1.7.2 Diagram




### 64.3.2.7.1.7.3 Fields

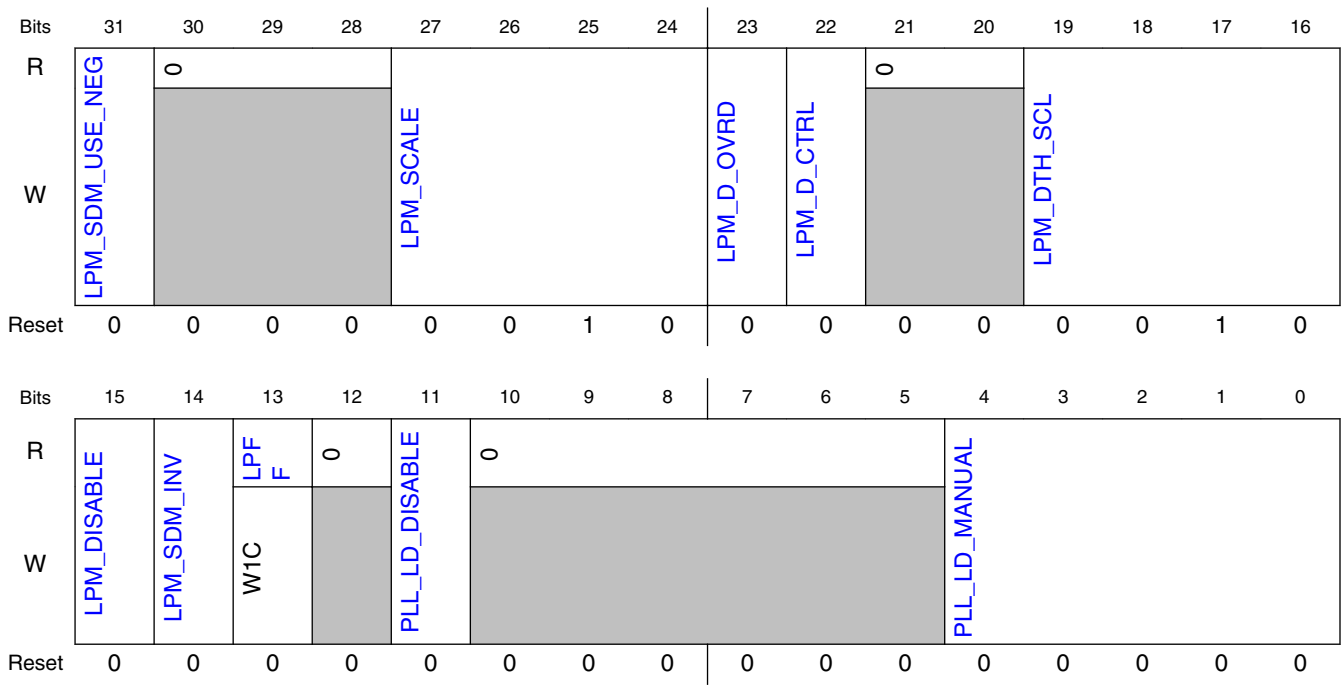
| Field | Function |
|---|---|
| 31-28<br>HPM_COUNT_ADJUST | HPM_COUNT_ADJUST<br>This register represents a signed three bit value that is used to adjust the High Port Calibration Frequency Count Difference.<br>The range of adjustment is -8 to +7, applied to the difference between Count 1 and Count 2. |
| 27-26<br>— | Reserved |
| 25-16<br>HPM_DENOM | High Port Modulator SDM Denominator<br>This is the denominator that is currently being applied to the High Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |
| 15-10<br>— | Reserved |
| 9-0<br>HPM_NUM_SELECTED | High Port Modulator SDM Numerator<br>This is the numerator that is currently being applied to the High Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## 64.3.2.7.1.8 PLL Low Port Modulator Control (LPM_CTRL)

#### 64.3.2.7.1.8.1 Offset

| Register | Offset |
|---|---|
| LPM_CTRL | 24h |

#### 64.3.2.7.1.8.2 Diagram




#### 64.3.2.7.1.8.3 Fields

| Field | Function |
|---|---|
| 31<br>LPM_SDM_USE_NEG | Use the Negedge of the Sigma Delta clock<br>If this bit is set then the negative edge of the Sigma Delta clock is used to launch the Low Port Modulation word to the VCO Loop Divider. |
| 30-28<br>— | Reserved |
| 27-24<br>LPM_SCALE | LPM Scale Factor<br>This register controls the scaling of the Baseband Frequency Word and is used to match the Modulation Frequency Deviation required to the Low Port Sigma Delta Modulator LSB size in Hz.<br>0000b - No Scaling<br>0001b - Multiply by 2<br>0010b - Multiply by 4<br>0011b - Multiply by 8<br>0100b - Multiply by 16<br>0101b - Multiply by 32 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0110b - Multiply by 64<br>0111b - Multiply by 128<br>1000b - Multiply by 256, this is the intended setting for normal operation.<br>1001b - Multiply by 512<br>1010b - Multiply by 1024<br>1011b - Multiply by 2048<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| 23<br>LPM_D_OVRD | LPM Dither Override Mode Select<br>When this bit is set, the Scaled Baseband Frequency Word applied to the Low Port Sigma Delta Modulator will be dithered if LPM_D_CTRL is set, and not dithered if LPM_D_CTRL is cleared.<br>If this bit is cleared, then the LPM Numerator will be dithered in Radio Receive mode, and also in Radio Transmit mode when the LPM Numerator approaches an Integer value in order to preserve the validity of the Sigma Delta Modulator output. |
| 22<br>LPM_D_CTRL | LPM Dither Control in Override Mode<br>If LPM_D_OVRD is set, this bit turns LPM Dithering on and off. |
| 21-20<br>— | Reserved |
| 19-16<br>LPM_DTH_SCL | LPM Dither Scale<br>This register controls the scale of the Dithering added to the Scaled Baseband Frequency Word before it is applied to the Low Port Sigma Delta Modulator as the LPM Numerator.<br>The unit for the ranges shown below is the LP SDM LSB in Hz.<br>0000b - Reserved<br>0001b - Reserved<br>0010b - Reserved<br>0011b - Reserved<br>0100b - Reserved<br>0101b - -128 to 96<br>0110b - -256 to 192<br>0111b - -512 to 384<br>1000b - -1024 to 768, this is the intended setting for normal operation.<br>1001b - -2048 to 1536<br>1010b - -4096 to 3072<br>1011b - -8192 to 6144<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| 15<br>LPM_DISABLE | Disable LPM SDM<br>This bit controls the Modulation of the Low Port Sigma Delta.<br>If this bit is set, the Low Port Sigma Delta Modulator will be active and control the PLL to maintain a steady frequency based on the current Integer, Numerator, and Denominator values that are being applied.<br>No Modulation or Dithering will be added to the steady frequency result. |
| 14<br>LPM_SDM_INV | Invert LPM SDM<br>If this bit is set the Scaled Baseband Frequency Word, including any Dithering, will be Inverted before it is applied to the Low Port Sigma Delta Modulator. |

*Table continues on the next page...*

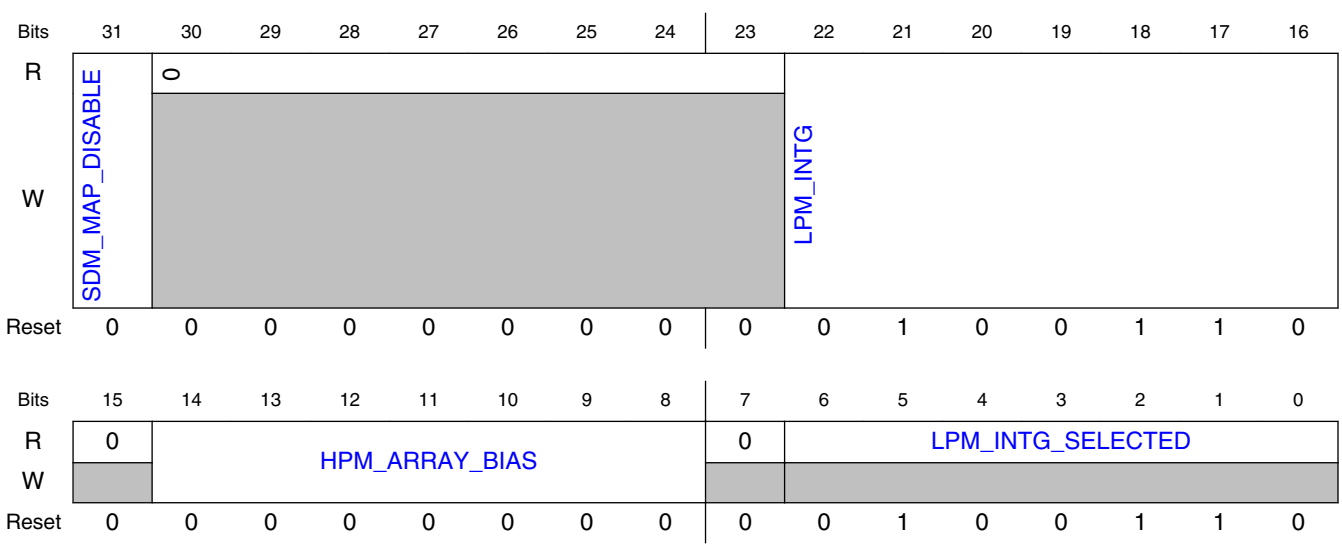
| Field | Function |
|---|---|
| 13<br>LPFF | LPM SDM Invalid Flag<br>This bit is set if the Low Port Sigma Delta Modulator output is invalid due to an error in the fraction applied, and this bit is cleared by writing a 1 to it. |
| 12<br>— | Reserved |
| 11<br>PLL_LD_DISABLE | Disable PLL Loop Divider<br>If this bit is set, the Low Port Sigma Delta Modulator output is disabled, and the PLL Loop Divider value applied to the PLL comes from the PLL_LD_MANUAL register. |
| 10-5<br>— | Reserved |
| 4-0<br>PLL_LD_MANUAL | Manual PLL Loop Divider value<br>If PLL_LD_DISABLE is set, this register is the value that is applied to the PLL Loop Divider. |

## 64.3.2.7.1.9 PLL Low Port Sigma Delta Control 1 (LPM_SDM_CTRL1)

### *64.3.2.7.1.9.1 Offset*

| Register | Offset |
|---|---|
| LPM_SDM_CTRL1 | 28h |

### *64.3.2.7.1.9.2 Diagram*



#### *64.3.2.7.1.9.3 Fields*

| Field | Function |
|---|---|
| 31<br>SDM_MAP_DISABLE | Disable SDM Mapping<br>If this bit is set, the Low Port Sigma Delta Modulator internal frequency mapping based on Protocol specific channel numbers is disabled, and the Radio Channel Frequency is selected by setting the LPM_INTG, LPM_NUM, and LPM_DENOM registers to get a frequency that equals :<br>((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM)) |
| 30-23<br>— | Reserved |
| 22-16<br>LPM_INTG | Manual Low Port Modulation Integer Value<br>If SDM_MAP_DISABLE is set, this register is the value that is applied to the Low Port Sigma Delta Modulator for the Integer, the nominal range is 36 to 39 in decimal. |
| 15<br>— | Reserved |
| 14-8<br>HPM_ARRAY_BIAS | Bias value for High Port DAC Array Midpoint<br>The value of this register represents a signed six bit value that can be used to adjust the midpoint of the High Port Modulator DAC.<br>The range of adjustment is -64 to +63, and the adjustment is made to the High Port Modulation Word before the multiply, and before the split into Integer and Fractional values.<br>The range of adjustment in Hz is approximately +/- 62.5 kHz |
| 7<br>— | Reserved |
| 6-0<br>LPM_INTG_SELECTED | Low Port Modulation Integer Value Selected<br>This shows the Integer value that is currently being applied to the Low Port Sigma Delta Modulator. |

## 64.3.2.7.1.10 PLL Low Port Sigma Delta Control 2 (LPM_SDM_CTRL2)

#### *64.3.2.7.1.10.1 Offset*

| Register | Offset |
|---|---|
| LPM_SDM_CTRL2 | 2Ch |

#### 64.3.2.7.1.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_NUM | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_NUM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.2.7.1.10.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_NUM | Low Port Modulation Numerator<br>If SDM_MAP_DISABLE is set, this register is the signed 27 bit value that is applied to the Low Port Sigma Delta Modulator for the Numerator. For valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

### 64.3.2.7.1.11 PLL Low Port Sigma Delta Control 3 (LPM_SDM_CTRL3)

#### 64.3.2.7.1.11.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_CTRL3 | 30h |

#### 64.3.2.7.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_DENOM | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_DENOM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.2.7.1.11.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_DENOM | Low Port Modulation Denominator<br>If SDM_MAP_DISABLE is set, this register is the signed 27 bit value that is applied to the Low Port Sigma Delta Modulator for the Denominator. For valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

### 64.3.2.7.1.12 PLL Low Port Sigma Delta Result 1 (LPM_SDM_RES1)

#### 64.3.2.7.1.12.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_RES1 | 34h |

#### 64.3.2.7.1.12.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_NUM_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_NUM_SELECTED | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.2.7.1.12.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_NUM_SELECTED | Low Port Modulation Numerator Applied<br>This is the value that is currently being applied to the Low Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## 64.3.2.7.1.13 PLL Low Port Sigma Delta Result 2 (LPM_SDM_RES2)

### *64.3.2.7.1.13.1 Offset*

| Register | Offset |
|---|---|
| LPM_SDM_RES2 | 38h |

### *64.3.2.7.1.13.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_DENOM_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_DENOM_SELECTED | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.2.7.1.13.3 Fields*

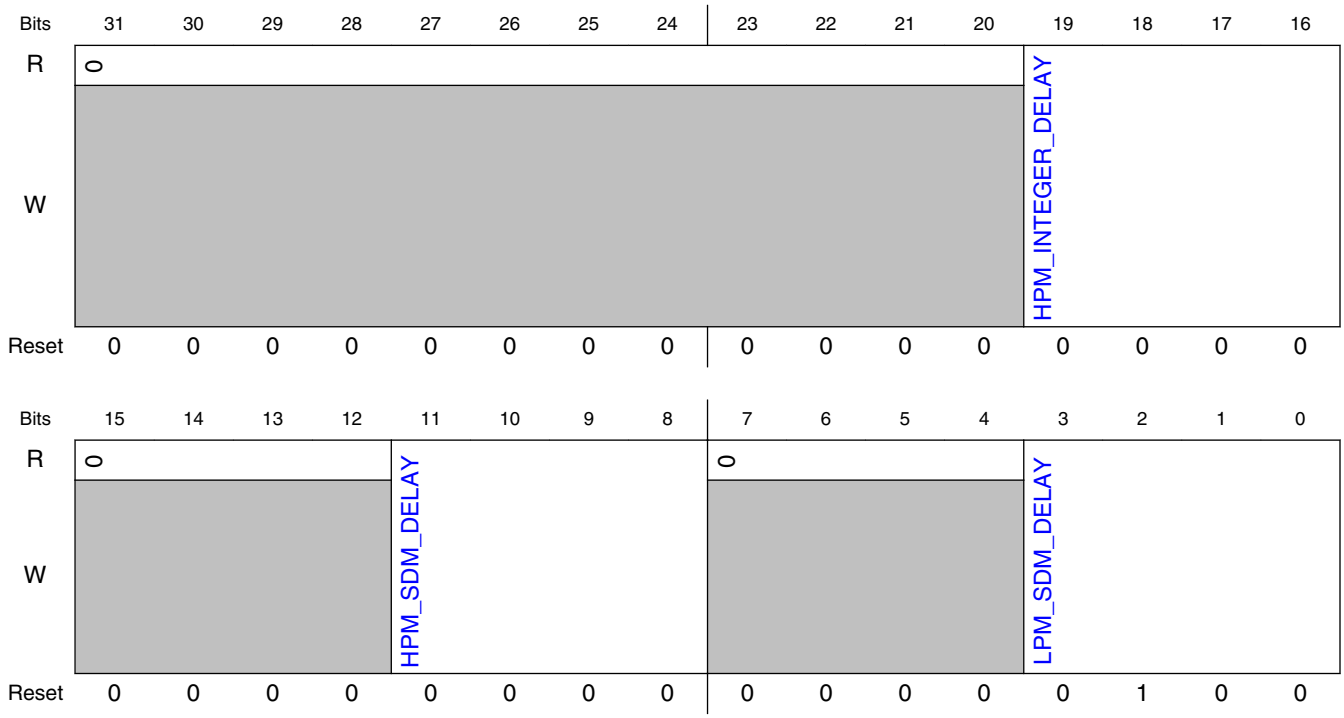
| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_DENOM_SELECTED | Low Port Modulation Denominator Selected<br>This is the value that is currently being applied to the Low Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## 64.3.2.7.1.14 PLL Delay Matching (DELAY_MATCH)

### *64.3.2.7.1.14.1 Offset*

| Register | Offset |
|---|---|
| DELAY_MATCH | 3Ch |

### 64.3.2.7.1.14.2 Diagram




### 64.3.2.7.1.14.3 Fields

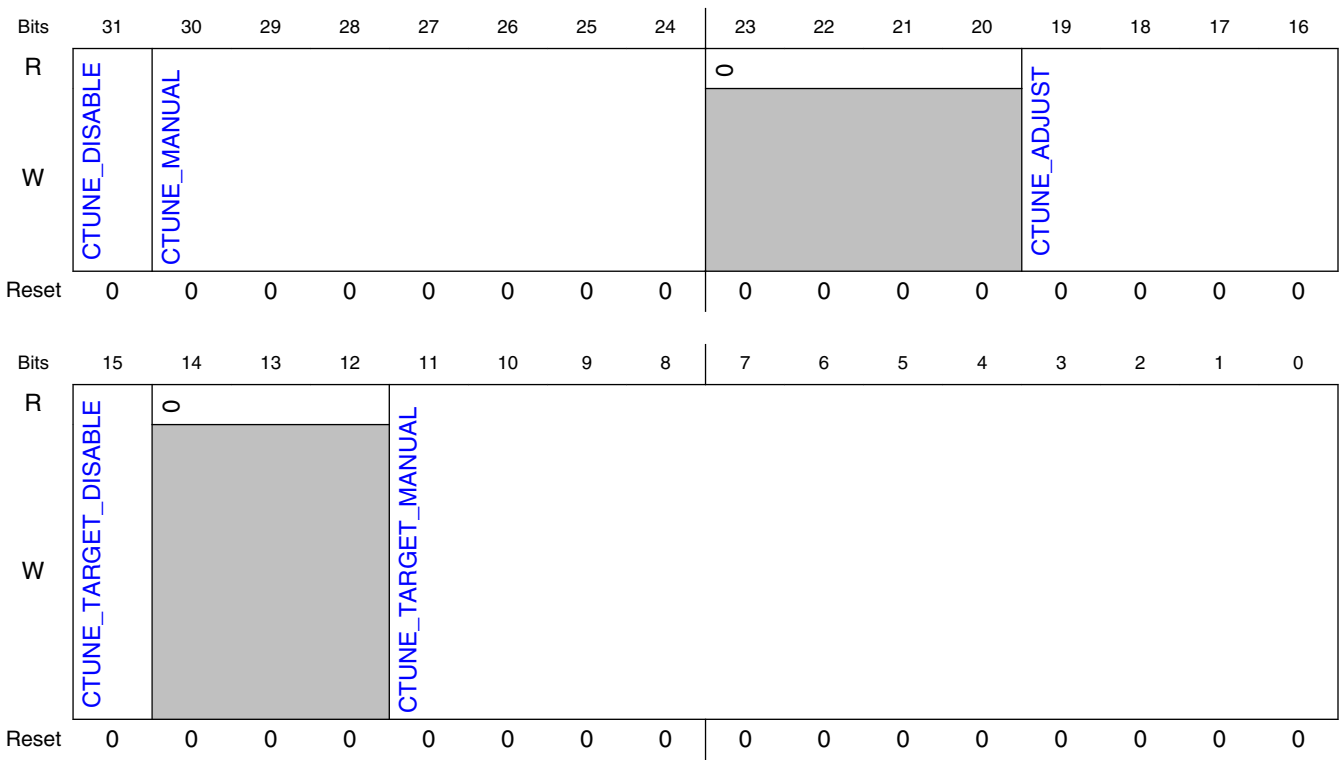
| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>HPM_INTEGER_DELAY | High Port Integer Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock divided by 2 to delay the High Port Integer modulation of the VCO High Port DAC Array. |
| 15-12<br>— | Reserved |
| 11-8<br>HPM_SDM_DELAY | High Port SDM Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock divided by 2 to delay the High Port Sigma Delta modulation of the VCO High Port Fraction. Note that the High Port SDM is clocked by the PLL Sigma Delta Clock but the modulation is based on a divide by 2 version of this same clock. |
| 7-4<br>— | Reserved |
| 3-0<br>LPM_SDM_DELAY | Low Port SDM Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock to delay the Low Port Sigma Delta modulation of the PLL Loop Divider. |

## 64.3.2.7.1.15 PLL Coarse Tune Control (CTUNE_CTRL)

### *64.3.2.7.1.15.1 Offset*

| Register | Offset |
|---|---|
| CTUNE_CTRL | 40h |

### *64.3.2.7.1.15.2 Diagram*




### *64.3.2.7.1.15.3 Fields*

| Field | Function |
|---|---|
| 31<br>CTUNE_DISABLE | Coarse Tune Disable<br>If this bit is set, the Coarse Tune Setting applied to the VCO comes from the CTUNE_MANUAL register. |
| 30-24<br>CTUNE_MANUAL | Manual Coarse Tune Setting<br>If CTUNE_DISABLE is set, this register is the value that is applied to the VCO as the Coarse Tune Setting. |
| 23-20<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19-16<br>CTUNE_ADJUST | Coarse Tune Count Adjustment<br>This register is a signed three bit value that adjusts the PLL Frequency Meter count used in the Coarse Tune Calibration.<br>The range of adjustment is -8 to +7, and the adjustment is only made to the PLL Frequency count used by the Coarse Tune Calibration sequence. |
| 15<br>CTUNE_TARGET_DISABLE | Disable Coarse Tune Target<br>If this bit is set, the Frequency Target presented to the Coarse Tune Calibrator comes from the CTUNE_TARGET_MANUAL register. |
| 14-12<br>— | Reserved |
| 11-0<br>CTUNE_TARGET_MANUAL | Manual Coarse Tune Target<br>If CTUNE_TD is set, this register is the value that is presented to the Coarse Tune Calibrator as the Frequency Target in MHz. The nominal range of this target is from 2360 to 2487 in decimal. |

## 64.3.2.7.1.16 PLL Coarse Tune Results (CTUNE_RES)

### *64.3.2.7.1.16.1 Offset*

| Register | Offset |
|---|---|
| CTUNE_RES | 54h |

### *64.3.2.7.1.16.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CTUNE_FREQ_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CTUNE_BEST_DIFF | | | | | | | | 0 | CTUNE_SELECTED | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.2.7.1.16.3 Fields*

| Field | Function |
|---|---|
| 31-28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27-16<br>CTUNE_FREQ_SELECTED | Coarse Tune Frequency Selected<br>This is the Frequency Target in MHz that is currently being presented to the Coarse Tune Calibrator. |
| 15-8<br>CTUNE_BEST_DIFF | Coarse Tune Absolute Best Difference<br>This is the absolute value of the best difference found during Coarse Tune between the targeted frequency count and the actual frequency count. |
| 7<br>— | Reserved |
| 6-0<br>CTUNE_SELECTED | Coarse Tune Setting to VCO<br>This is the current VCO Coarse Tune setting, it is the result of the Coarse Tune Calibration, unless overridden using CTUNE_DISABLE. |

## 64.3.3 Receiver Digital Module

### 64.3.3.1 Introduction

The RX DIG module for Radio 3.0 implements the receive digital signal processing functions common to BTLE, 802.15.4, and Generic Link Layer modes. Generally this includes conditioning the input samples from the ADC and produces a data output for the PHY and an RSSI/LQI for the Link Layers.

#### 64.3.3.1.1 Features

The rx_dig module includes the following features:

- Decimation Filter
- I/Q Mismatch Correction
- DC Offset Calibration, Estimation and Correction
- Channel Filter
- Sample Rate Converter
- DC Residual Correction
- AGC & RSSI
- Normalizer
- RC Calibration
- AuxPLL Frequency Calibration

#### 64.3.3.1.2 Modes and operations

The rx_dig module supports BTLE, 802.15.4, and Generic Link Layer modes of operation through use of programmable values such as the oversampling rate, channel filter coefficients, etc. Another configuration is possible to simply pass the values received from the ADC.

The table below shows the data rates supported by RX DIG. Note that the ADC is clocked at twice the reference clock frequency, so ADC clock of 64Mhz corresponds to a 32MHz reference clock, and an ADC clock of 52MHz corresponds to a 26MHz reference clock.

**Table 64-21. RX Digital Modes and Rates**

| ADC clock (MHz) | Decimator OSR | Decimation Filter Output Rate(MHz) | SRC conversion factor | SRC Output Rate (MHz) | Use-Case |
|---|---|---|---|---|---|
| 64 | 8 | 8 | bypass | 8 | Generic FSK at 2Mbps |
| | 16 | 4 | bypass | 4 | IEEE 802.15.4, BTLE, or Generic FSK at 1 Mbps |
| | 32 | 2 | bypass | 2 | Generic FSK at 500 kbps |
| | 64 | 1 | bypass | 1 | Generic FSK at 250 kbps; |
| | | | | | |
| 52 | 6 | 8.67 | ZOH; 12/13 | 8 | Generic FSK at 2Mbps |
| | 8 | 6.5 | FOH; 8/13 | 4 | BTLE, or Generic FSK at 1Mbps |
| | 12 | 4.33 | ZOH; 12/13 | 4 | BTLE, or Generic FSK at 1Mbps (alternate) |
| | 16 | 3.25 | FOH; 8/13 | 2 | Generic FSK at 500 kbps |
| | 24 | 2.167 | ZOH; 12/13 | 2 | Generic FSK at 500 kbps (alternate) |
| | 48 | 1.625 | FOH; 8/13 | 1 | Generic FSK at 250 kbps |
| | 64 | 1.083 | ZOH; 12/13 | 1 | Generic FSK at 250 kbps (alternate) |

#### 64.3.3.1.3 Block diagram

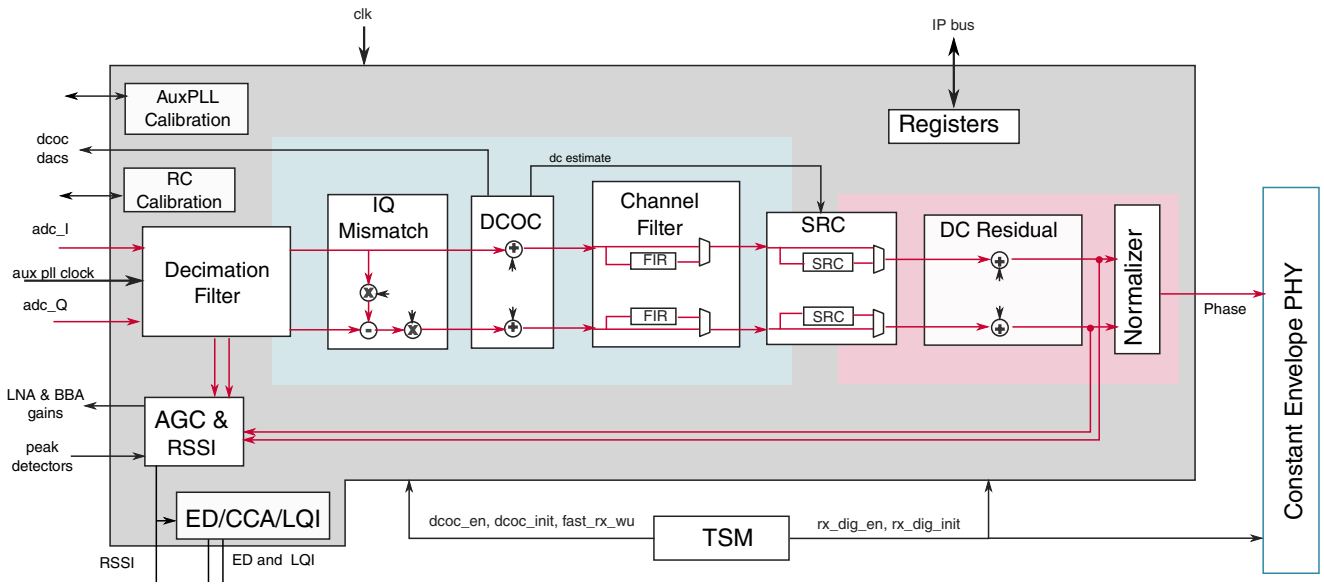



**Figure 64-32. RX DIG Block diagram**

### 64.3.3.2 Functional description

The following sections describe functional details of the RX_DIG module.

The RX_DIG module has two overall funtions. Samples from the ADC are computed to usable formats for the demodulator. Additionally, the module conditions the received sample in various ways, including sending feedback to the analog circuitry.

#### 64.3.3.2.1 Decimation Filter

The decimation filter is a 4th order Cascaded Integrator Comb (CIC) filter. Its frequency response comprises a 4th order sinc filter that supports oversampling ratios (OSR) of 4, 6, 8, 12, 16, 24 , 32, 48, and 64. There are two decimation filters: one for the I channel and one for the Q channel. The decimators consume the sigma-delta modulated output from the ADCs at the Aux PLL frequency clock rate (2x reference clock rate) and output samples at the decimation rate in the reference clock domain which used by the rest of the logic in RX DIG. Refer to Table 64-21 for information on the supported decimation rates.

A block diagram of the decimation filter is shown below.

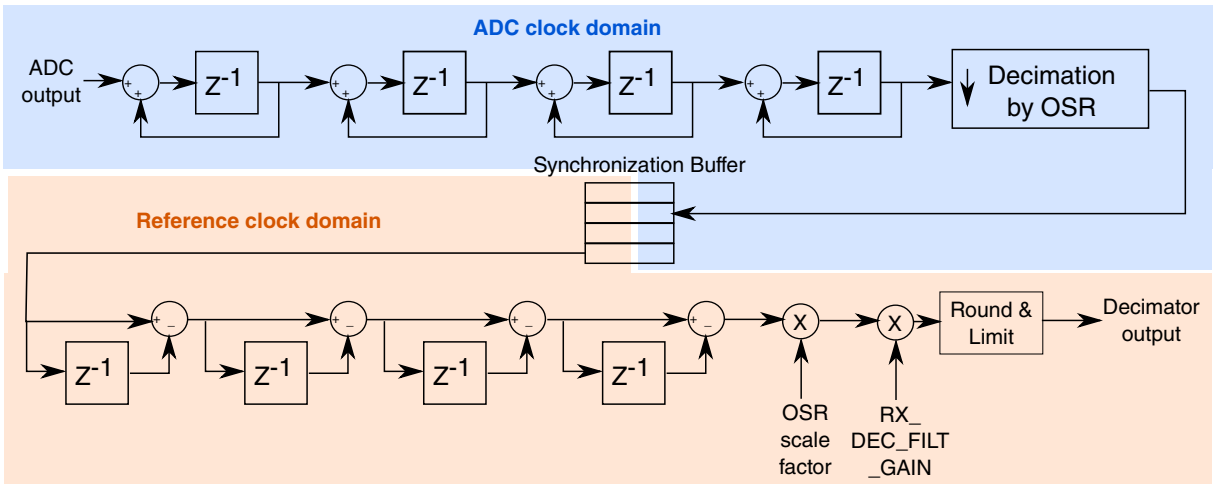



**Figure 64-33. Decimation Filter Block Diagram**

The decimator output scaling is dependant on the OSR scale factor (fixed in hardware, depends on the programmed OSR) and the programmable RX_DEC_FILT_GAIN. The RX_DEC_FILT_GAIN is intended to be programmed based on the maximum signal level at the ADC input, the ADC gain (1code/1.7V), and the OSR scale factor so that the maximum signal range (+2047/-2048) can be realized in the decimator output. An example of RX_DEC_FILT_GAIN programming is shown in the table below for the case where the maximum input signal level to the ADC is 1.0V. The "Maximum Decimator Output" column is computed as

```
Maximum Decimator Ouput = 1.0V * (1code/1.7V) * OSR^4  * OSR_scale_factor *
RX_DEC_FILT_GAIN / 2
```

**Table 64-22. Decimation Filter Gain Table Example, ADC input 1.0V**

| OSR | Gain at output of 4th order CIC (OSR^4) | OSR scale factor | RX_DEC_FILT_GAIN | Maximum Decimator Output |
|---|---|---|---|---|
| 4 | 256 | $2^{4}$ | 1+1/2+1/8+1/16=1.6875 | 2033 |
| 6 | 1296 | $2^{2}$ | 1+1/4+1/16+1/32=1.3438 | 2049 (saturated to +2047/-2048) |
| 8 | 4096 | $2^{0}$ | 1+1/2+1/8+1/16=1.6875 | 2033 |
| 12 | 20736 | $2^{-2}$ | 1+1/4+1/16+1/32=1.3438 | 2049 (saturated to +2047/-2048) |
| 16 | 65536 | $2^{-4}$ | 1+1/2+1/8+1/16=1.6875 | 2033 |
| 24 | 331776 | $2^{-6}$ | 1+1/4+1/16+1/32=1.3438 | 2049 (saturated to +2047/-2048) |
| 32 | 1048576 | $2^{-8}$ | 1+1/2+1/8+1/16=1.6875 | 2033 |
| 48 | 5308416 | $2^{-10}$ | 1+1/4+1/16+1/32=1.3438 | 2049 (saturated to +2047/-2048) |
| 64 | 16777216 | $2^{-12}$ | 1+1/2+1/8+1/16=1.6875 | 2033 |

The decimator is controlled via several programmable bitfields, and there are also a few status bits , as noted below:

- The OSR is programmed using RX_DEC_FILT_OSR
- The RX_DEC_FILT_LP bit can be used to reduce the power consumption of the decimator. When this bit is set, a decimate-by-2 operation is performed on the ADC samples (by simply averaging the ADC output over a window of 2 samples), and the CIC is then operated at 1/2 of the rate programmed via RX_DEC_FILT_OSR.
- The decimation output is scaled based on the OSR and a programmable fractional gain (RX_DEC_FILT_GAIN), as described above. After decimation, the samples are output at the appropriate lower rate in 12-bit signed format.
- The decimator outputs RX_DEC_FILT_SAT_I and RX_DEC_FILT_SAT_Q flags to indicate if the decimator output has saturated during an RX burst
- The single-bit ADC samples can be sampled on either the posedge (default, RX_ADC_NEGEDGE=0) or negedge (RX_ADC_NEGEDGE=1) of the adc_clk.
- The single-bit ADC samples can be mapped as 1'b1->+1, 1'b0->-1 (default, RX_ADC_POL=0) or as 1'b0->+1, 1'b1->-1 (RX_ADC_POL=1).
- The decimator includes a sample buffer to handle the synchronization between the ADC output (adc_clk) clock domain and decimator output (clk) clock domains.
  - The decimator outputs a flag (RX_DEC_FILT_HAZARD) to indicate if a hazard condition has been detected (write and read pointers match) .
  - By default, synchronization hazard correction logic is enabeld (RX_DEC_FILT_HZD_CORR_DIS=0) . In this case, if a hazard condition is detected, the buffer pointers will be adjusted to overcome the hazard condition, though there will be a few corrupted samples at the decimator output. Conversely, if a hazard condition is detected and correction is disabled, the buffer pointers will not be adjusted, and it is possible that the decimator output could be corrupted for a much longer period.

For an ADC clocked at a 64 MHz reference clock, an OSR of 8 is used for BTLE at 2Mbps and an OSR of 16 is used for BLE at 1Mbps, and for IEEE 802.15.4 mode. The frequency response of the decimation filter for these two cases is shown in the figures below.

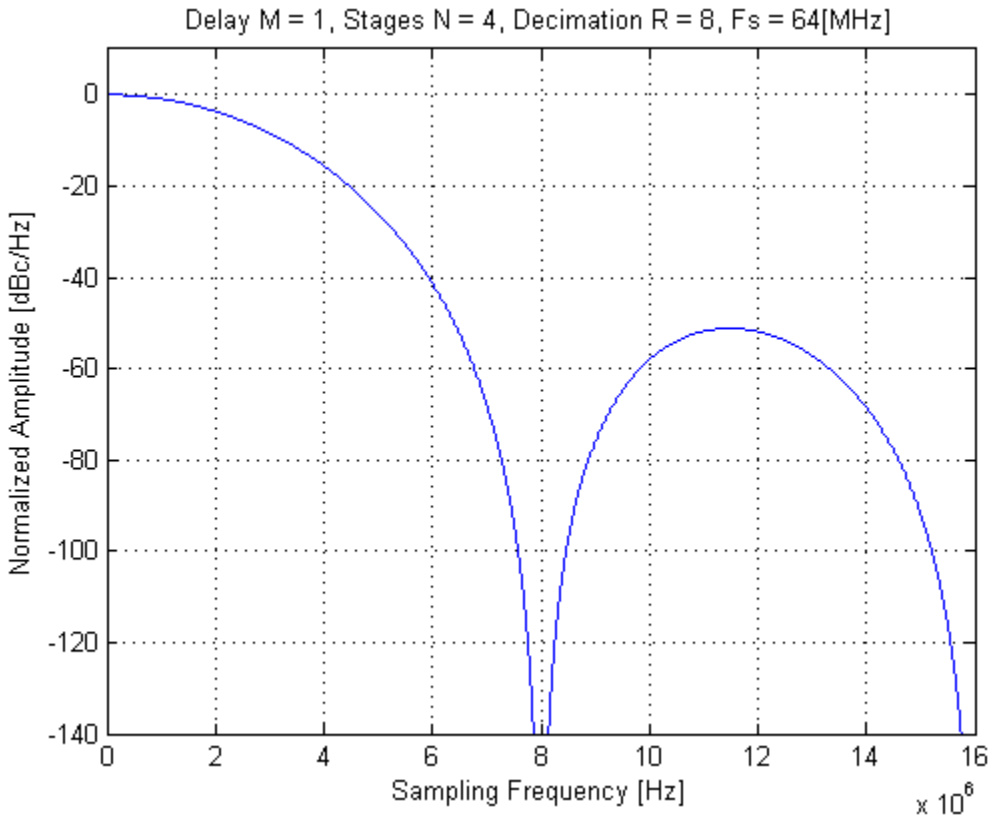



**Figure 64-34. 4th order CIC (OSR=8) Decimation Filter Frequency Response**

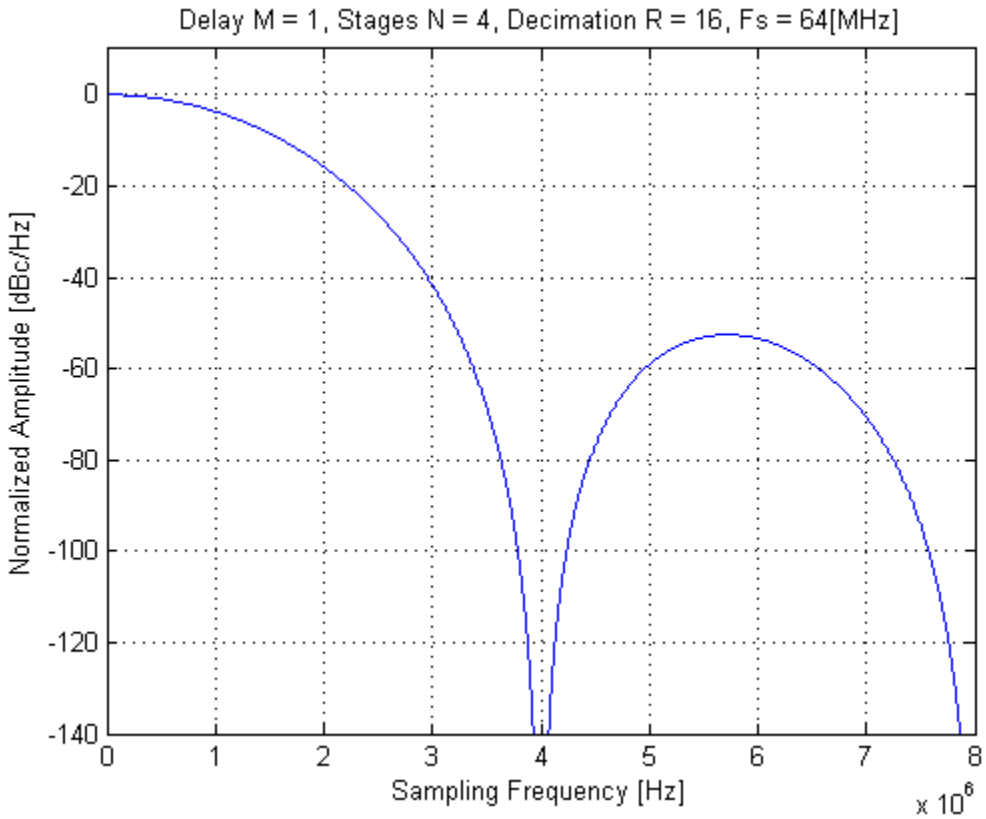



**Figure 64-35. 4th order CIC (OSR=16) Decimation Filter Frequency Response**

The decimator block provides a separate output for wide-band RSSI measurement that is used by the AGC algorithm and CCA1/CCA3/ED/LQI. This output is at 8MHz (32MHz reference clock) or 8.667MHz (26MHz reference clock). As the objective of the wide-band RSSI measurement is to compute a dBVrms value of the ADC output stream with minimal delay, a 2nd order CIC configuration is used. This 2nd order CIC shares the first two CIC integrators with the main datapath, but uses 2 dedicated difference stages.

The decimator for wide-band RSSI measurements is controlled via several programmable bitfields, and there are also a few status bits , as noted below:

- The decimation ratio is determined by XCVR_CTRL[REF_CLK_FREQ]. For a 32MHz reference clock, the decimation ratio is 8 so the decimator output is 8MHz; for a 26MHz reference clock, the decimation ratio is 6 so the decimation output rate is 8.667MHz
- After decimation, the samples are output at the appropriate lower rate in 10-bit signed format. The decimation output is scaled based on the OSR so that the maximum signal level is +/-576 codes.
- The decimator includes a buffer to handle the synchronization between the ADC output (adc_clk) clock domain and decimator output (clk) clock domains.

- The wideband RSSI decimator outputs a flag (RX_RSSI_FILT_HAZARD) to indicate if a hazard condition has been been detected (write and read pointers match)
- By default, synchronization hazard correction logic is enabeld (RX_DEC_FILT_HZD_CORR_DIS=0) . In this case, if a hazard condition is detected, the buffer pointers will be adjusted to overcome the hazard condition, though there will be a few corrupted samples at the decimator output. Conversely, if a hazard condition is detected and correction is disabled, the buffer pointers will not be adjusted, and it is possible that the decimator output could be corrupted for a much longer period.

#### 64.3.3.2.2 AGC & RSSI

The AGC & RSSI module has two main functions. It estimates the energy of an incoming signal (RSSI) and it executes the Automatic Gain Control (AGC) by varying the front-end analog gain. These two functions are designed into the same block and the AGC operation relies, at times, on the estimates provided by the RSSI estimator.

##### 64.3.3.2.2.1 RSSI estimator

The role of the RSSI (Received Signal Strength Indication) estimator is to measure the energy of an incoming signal. The estimator:

- provides a wideband ADC signal level measurement for AGC operation using a decimated version of the ADC output (*RSSI_adc*);
- provides an inband received energy estimate using the "Gain adjustment" block in Figure 64-39;
- targets RSSI accuracy with tunable gains and coefficients.

RSSI accuracy is better than +/- 3dB for input signals in the range of -50dBm to 0dBm. This is the intended operating range for the AGC system. RSSI accuracy is better than +/- 6dB for all input signals below -50dBm.

RSSI estimation is performed using the binary samples output by the ADC. After this operation the signal is filtered using a second order CIC and rate reduction (to 8 MHz if 32 MHz crystal is used or to 8.66MHz for a crystal frequency of 26 MHz). The signal magnitude is then estimated with an L1-norm approach as follows:

$$\textbf{\textit{Magnitude}}(i) = max(|I_i|,|Q_i|) + {}^{3}/_{8} * min(|I_i|,|Q_i|)$$

The estimated magnitude is then passed through a smoothing, low-pass single-tap IIR filter with the transfer function (update factor is α in Figure 64-39):

$$y[n] = (1-\alpha)*y[n-1] + \alpha*x[n]$$

In order to reduce the filter settling duration, when *cca1_ed_trig* is asserted, the α IIR output is seeded with the first value of the estimated magnitude. Configuration of this filter is described in Table 64-25, parameter *rssi_iir_weight*. Then a down-sampler is used to reduce the sampling rate (8 MHz or 8.66 MHz) to a lower rate. A down-sampling cycle is as follows: *N* samples are accumulated, then the results is divided by *N*, division result is output (at a rate *N* times smaller than the input signal) and the internal accumulator is reset to 0.

Following the smoothing filter and down-sampler, the RSSI estimator converts the approximated magnitude to dB. This conversion is achieved using a look-up table ("lin2dB" block) that approximates:

$$RSSI_adc = 20*log_{10}(^{approx.\ magnitude}/_{ADC\ full\ scale})$$

The computed value *RSSI_adc* is intended to be used by the AGC mechanism.

*RSSI_adc* estimate is converted to an estimate for the signal strength at the antenna, in the "Gain Adjustment" block, using parameters as follows:

$$RSSI = RSSI_adc - lnm_gain - bbf_gain + rssi_adj+13$$

where *rssi_adj* is described in Table 64-25 and represents a constant value which allows taking into account different adjustments needed. The adjustment which accounts for scaled dBVrms to dBm conversion is considered using the constant 13. *lnm_gain* and *bbf_gain* values have resolution of a quarter dB.

The rest of the blocks in Figure 64-39 are used for CCA1, CCA3 and ED. In order to obtain these measurements an averager is used ("RSSI Averager") which essentially is another down-sampler with a factor *M*. This block functions in the same manner as the *N* order down-sampler. In the wideband mode context, the block "RSSI Averager" and the sub-sequent blocks are enabled only if a CCA/ED measurement is in progress or if the test mode is enabled.

#### 64.3.3.2.2.2 AGC

The Automatic Gain Control (AGC) system is designed to ensure that the received signal does not become distorted because of clipping in the analog receive chain, while attempting to maintain an optimal signal level for demodulation. There are two locations where gain may be adjusted: analog BBA, and analog LNA. The AGC system

manipulates these gains to maintain a targeted ADC input signal level. The targeted ADC signal level is chosen as a compromise between receiver linearity, blocker headroom requirements and the goal to maximize the digital signal SNR that is provided to the protocol-specific PHY demodulators.

Additionally, the AGC system may be manually controlled by setting the USER_BBA_GAIN_EN and USER_LNA_GAIN_EN bits. If these bits are set, the gains selected in BBA_USER_GAIN and LNA_USER_GAIN will be applied.

The key components of the AGC system are: peak detectors in the analog front end, gain control, RSSI estimation, and a state machine. Additionally, the AGC system has appropriate interation with the DCOC block to maintain a coherent state of the gains.

If the manual control of the gains is not enabled, and the AGC system is enbled, the system will automatically control the gains based on peak detector states and/or RSSI measurements.

The AGC sytem operated with a state machine that is depicted in Figure 64-36 and has the following significant modes:

- **IDLE**: No actions are taken. The system waits for either rx_init (for normal receive operation) or cal_init (for DC calibration).
- **DC Calibration**: When cal_init is observed while in IDLE, the AGC enters the DC Calibration state. According to values programmed in DCOC_CAL_GAIN, gains will be applied for DC CAL 1, 2, 3 phases and the DCOC block can take measurements of the DC level with those gains applied. During DC Calibraion, the AGC system takes no other action.
- **FAST AGC**: When rx_init is observed while in IDLE, the system advances to FAST AGC. Maximum gain is applied immediately. While in FAST AGC, the high level peak detectors are observed to see if clipping is occurring in the analog. If a high peak detector becomes set, a gain adjustment is applied based on the settings AGC_DOWN_LNA_STEP_SZ, AGC_DOWN_BBA_STEP_SZ, and AGC_GAIN_TBL. Depending on which of the two peak detectors is set, the corresponding step is taken in the gain table. When a change is made, a timer prevents further changes until the gain change has had time to settle. This time is defined by the settings LNA_GAIN_SETTLE_TIME and BBA_GAIN_SETTLE_TIME. This time is also utilized to reset the state of the peak detectors. FAST AGC can be exited by reaching the minimum possible gain setting, or if the fast_expire timer expires. This time is configured with the AGC_FAST_EXPIRE setting.
- **SLOW AGC**: The SLOW AGC state allows for observation of RSSI, while still monitoring the peak detectors. Alternatively, RSSI can be ignored and the low peak detectors can be used to take upward gain steps. These steps are smaller than the steps taken in FAST AGC. Before entering SLOW AGC, there is a PRESLOW state

where RSSI_adc is quickly checked. If there is not enough ADC headroom (as determined by PRESLOW_DOWN_THRESH), a small downward gain step is taken before proceding to SLOW AGC. If there is too much ADC headroom (as determined by PRESLOW_UP_THRESH), a small upward gain step is taken before proceeding to SLOW AGC. Similar to FAST AGC, SLOW AGC will allow for a gain settling time whenever an adjustment is made. SLOW AGC can be exited because of several condtions. If a high peak detector indicates clipping, the system immediately moves to the FAST AGC state. If a drastic change in measured RSSI occurs, the system will similarly transition to the FAST AGC state. Finally, if the slow_expire timer (ipr_agc_fast_expire when in slow mode) reaches the pre-determined time, the system will move to HOLD AGC.

- **HOLD**: The AGC HOLD state will not take any actions unless a high peak detector is set, a large RSSI change is experienced, or the hold_expire timer expires. These conditions cause transitions into approriate states to take action. The expiration of the hold_timer moves the system back to SLOW AGC, where monitoring of the RSSI can be done and gain increases can happen, if appropriate.

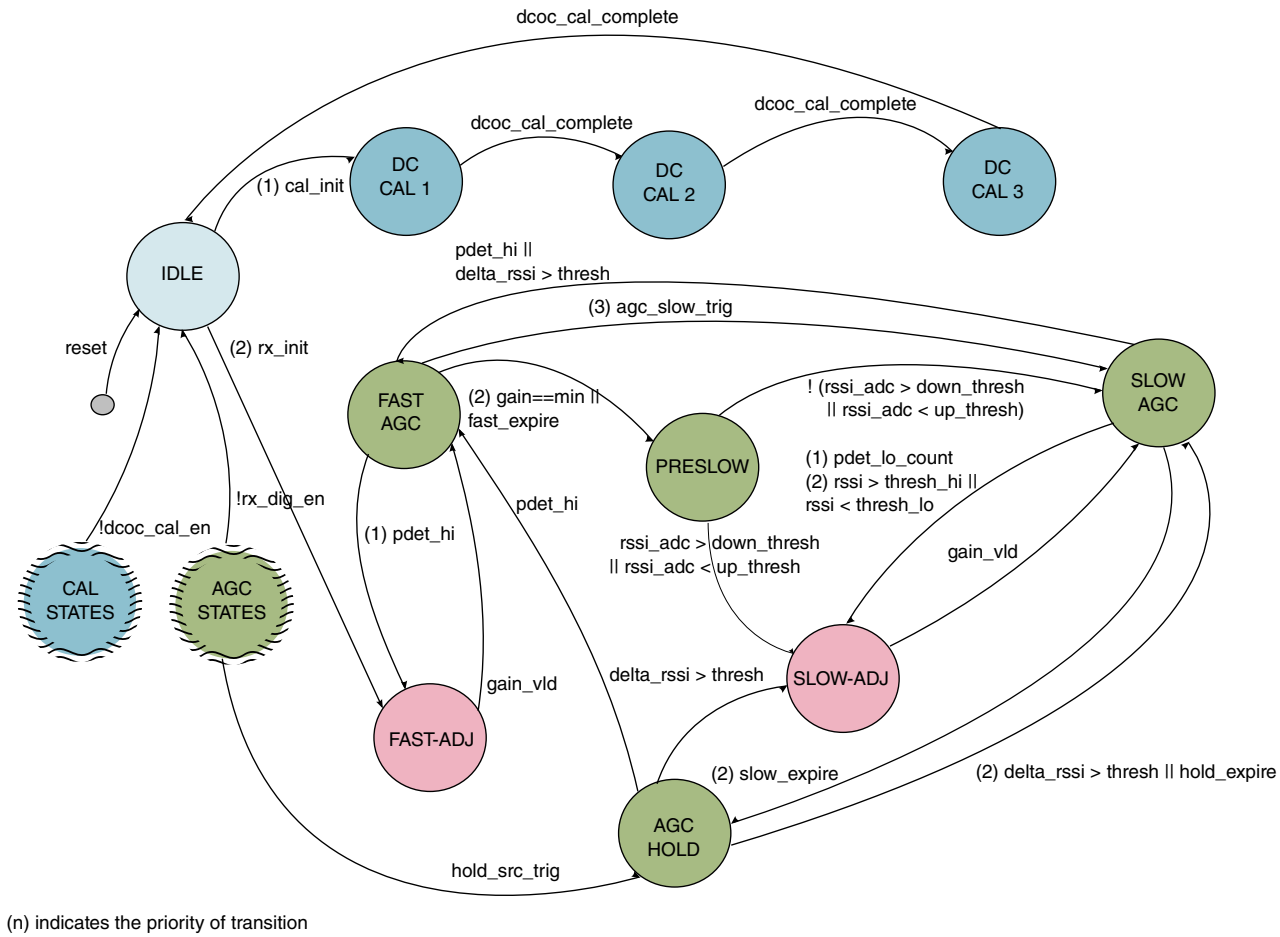



**Figure 64-36. AGC State Machine**

As previously mentioned, gain steps are performed by striding through the user programmed gain table. There are 27 entries in the table, and each entry contains an index for LNA gain and an index for BBA gain. The mapping of these indicies to codes and nominal gains is described in Table 64-23 and Table 64-24. Entry 0 is interpreted as the lowest possible gain setting, and entry 26 the highest.

**Table 64-23. LNA Gains**

| lna_gain (AGC_GAIN_TBL) | LNA Gain (dB) | lna_gain code |
|---|---|---|
| 0 | -8.6 | 01 |
| 1 | -3.5 | 02 |
| 2 | 2.2 | 03 |
| 3 | 13.9 | 04 |
| 4 | 19.8 | 08 |
| 5 | 22.7 | 0C |

*Table continues on the next page...*

**Table 64-23. LNA Gains (continued)**

| lna_gain (AGC_GAIN_TBL) | LNA Gain (dB) | lna_gain code |
|---|---|---|
| 6 | 28.6 | 18 |
| 7 | 34.4 | 30 |
| 8 | 39.9 | 9C |
| 9 | 45.4 | FC |

**Table 64-24. BBA Gains**

| bba_gain (AGC_GAIN_TBL) | BBA Gain (dB) | bba_res_tune code |
|---|---|---|
| 0 | 0 | A |
| 1 | 3 | 9 |
| 2 | 6 | 8 |
| 3 | 9 | 7 |
| 4 | 12 | 6 |
| 5 | 15 | 5 |
| 6 | 18 | 4 |
| 7 | 21 | 3 |
| 8 | 24 | 2 |
| 9 | 27 | 1 |
| A | 30 | 0 |

### 64.3.3.2.3 RSSI, Energy Detection, CCA1 and LQI module

There are two modules that achieve RSSI, Energy Detection, CCA, and LQI: rx_agc_rssi_est and rx_cca_ed_lqi.

See the AGC section for additional RSSI details.

The RSSI module is responsible with the generation of estimates used for AGC and RSSI or required by the upper layers of the protocols stack. Two modes are defined:

- **Wideband mode**, which uses as input the samples from the output of the ADC decimator (signals *dec_8m_out_i[10:0]* and *dec_8m_out_q[10:0]*). In this mode the incoming samples have sampling frequency of either 8 MHz (if 32 MHz crystal is used) or 8.66 MHz(if 26 MHz crystal is used).
- **Narrowband mode**, which uses as input the samples from the output of the min-max residual DC offset compensator (signals *narrow_band_out_i[11:0* and *narrow_band_out_q[11:0]*). In this mode, the incoming samples have sampling frequency of either 2, 4 or 8 MHz depending on PHY configuration.

The cca_ed_lqi module uses rssi outputs from the rssi_est module.

**Table 64-25. Parameters**

<table>
<tr><th>Parameter</th><th>Notes</th></tr>
<tr><td>meas_trans_to_idle</td><td>Flag establishing the state machine (see Figure 64-38) transion following an LQI or CCA1/ED measurement.<br><table><tr><th>Value</th><th>Behaviour</th></tr><tr><td>0</td><td>Module transitions to RSSI state</td></tr><tr><td>1</td><td>Module transitions to IDLE state</td></tr></table></td></tr>
<tr><td>cca1_thresh[7:0]</td><td>Threshold used for CCA1 channel state determination expressed in dBm. cca1_thresh is not programmed in RX_DIG, it is programmed in the 802.15.4 link layer.</td></tr>
<tr><td>rssi_iir_weight[3:0]</td><td>Update factor used for IIR filtering of L1 magnitude and noise (α in Figure 64-39 and Figure 64-40).<br><table><tr><th>Bit-field value</th><th>α</th></tr><tr><td>0 (bypass)</td><td>1</td></tr><tr><td>1</td><td>1/2</td></tr><tr><td>2</td><td>1/4</td></tr><tr><td>3</td><td>1/8</td></tr><tr><td>4</td><td>1/16</td></tr><tr><td>5</td><td>1/32</td></tr><tr><td>6-15</td><td>Reserved</td></tr></table></td></tr>
<tr><td>rssi_n_window_nb[2:0]</td><td>Downsampling factor for filtered magnitude (N in Figure 64-39 and Figure 64-40).<br><table><tr><th>Bit-field value</th><th>N</th></tr><tr><td>0 (bypass)</td><td>1</td></tr><tr><td>1</td><td>2</td></tr><tr><td>2</td><td>4</td></tr><tr><td>3</td><td>8</td></tr><tr><td>4</td><td>16</td></tr><tr><td>5</td><td>32</td></tr></table></td></tr>
<tr><td>rssi_noise_avg_factor[2:0]</td><td>Factor used for RSSI and SNR averaging (M in Figure 64-39 and Figure 64-40).<br><table><tr><th>Bit-field value</th><th>M</th></tr><tr><td>0 (bypass)</td><td>1</td></tr><tr><td>1</td><td>64</td></tr><tr><td>2</td><td>70</td></tr><tr><td>3</td><td>128</td></tr><tr><td>4</td><td>139</td></tr></table></td></tr>
</table>

*Table continues on the next page...*

## Table 64-25. Parameters (continued)

<table>
<tr><th>Parameter</th><th>Notes</th></tr>
<tr><td></td><td>
<table>
<tr><th>Bit-field value</th><th>M</th></tr>
<tr><td>5</td><td>256</td></tr>
<tr><td>6</td><td>277</td></tr>
<tr><td>7</td><td>512</td></tr>
</table>
</td></tr>
<tr><td>rssi_noise_avg_delay[5:0]</td><td>Programmable delay used to delay the enable of averagers "RSSI Averager" and "Noise Averager" after a cca1_ed_trig assertion (in wideband mode) or after a aa_sfd_matched assertion (in narrowband mode). The delay is expressed in ticks of frequency Fs/N, where Fs is the ADC decimator output sampling frequency in wideband mode or the base-band sampling frequency in the narrowband mode.</td></tr>
<tr><td>lqi_corr_thresh[7:0]</td><td>Threshold used to compare corr_mag (see Figure 64-40).</td></tr>
<tr><td>corr_cnt_thresh[7:0]</td><td>Threshold used to compare the counted correlation magnitudes exceeding lqi_corr_thresh (see Figure 64-40).</td></tr>
<tr><td>sel_snr_mode</td><td>Selection of the SNR input of LQI Calculator block in Figure 64-40.
<table>
<tr><th>Bit value</th><th>Used input</th></tr>
<tr><td>0</td><td>SNR estimate</td></tr>
<tr><td>1</td><td>Mapped correlation magnitude</td></tr>
</table>
</td></tr>
<tr><td>lqi_rssi_sens[3:0]</td><td>Unsigned integer used for calculation of LQI Sensitivity, through relation:<br>Sensitivity = -103+lqi_rssi_sens.</td></tr>
<tr><td>lqi_rssi_weight[2:0]</td><td>RSSI weight used for LQI calculation, $w_{RSSI}$.
<table>
<tr><th>Bit-field value</th><th>$w_{RSSI}$</th></tr>
<tr><td>0</td><td>2.0</td></tr>
<tr><td>1</td><td>2.125</td></tr>
<tr><td>2</td><td>2.25</td></tr>
<tr><td>3</td><td>2.375</td></tr>
<tr><td>4</td><td>2.5</td></tr>
<tr><td>5</td><td>2.625</td></tr>
<tr><td>6</td><td>2.75</td></tr>
<tr><td>7</td><td>2.875</td></tr>
</table>
</td></tr>
<tr><td>rssi_adj[7:0]</td><td>RSSI calculation adjustment (8-bit signed with quarter dB resolution).</td></tr>
<tr><td>snr_lqi_weight[3:0]</td><td>SNR weight used for LQI calculation, $w_{SNR}$.
<table>
<tr><th>Bit-field value</th><th>$w_{SNR}$</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

## Table 64-25. Parameters (continued)

<table>
<tr><th>Parameter</th><th>Notes</th></tr>
<tr><td></td><td>
<table>
<tr><th>Bit-field value</th><th>$w_{SNR}$</th></tr>
<tr><td>2</td><td>1.125</td></tr>
<tr><td>3</td><td>1.250</td></tr>
<tr><td>4</td><td>1.375</td></tr>
<tr><td>5</td><td>1.500</td></tr>
<tr><td>6</td><td>1.625</td></tr>
<tr><td>7</td><td>1.750</td></tr>
<tr><td>8</td><td>1.875</td></tr>
<tr><td>9</td><td>2.000</td></tr>
<tr><td>10</td><td>2.125</td></tr>
<tr><td>11</td><td>2.250</td></tr>
<tr><td>12</td><td>2.375</td></tr>
<tr><td>13</td><td>2.500</td></tr>
<tr><td>14</td><td>2.625</td></tr>
<tr><td>15</td><td>2.750</td></tr>
</table>
</td></tr>
<tr><td>snr_adj[5:0]</td><td>SNR calculation adjustment (6-bit signed with half dB resolution).</td></tr>
<tr><td>lqi_bias[3:0]</td><td>Bias used for LQI calculation, $LQI_{bias}$ using formula:<br>$LQI_{bias}$ = -36 + 3*lqi_bias.</td></tr>
</table>

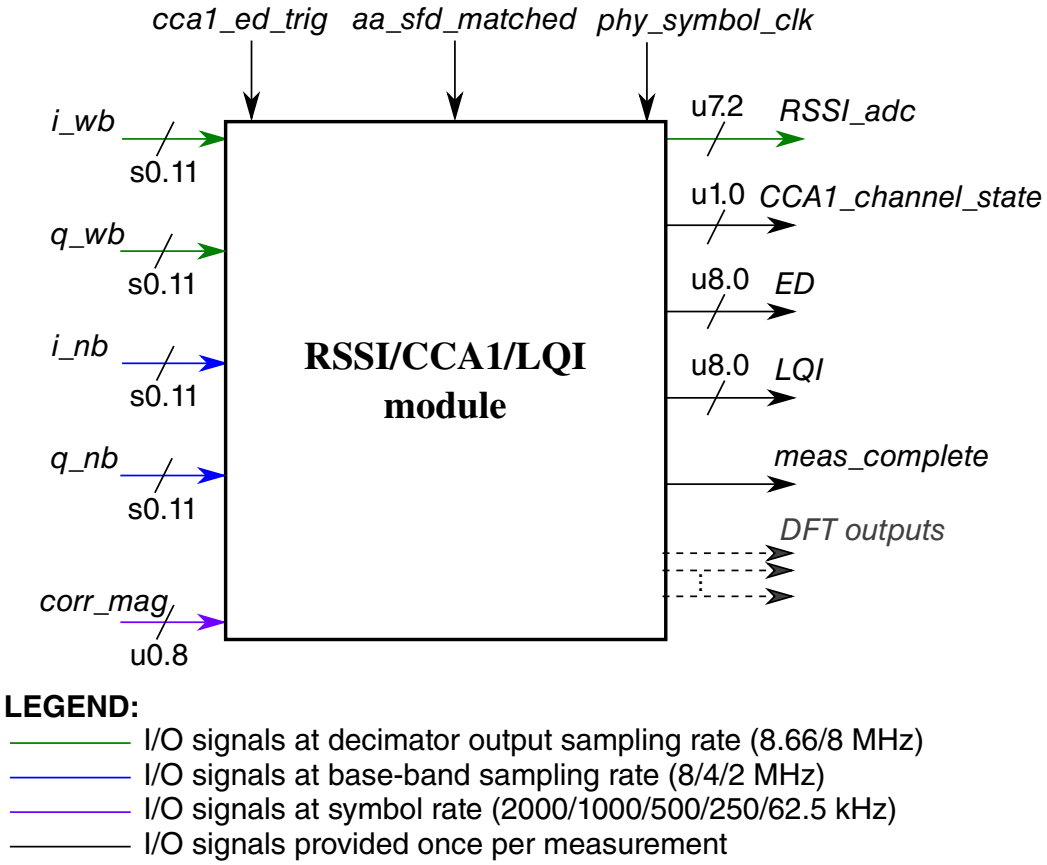



**Figure 64-37. Top level view of the block diagram**

Following table is showing the valid configurations and the valid outputs thereto.

**Table 64-26. Valid configurations and corresponding outputs**

| *cca1_ed_trig* | *sfd_aa_match* | State | Valid outputs |
|---|---|---|---|
| 0 | 0 | RSSI | *RSSI_adc* |
| 1 | 0 | CCA1/ED measurement | *RSSI_adc*<br>*zb_cca1_channel_state*<br>*zb_cca3_channel_state*<br>*ED* |
| 0 | 1 | LQI measurement | *LQI* |
| 1 | 1 | LQI measurement | *LQI* |

Following figure describes the state machine of the module.

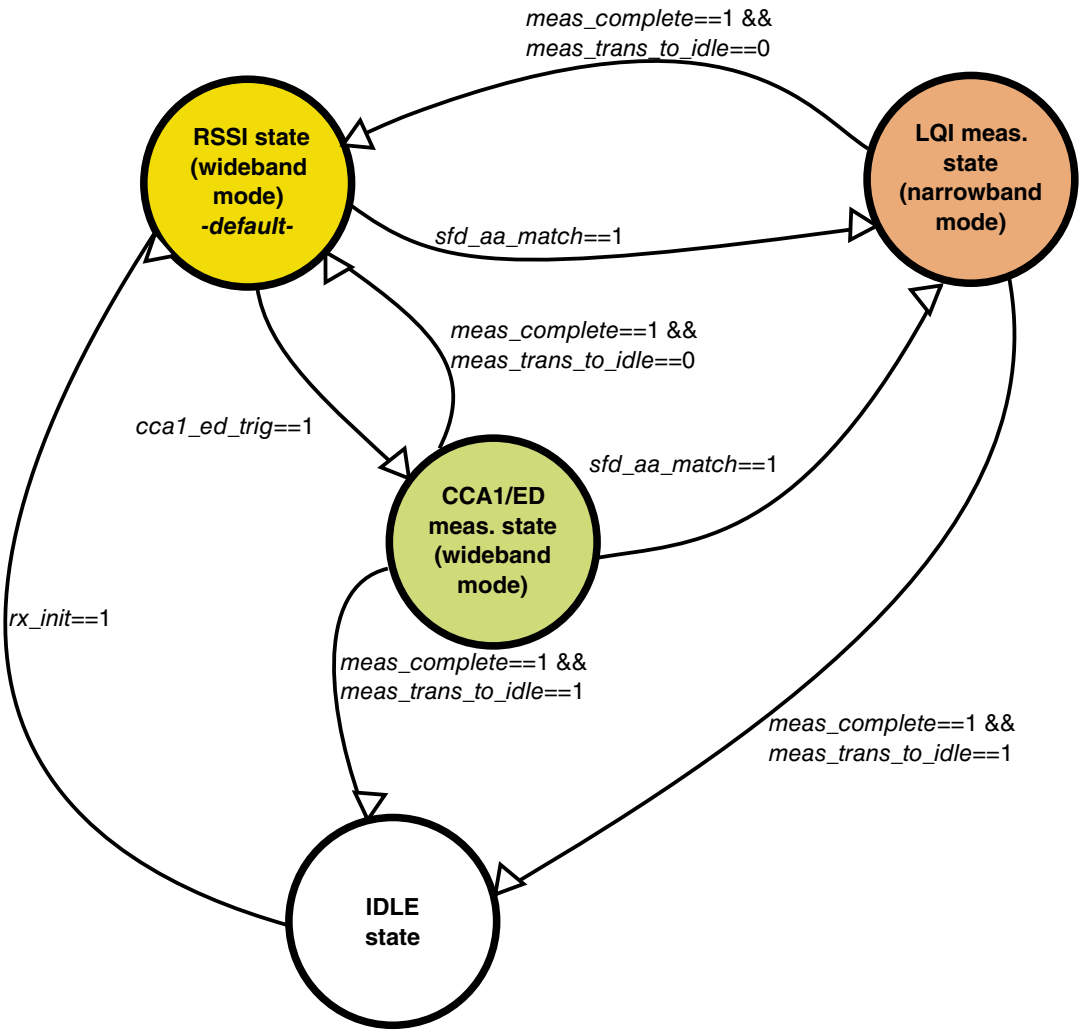



**Figure 64-38. Module state machine**

### 64.3.3.2.3.1 Wideband mode

The instantiation of the block diagram corresponding to wideband mode is illustrated in Figure 64-39. Blocks drawn using dashed lines are shared between the wideband and narrowband modes. Following functions are intended in the wideband mode (as described in Table 64-26):

- Estimation of RSSI at the output of the ADC decimator (*RSSI_adc*) used for AGC and of received inband RSSI;
- Energy detection (*ED*), required by the upper layers, which is an 8 bits value achieved through measuring the energy of the channel;
- Clear Channel Assessment Mode 1 (*zb_cca1_channel_state*), required by the upper layers, which is a flag indicating whether the channel is busy or idle.

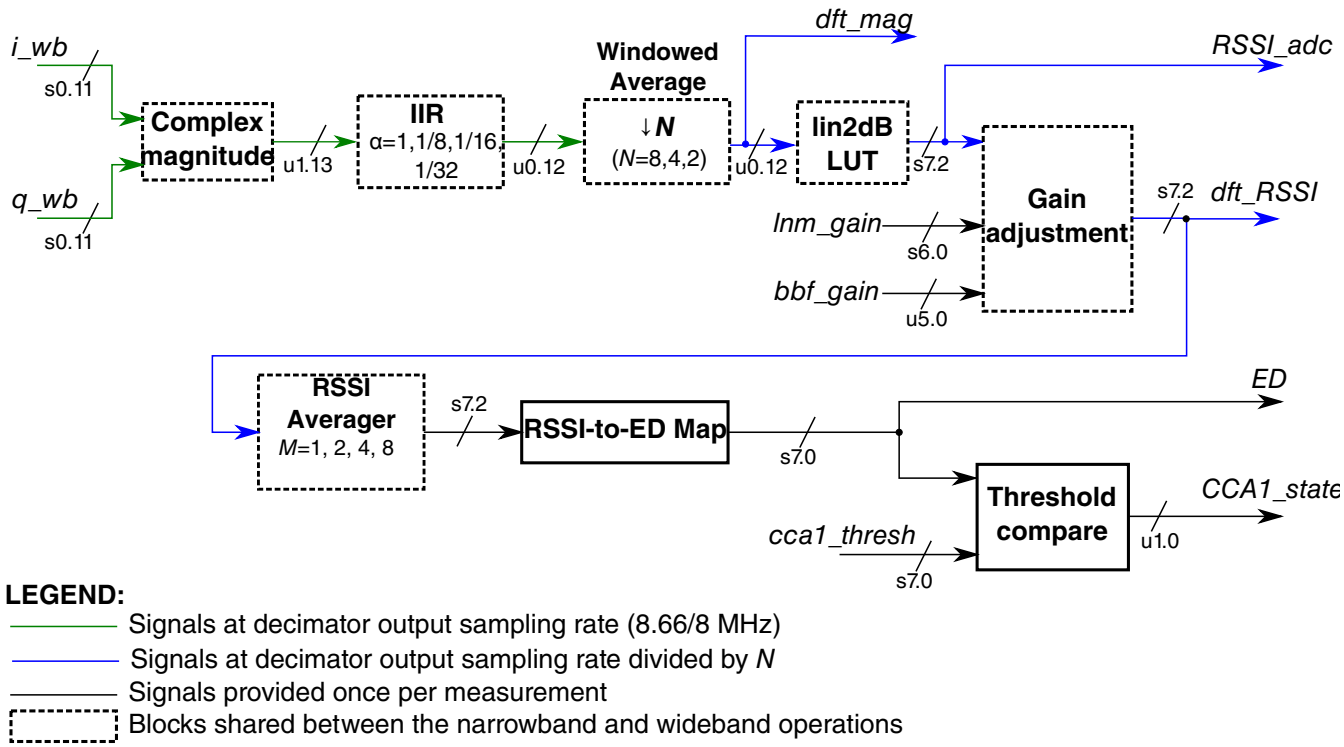



**Figure 64-39. Block diagram instantiated for wideband mode**

#### *64.3.3.2.3.1.1 ED and CCA1*

Module transitions to CCA1/ED measurement state and ED and CCA1 operations start based on a command signal *cca1_ed_trig* (configuration described in Table 64-26). When this signal is asserted, signal *meas_complete* is forced to low state.

When a *cca1_ed_trig* command is asserted, "RSSI averager" is enabled after a programmable delay in order to take into consideration the settling time of the IIRs (programmable delay is configured via parameter *rssi_noise_avg_delay*). After a result is produced by the "RSSI averager" block (i.e. after *M*N* samples have been input to the RSSI/CCA/ED/LQI module since "RSSI averager" has been enabled) in Figure 64-39 signal *meas_complete* is asserted. This signal is intended to let know the external modules consuming the outputs that ED, CCA1 and CCA3 values are ready. In a non-test mode, the module returns to RSSI state after asserting *meas_complete* signal, if parameter *meas_trans_to_idle* is 0 or it transitions to IDLE state if parameter *meas_trans_to_idle* is 1 (see Table 64-25). The measured values of ED, CCA1 and CCA3 are stored until the next deassertion of signal *meas_complete*.

However, the module may be configured to remain in CCA1/ED measurement state for an arbitrary period of time in a test scenario.

**Table 64-27. Typical CCA/ED measurement window values and corresponding configuration**

| Crystal frequency [MHz] | ADC decimator output sampling frequency [MHz] | *N* | *M* | Approximative duration [µs] |
|---|---|---|---|---|
| 32 | 8 | 16 | 64 | 128 |
| 26 | 8.66 | 16 | 70 | 128 |
| 32 | 8 | 8 | 128 | 128 |
| 26 | 8.66 | 8 | 139 | 128 |
| 32 | 8 | 8 | 64 | 64 |
| 26 | 8.66 | 8 | 70 | 64 |
| 32 | 8 | 4 | 64 | 32 |
| 26 | 8.66 | 4 | 70 | 32 |

Based on computed value *RSSI* an energy detection value *ED* is computed, expressed in dBm. This value is obtained by rounding the output of the "RSSI Averager" block to 8 signed bits by using block "RSSI-to-ED Map" in Figure 64-39.

Assessment of the channel's state is by comparing value *ED* with threshold *cca1_thresh*. If the threshold is exceeded at the end of the measurement then *zb_cca1_channel_state* is put to '1'. Otherwise it is put to '0'.

The CCA3 estimation *zb_cca3_channel_state* is realized by combining *zb_cca1_channel_state* and *zb_cca2_channel_state* . This is handled outside RX_DIG.

#### 64.3.3.2.3.2 Narrowband mode

The instantiation of the block diagram corresponding to narrowband mode is illustrated in Figure 64-40. Blocks drawn using dashed lines are shared between the wideband and narrowband modes. Following functions are intended for this mode (as described in Table 64-26):

- Link Quality Indication determination (*LQI*), required by the upper layers. LQI is an 8 bits value obtained by combining inband RSSI (corresponding to channel filtered samples), SNR (signal-to-noise ratio) and correlation magnitudes;

LQI is computed in three steps:

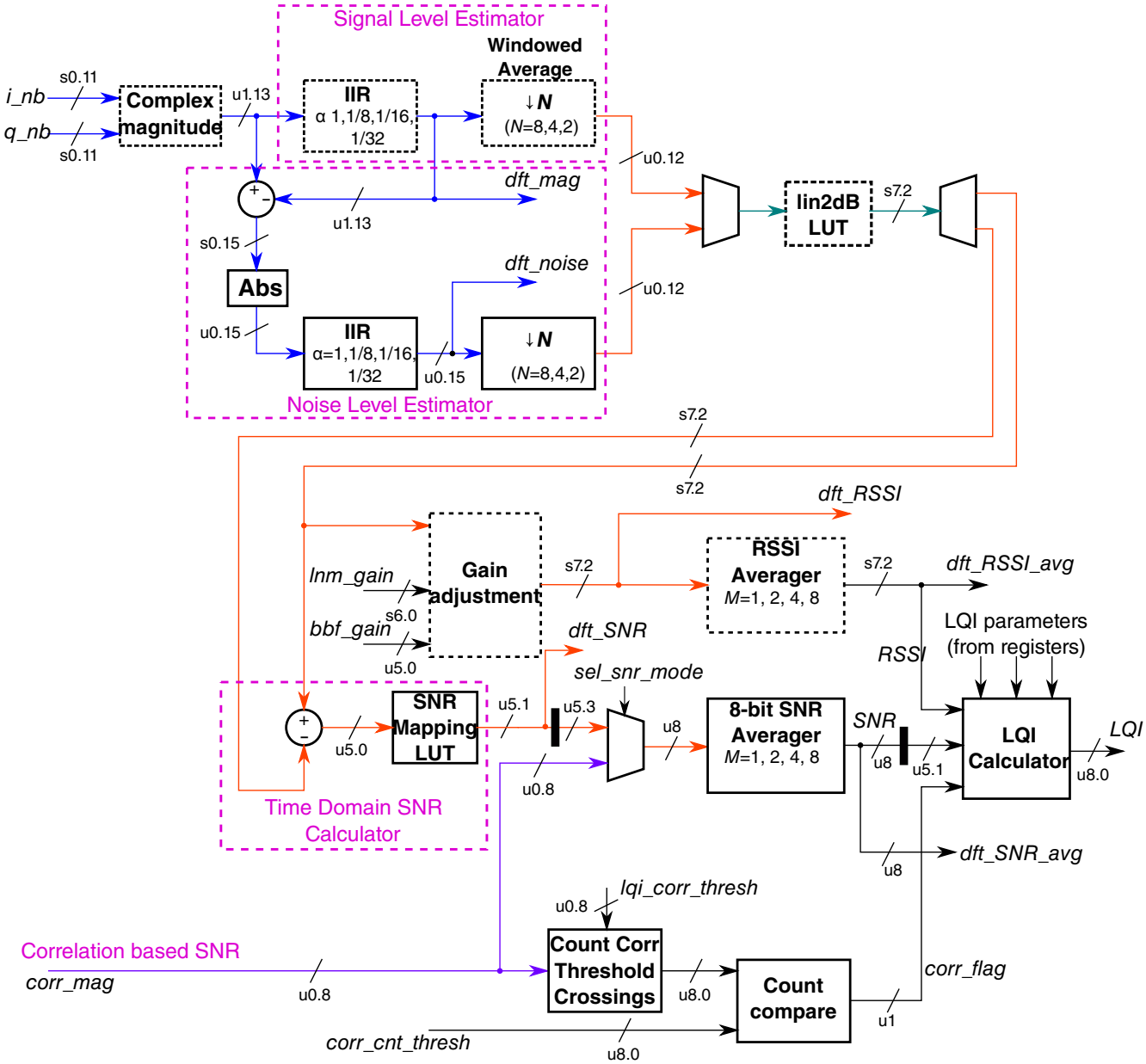
- RSSI (using narrow band I/Q samples) and SNR estimation;
- Correlation flag determination, used to determine whether receiver is at the sensitivity boundary or not;
- LQI calculation which represents a linear combination of RSSI, SNR and correlation flag.

An LQI measurement is triggered by the assertion of *aa_sfd_matched* signal. Following this event, the module transitions to LQI measurement state. When *aa_sfd_matched* is asserted the signal *meas_complete* is deasserted (i.e. forced to low state). The blocks "RSSI averager" and "Noise averager" are enabled after a programmable delay in order to take into consideration the settling time of the IIRs (programmable delay is configured via parameter *rssi_noise_avg_delay*). Values *N* and *M* (configured through parameters *mag_downsamp_factor* and *rssi_noise_avg_factor* respectively), and the base-band sampling frequency *Fs*, directly determine the LQI measurement window duration. At the end of this window *meas_complete* is asserted and measured value *LQI* is provided, after a result is produced by the "RSSI averager" block (i.e. after *M*N* samples have been input to the RSSI/CCA/ED/LQI module since "RSSI averager" has been enabled in Figure 64-39). This signal is intended to let know the external modules consuming the outputs that LQI value is ready. The measured value of LQI is stored until the next deassertion of signal *meas_complete*. In a non-test mode, the module returns to RSSI state after asserting *meas_complete* signal, if parameter *meas_trans_to_idle* is 0 or it transitions to IDLE state if parameter *meas_trans_to_idle* is 1 (see Table 64-25).

However, the module may be configured to remain in LQI measurement state for an arbitrary period of time in a test scenario.

**Table 64-28. Examples of configuration of LQI block**

| Protocol | *N* | *M* | *Fs* [MHz] | Window duration [µs] |
|---|---|---|---|---|
| IEEE 802.15.4 | 4 | 128 | 4 | 128 |
| BLE | 8 | 32 | 8 | 32 |
| BLE | 8 | 64 | 8 | 64 |
| FSK-500kbps | 4 | 64 | 2 | 128 |
| FSK- 250kbps | 4 | 64 | 2 | 128 |




**LEGEND:**

- Signals at base-band sampling rate $F_S$ (8/4/2 MHz)
- Signals at $F_S/N$
- Signals at $2*(F_S/N)$
- Signals at symbol rate (2000/1000/500/250/62.5 kHz)
- Signals provided once per measurement
- Blocks shared between the narrowband and wideband operations

**Figure 64-40. Block diagram instantiated for narrowband mode**

#### 64.3.3.2.3.2.1 RSSI and SNR estimation

RSSI is estimated in a similar manner as described in RSSI estimator. The downsampling with *N* block is basically an accumulator. At the end of *N* accumulated values, it divides the result by *N*, outputs the result and resets the internal accumulator to 0. In the same manner function the blocks "RSSI Averager" and "SNR Averager" in Figure 64-40.

In order to estimate the SNR, a noise standard deviation estimation is needed. Noise is estimated after taking the absolute value of the subtraction of smoothed magnitude from the magnitude samples. After IIR and downsampling by *N* of magnitude and noise, conversion from linear domain to dBm is needed. In order to save die-size, "lin2dB" block is shared between the two processing paths by using a synchronous pair MUX/DEMUX. The sampling rate at the input of the "lin2dB" block is doubled due to muxing.

The signal magnitude and noise processing paths are outputting estimates of magnitude and noise respectively, expressed in dBm. The SNR is estimated by subtracting noise from signal. However, an estimation error is present due to the fact that signal and noise are estimated by averaging L1 magnitude samples and not energy values. In order to compensate for this error a non-linear SNR mapping look-up table (block "SNR Mapping LUT") is used in order to map measured SNR to more accurate SNR values. LUT values are specified in Table 64-29. These values have 0.5 dB resolution.

**Table 64-29. SNR mapping LUT**

| Input value (format (u,5,0)) | Output value (format (u, 5,1)) | Input value (format (u,5,0)) | Output value (format (u, 5,1)) |
|---|---|---|---|
| {0,1,2,...,8} | 0 | 17 | 12 |
| 9 | 1 | 18 | 13.5 |
| 10 | 3 | 19 | 15 |
| 11 | 4.5 | 20 | 16 |
| 12 | 6 | 21 | 17.5 |
| 13 | 7.5 | 22 | 19.5 |
| 14 | 8.5 | 23 | 21.5 |
| 15 | 10 | 24 | 25 |
| 16 | 11 | {25,26,27,..,31} | 26.5 |

The estimated SNR is adjustable by addition to the estimated value of the value of the parameter *snr_adjustment_factor* (signed integer) in Table 64-25.

After estimating RSSI and SNR the two quantities are passed through the "RSSI Averager" and "SNR Averager" blocks respectively.

#### 64.3.3.2.3.2.2 Correlation flag determination

During the LQI measurement window, a counter is used to count how many times the correlation products, provided at symbol rate, exceed the correlation threshold *lqi_corr_thresh*. The count is compared with threshold *corr_cnt_thresh*. *corr_cnt_thresh* takes values between 0 and the maximum number of symbols which can be received within the measurement window.

**Table 64-30. Examples of maximum number of symbols in the measurement window**

| Protocol | Symbol rate [kHz] | Window duration [µs] | Maximum *corr_cnt_thresh* |
|---|---|---|---|
| IEEE 802.15.4 | 62.5 | 128 | 8 |
| BLE | 1000 | 32 | 32 |
| FSK-500kbps | 500 | 64 | 32 |

The output of the comparator, *corr_flag*, will take value 0 if *corr_cnt_thresh* is exceeded and 1 otherwise.

#### 64.3.3.2.3.2.3 LQI calculator

Based on narrow band *RSSI* (expressed in dBm), *SNR* (dB) and *corr_flag*, *LQI_raw* is computed as described in following formula:

$$LQI_raw = (RSSI\text{-}Sensitivity)*w_{RSSI} + SNR*w_{SNR} - corr_flag*w_{CorrFlag}+LQI_{bias}$$

Configuration of parameters *Sensitivity*, $w_{RSSI}$, $w_{SNR}$ and $LQI_{bias}$ is described in Table 64-25. $w_{CorrFlag}$ is a constant and takes value 200.

After computing *LQI_raw*, the obtained value is saturated to interval [0,255] and then quantized using 8 integer bits. Thus *LQI* is determined.

### 64.3.3.2.4 I/Q Mismatch Correction

The I/Q mismatch block corrects for I/Q gain and phase mismatch. It can correct up to +/- 9 degrees of phase mismatch and +/- 4 dB of gain mismatch. After correction, phase mismatch is reduced to less than 0.5 degree and gain mismatch less than 0.3 dB.

During factory calibration, the I/Q phase adjustment coefficient (iqmc_phase_adj) and the I/Q gain adjustment coefficient (iqmc_gain_adj) are determined. Calibration is enabled by setting the iqmc_cal_en bit. The following values should be used for calibration: iqmc_num_iter=0x80, and rx_dec_filt_osr=2 (OSR of 16). The IQMC_CAL register should be set to its default value 0x0000_0400. A CW input with a 250 kHz (+/-75 kHz) offset and -2 dBm (+/- 6 dBm) level at the ADC I channel input is required

for calibration, though the calibration should also work with lower signal levels down to -18 dBm. With the standard number of iterations (iqmc_num_iter=0x80), calibration takes approximately 2 ms. After calibration is completed the IQMC_CAL register is updated and the iqmc_cal_en bit is cleared automatically. The gain calibration and phase calibration loops are shown in the figures below.

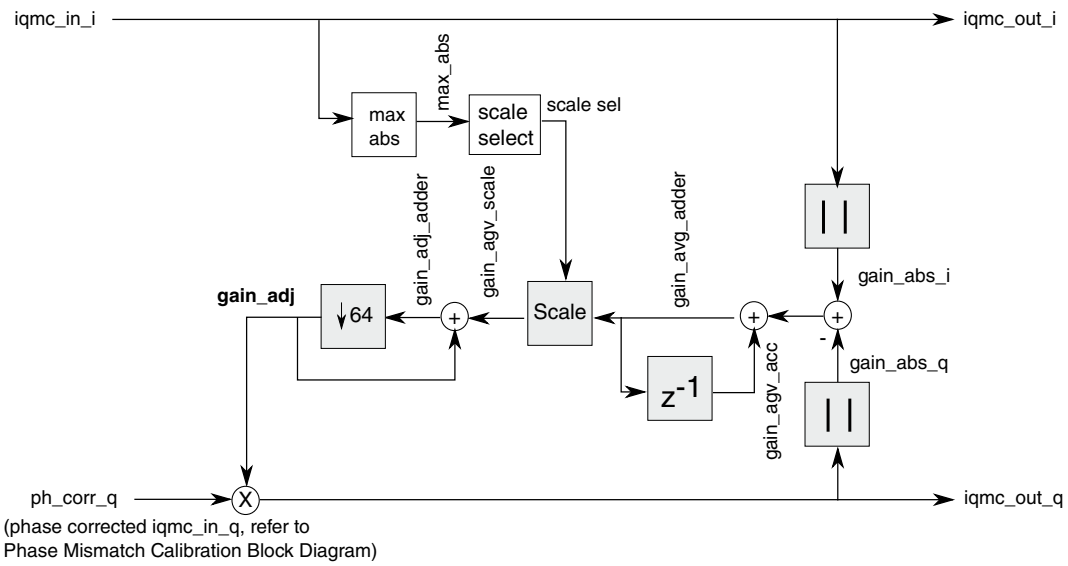



**Figure 64-41. I/Q Gain Mismatch Calibration Block Diagram**

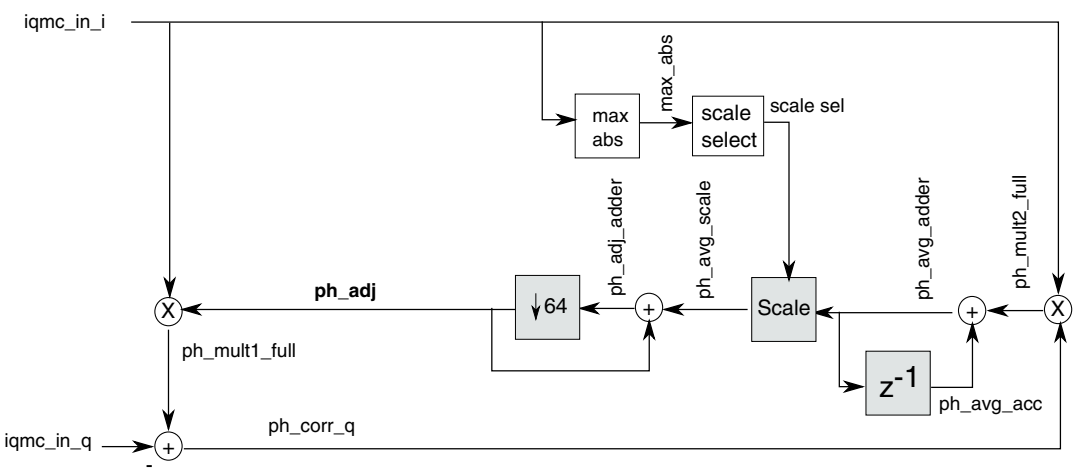



**Figure 64-42. I/Q Phase Mismatch Calibration Block Diagram**

When the receiver is enabled for packet reception, the calibration loops are disabled and the corrections are static. For gain correction, the Q channel is scaled by the gain adjustment coefficient. For phase correction, the I channel is scaled by the phase

adjustment coefficient and subtracted from the Q channel. The gain and phase correction values are stored in programming model registers and may be stored in flash to eliminate the need to run I/Q mismatch calibration after each reset.

There is also a separate gain adjusment coefficient (iqmc_dc_gain_adj) which is used only during DCOC calibration; that is, during the period after the TSM dcoc_en is asserted but before the TSM rx_dig_en is asserted. The applied phase adjustment coefficient is 0 during this time. There is no hardware calibration support to determine iqmc_dc_gain_adj.

The gain and phase corrections are illustrated in the figure below.

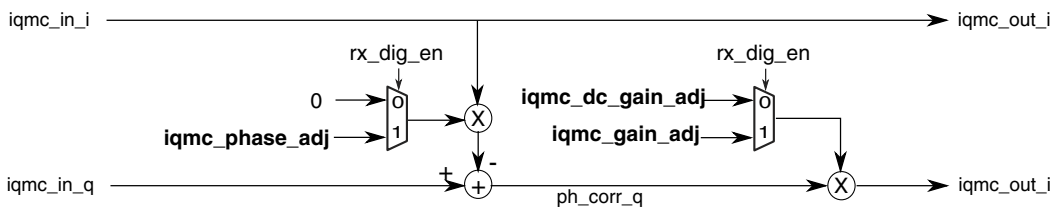



**Figure 64-43. I/Q Mismatch Correction Block Diagram**

### 64.3.3.2.5 DC Offset Correction

DC offsets in a direct conversion receiver can originate in RF (e.g., LO leakage, IP2, etc.), baseband (e.g., operational amplifier offsets, mismatches, etc.) and interfaces (due to DC level mismatches between various RF/analog circuits).

The DC (or origin) offset estimation and correction loop comprises of two distinct DC estimators. One of the estimators is used during wake-up calibration of DC offsets, while a second adaptive estimator is used during normal receiver operation. Both DC estimates are computed on the Rx I/Q data after the decimation stage. In the receiver, there are three distinct DC offset correction points, which are at the TZA output, BBA output (after first BBA active filter stage) and in the digital domain after the decimation filter. The DC estimation and correction systems operate as follows:

- A DC offset calibration is carried out at receiver warm-up to compute independent I/Q DC offsets at three distinct RX gain points.
- A DC correction is then estimated for all the entries in the AGC gain table using the 3 calibration estimates. Note that only TZA and BBA DC corrections are stored in the AGC gain lookup table.
- During AGC, the system applies a DC offset correction corresponding to the {LNA, BBA} gain combination that is applied.
- DC tracking estimates are reset upon a gain change and the algorithm is seeded with the calibrated DC estimates corresponding to the active gain setting

- Once AGC gain is settled, DC offset tracking estimation continues for a programmed amount of time
- After a pre-programmed interval expires, the tracked DC offset is sampled and a DC offset correction is estimated and applied
- Based on configuration, the DC offset tracking update can be applied once or iterated upon a programmable number of times.

A block diagram of the RX digital DCOC is shown in the figure below.

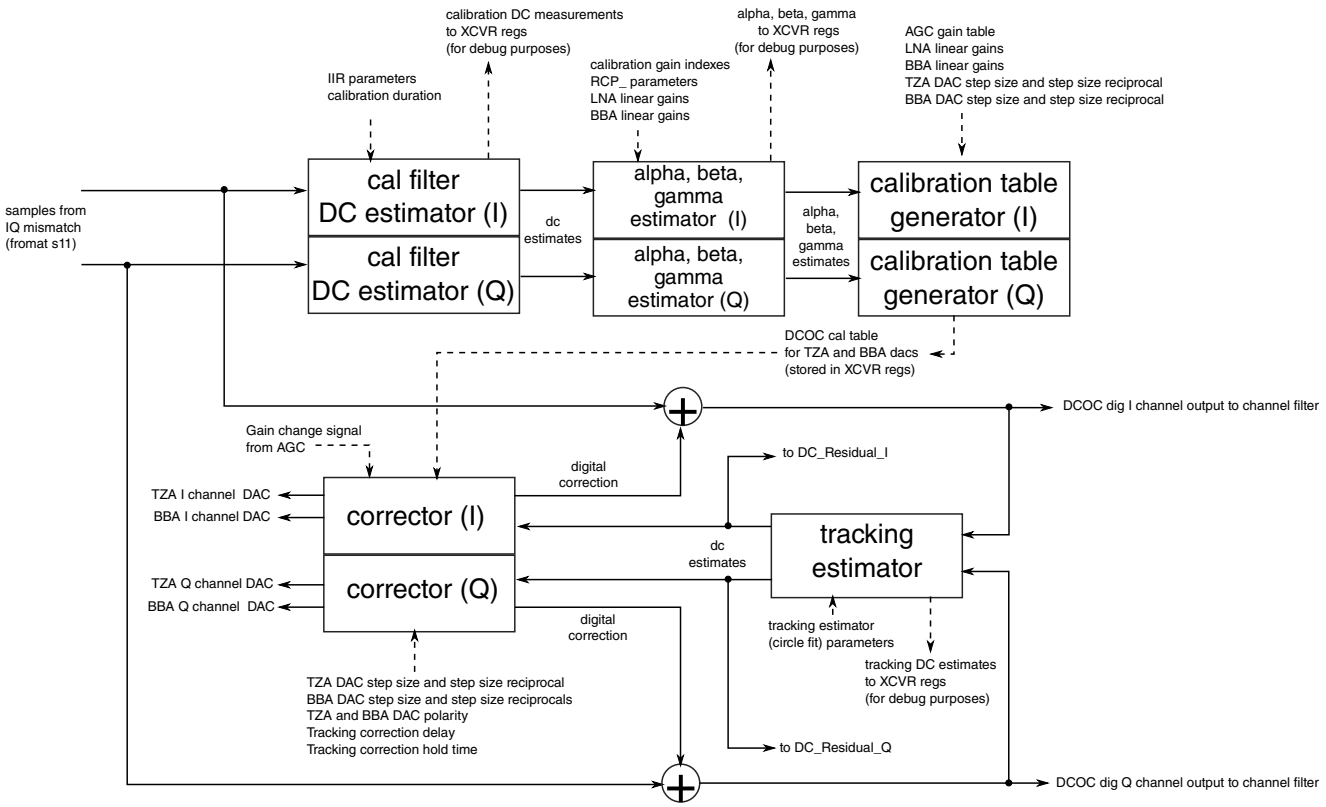



**Figure 64-44. RX_DIG DCOC Block Diagram**

### 64.3.3.2.5.1 Operating Modes

The DCOC supports several programmable bits to control the DCOC behavior. The combinations of these control bits which are supported are described in the table below.

**Table 64-31. DCOC Operating Modes**

| Mode | RX_ DCOC_EN | RX_DCOC _CAL_EN | CORRECT _EN | CORRECT _SRC | TRK_ FROM_0 | TRK_EST _OVR | MAN |
|---|---|---|---|---|---|---|---|
| Calibrate and Correct with Tracking | 1 | 1 | 1 | 1 | 0 or 1 | 0 or 1 | 0 |

*Table continues on the next page...*

**Table 64-31. DCOC Operating Modes (continued)**

| Mode | RX_ DCOC_EN | RX_DCOC _CAL_EN | CORRECT _EN | CORRECT _SRC | TRK_ FROM_0 | TRK_EST _OVR | MAN |
|---|---|---|---|---|---|---|---|
| Calibrate and Correct without Tracking | 1 | 1 | 1 | 0 | X | 0 or 1 | 0 |
| Correct with Tracking (no calibration) | 1 | 0 | 1 | 1 | 0 or 1 | 0 or 1 | 0 |
| Correct without Tracking (no calibration) | 1 | 0 | 1 | 0 | X | 0 or 1 | 0 |
| Debug: Calibration only (no correction, no tracking). | 1 | 1 | 0 | X | X | 0 or 1 | 0 |
| Debug: Manual mode (manual DAC values and DCOC_DIG_MAN corrections applied). No calibration, no correction | X | 0 | X | X | X | X | 1 |
| Debug: Manual mode (manual DAC values and DCOC_DIG_MAN corrections applied) with tracking estimator enabled. No calibration, no correction | 1 | 0 | 0 | X | 0 or 1 | 1 | 1 |
| DCOC disabled. Manual DAC values applied. No calibration, no correction | 0 | 0 | X | X | X | X | 0 |

#### 64.3.3.2.5.2 DCOC State Diagram

The main states of the DCOC are described in the bullet list and figure below.

- WAIT (IDLE)
- CALIBRATION. State where DCOC is performing calibration measurements, estimating alpha/beta/gamma, and starts writing the DCOC OFFSET table
- APPLY_IDX_CAPT. This state is also associated with calibration. DCOC transitions to this state from CALIBRATION when the table index matches the AGC's starting index. The DCOC values matching this index are applied to the DCOC DACs, and DCOC then finishes writing the rest of the DCOC OFFSET table
- NO_CAL_INIT. State where no calibration is performed, but correction is enabled. In this state, the DCOC will output the DCOC DAC values from the DCOC OFFSET table corresponding to the AGC's starting index.
- CORRECT. State where the AGC is enabled and DCOC changes the DCOC DAC values when AGC gain changes by reading from the DCOC OFFSET table.
- CORRECT_AND_TRACK. Similar to CORRECT state, but the DCOC tracking estimator is also activated, to further refine the DCOC DAC values after an AGC gain change, and also to apply a digital correction to the DCOC output.

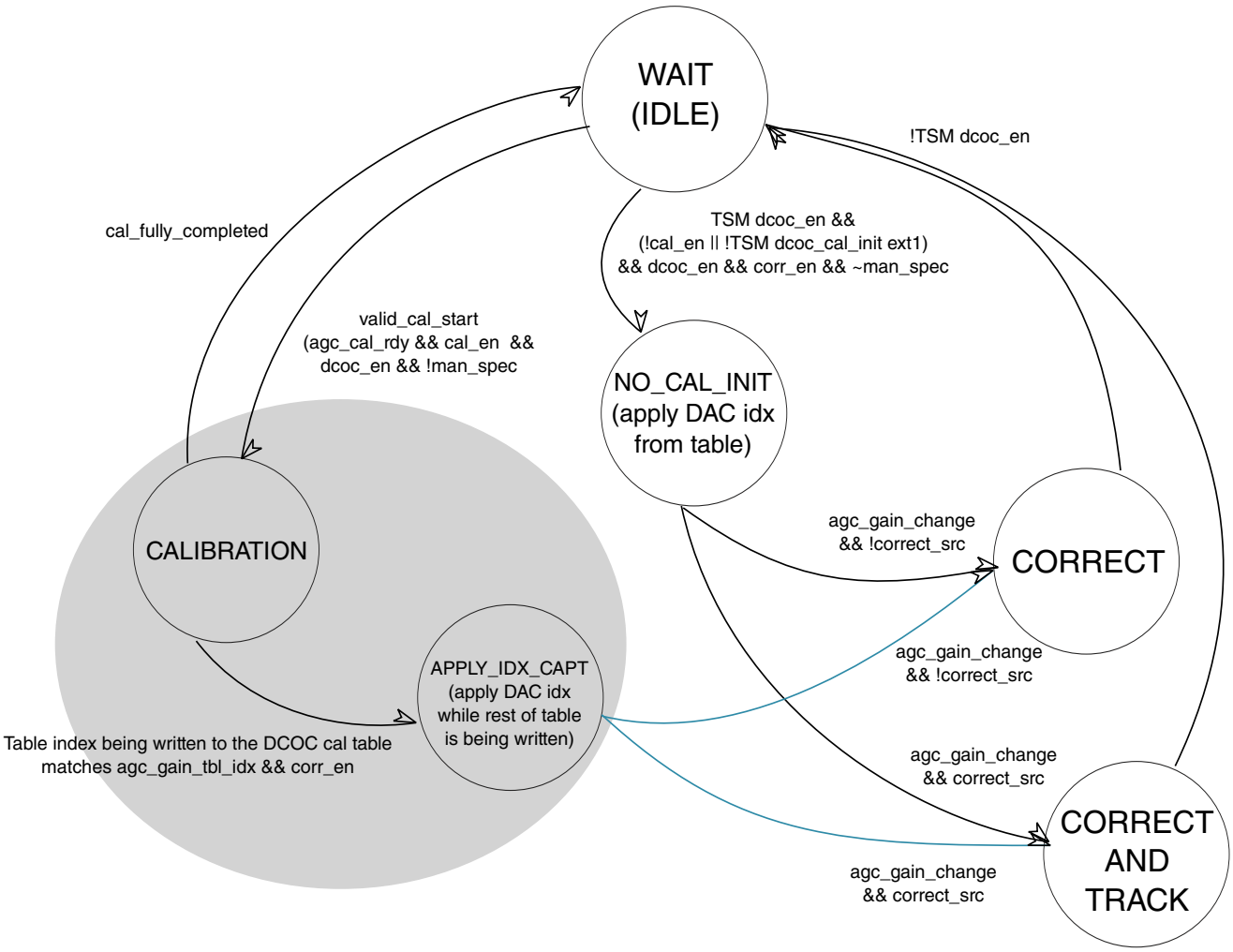



**Figure 64-45. DCOC State Diagram**

### 64.3.3.2.5.3 DC Offset Calibration

If the RX_DIG_CTRL[RX_DCOC_EN] and RX_DIG_CTRL[RX_DCOC_CAL_EN] bits are set, the direct conversion receiver will perform a DC offset calibration for I/Q receive data paths at every RX warm-up. The calibration will compute a table of DC correction values (DCOC_OFFSET_n registers) for each of the 27 AGC gain table entries.

The calibration starts when the TSM DCOC_INIT signal pulses high then low.The total calibration time includes the time for each of three DC estimates (DCOC_CAL_DURATION) and overhead to compute the DCOC_OFFSET_n table :

```
calibration time (in µs) = 3*DCOC_CTRL_0[DCOC_CAL_DURATION] + (4*27 + 15)/
reference_clock_frequency
```

A value of DCOC_CAL_DURATION=19 and a reference clock frequency of 32MHz results in a calibration time of 60.8us. The system will operate correctly as long as the calibration completes before the 1us TSM RX_INIT pulse completes.

The figure below illustrates the DC Offset Calibration timing. To allow as much time as possible for the DACs to settle before the AGC is enabled, the DCOC will initialize the DACs as soon as it calculates the DCOC table entry corresponding to the initial AGC table index. Normally, the AGC table index begins at 26 (as shown in the figure below), so the DCOC always calculates and writes the DCOC calibration table from index 26 to index 0.

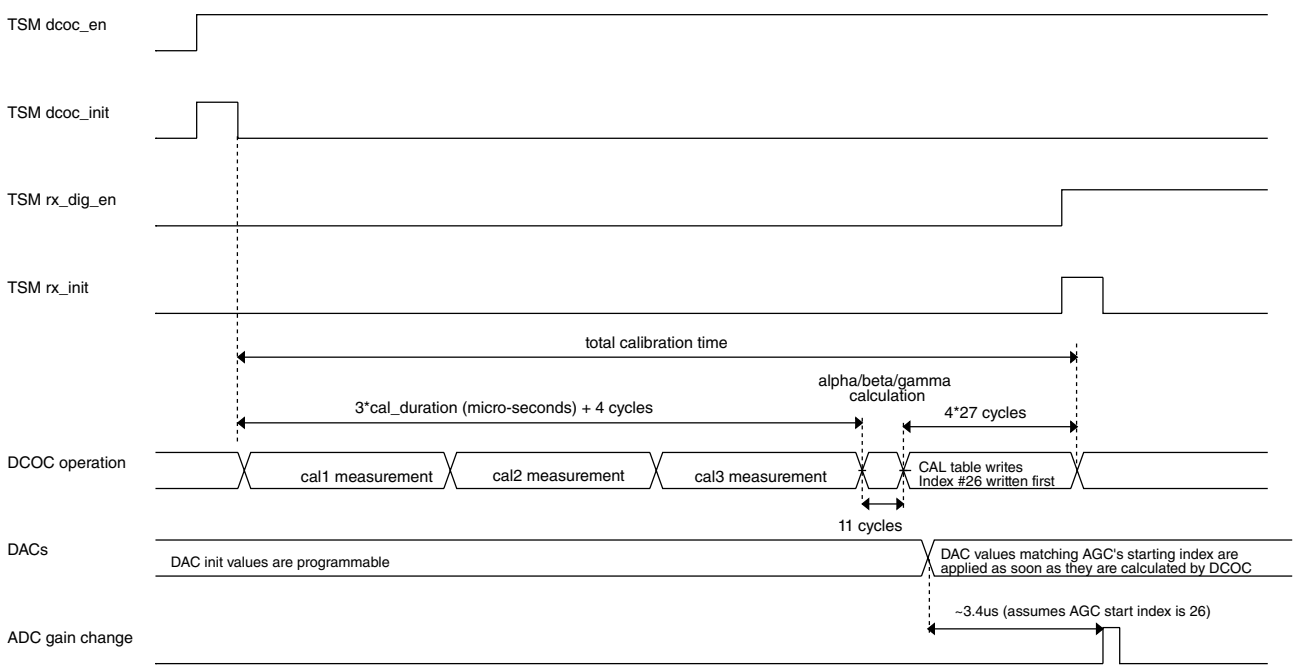



**Figure 64-46. DCOC Calibration Timing Diagram**

#### *64.3.3.2.5.3.1 DC Offset Estimation during Calibration Phase*

During the calibration phase, the DCOC makes DC offset estimates at three distinct RX LNA/BBA gain points. The three gain points are programmed in the DCOC_CAL_GAIN register, and are described more in the next section. The duration of the DC offset estimate for each of the 3 gain points is programmable up to 31us as defined by DCOC_CTRL_0[DCOC_CAL_DURATION].

DC offset estimation during calibration is carried out by using a cascade of 3 single-tap IIR filters of the type

$$y_k[n] = (1-\alpha_k)\,y_k[n-1] + \alpha_k x_k[n]$$

where $\alpha_i$ are programmable co-efficients assuming values shown below as programmed in the DCOC_CAL_IIR register:

- $\alpha_1$: {1, 1/4, 1/8, 1/16}
- $\alpha_2$: {1, 1/4, 1/8, 1/16}
- $\alpha_3$: {1/4, 1/8, 1/16, 1/32}

**Figure 64-47. DC Offset Estimation using a filter bank during calibration**

Using a cascade of 3 IIR filters as shown with $\alpha_1$=1/16; $\alpha_2$=1/16; $\alpha_3$=1/16, the transfer function of the cascaded filter is a narrowband filter with a 3dB bandwidth of approximately 170kHz and a settling time of < 18µs.

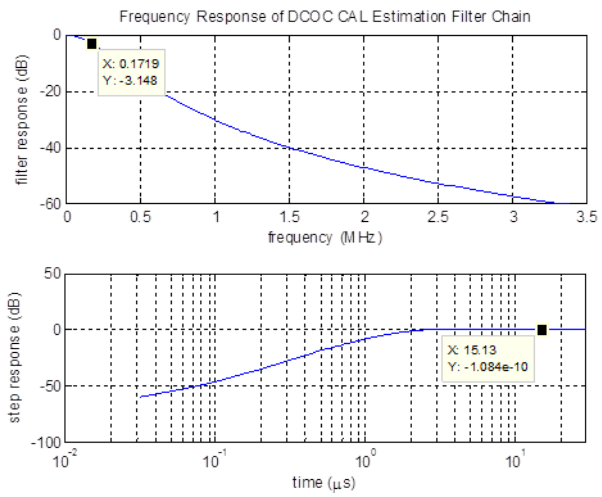



**Figure 64-48. Magnitude and Step response of DC Offset Calibration Estimator**

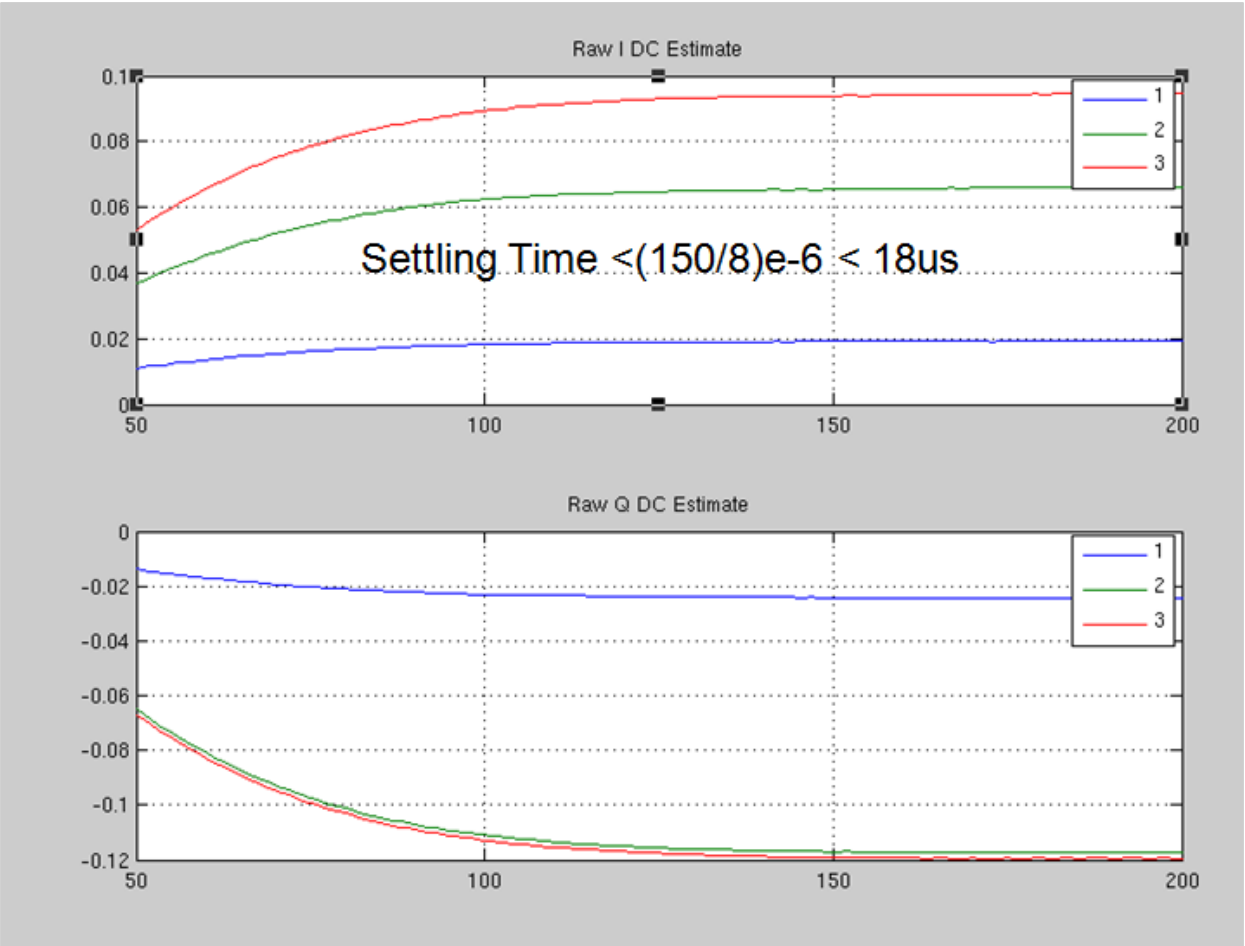



**Figure 64-49. 3-step {I,Q} DC estimation during calibration**

For debug, the DC offset estimates for the three gain points can be read after calibration from the DCOC_CAL1, DCOC_CAL2, and DCOC_CAL3 registers.

The DCOC operates on the decimated output of the ADC. As BTLE and 802.15.4 have different oversample rates, different IIR filter settings are expected to be used assuming a settling time of 20us is desired. Recommended sets of IIR1/2/3 parameters are shown below.

- Decimation rate 8: $\alpha_1 = 1/8$, $\alpha_2 = 1/16$, $\alpha_3 = 1/16$
- Decimation rate 16: $\alpha_1 = 1/4$, $\alpha_2 = 1/8$, $\alpha_3 = 1/8$

#### *64.3.3.2.5.3.2 DC Estimation at 3 operation gain points of the receiver*

The ZIF receiver performs a 3 point DC Offset calibration. The calculations shown below are shown for one branch of Rx (I or Q) only. In hardware the calculations are carried out independently for both I and Q branches and stored independently in the AGC/DCOC lookup table

The DC offset calibration scheme approximates the DC offset behavior of the system observed at three distinct gain settings by calculating the DC offset for the three gain points, and decomposing the correction into 3 DC components, which are

- α - An RF component (such as LO feed through), which scales with both LNA and BBA gains
- β - A TZA DC offset which scales with BBA gains
- γ - represents an un-scaled DC offset at the ADC input

These DC components are illustrated in the figure below.

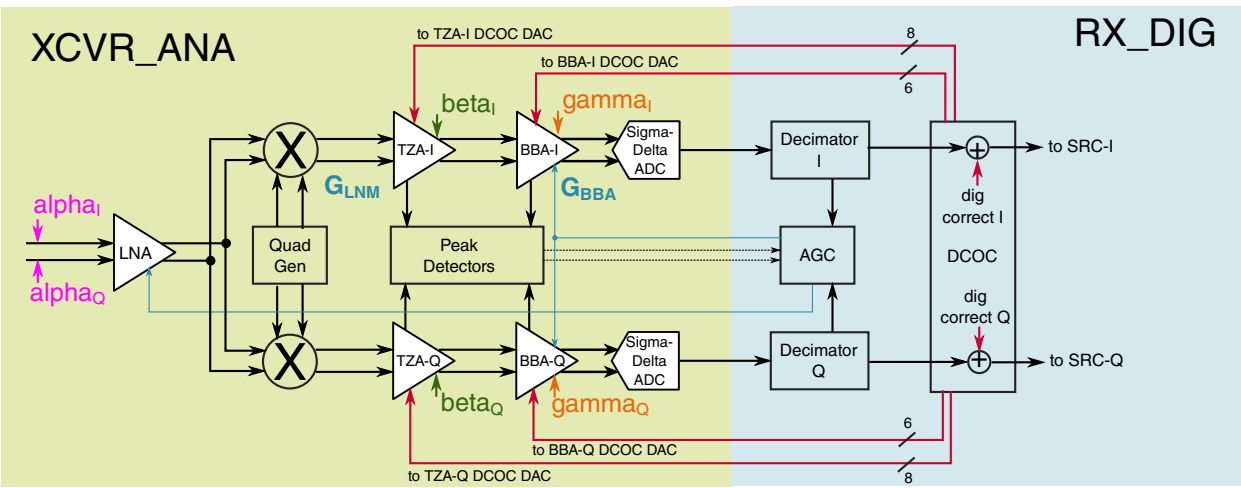



**Figure 64-50. Receiver model showing three contributors of DC offset**

The DC offset correction after calibration is applied at 2 points in the receiver, which are at

- the TZA output using an 8-bit DAC with LSB = 1.2V/2^7*(22.9kΩ/34kΩ) = 6.31mV
- the BBA output using a 6-bit DACs with LSB = 1.2V/2^5*(63.44kΩ/120kΩ) = 19.83mV

Calibration is performed open-loop. During calibration, the DCOC normally measures the DC values while the DACs are set to apply an offset of 0V. However, other DAC values can be used during calibration by using the TZA_DCOC_INIT_Q/I and BBA_DCOC_INIT_Q/I bitfields.

**NOTE**

The equations below are for the case when the TZA_DCOC_INIT_Q/I and BBA_DCOC_INIT_Q/I use their default reset values (which represent DAC values of 0). The α-hat/β-hat/γ-hat calculations in the RTL also support the scenario where non-default values are used; in that case, the α-

hat/β-hat/γ-hat are calculated such they should yield the same estimates as if DAC values of 0 were used.

The equations below provide the calculations from 3 distinct DC offset measurements to the estimation of the three decomposed DC offset components.

At any given time, the total DC offsets observed at ADC input would follow the formula

$RxDCOC_pre_ADC \cong (\alpha * G_L + \beta) * G_B + \gamma$

where

- $G_L$ – Voltage gain of the LNA stage (in linear domain)
- $G_B$ – Voltage gain of the BBA stage (in linear domain)

The ADC has an effective gain of 1code/1.7V. In the decimator, there is a decimation-rate-dependent gain factor, and also a programmable gain factor as described in Decimation Filter. Let ADC_gain represent the combined gain through the ADC and decimator in terms of codes/mV:

```
ADC_gain = 1/1700mV * (OSR^4) * (OSR_scale_factor) * RX_DEC_FILT_GAIN / 2
```

For example, if OSR is 8 (OSR_scale_factor is therefore 2^0 as shown in Table 64-22) and RX_DEC_FILT_GAIN is programmed for a gain of 1.6875, ADC_gain is 2.033codes/mV.

Note that the ADC_gain factor is a component of the value programmed for the BBA_DCOC_STEP, BBA_DCOC_STEP_RECIP, DCOC_TZA_STEP_GAIN, and DCOC_TZA_STEP_RECIP bitfields.

**NOTE**

The DCOC calibration actually performs slighty better when the ADC_gain factor shown above is multiplied by a scaling factor of ~0.96; this is due to the fixed point arithmetic implementation of the calibration algorithm. The scaling factor effectively causes the calibration algorithm to compute slightly larger TDAC and BDAC values (see below) than it would otherwise calculate.

The total DC offset observed at the RxDCOC is

$RxDCOC_post_ADC \cong ADC_gain * ((\alpha * G_L + \beta) * G_B + \gamma)$

The DC offset calibration is carried at three distinct {LNA, BBA} gain settings, as summarized below.

**Table 64-32. DC offset calibration at selected {LNA,BBA} gain settings**

| DCOC_CAL | LNA linear gain (V/V) | BBA linear gain (V/V) | DC offset Measured at ADC input |
|---|---|---|---|
| **DCOC[1]** | $G_{L_LO}$ | $G_{B_LO}$ | ADC_gain *((α * $G_{L_LO}$ + β )*$G_{B_LO}$ + γ ) |
| **DCOC[2]** | $G_{L_HI}$ | $G_{B_LO}$ | ADC_gain *((α * $G_{L_HI}$ + β )*$G_{B_LO}$ + γ ) |
| **DCOC[3]** | $G_{L_LO}$ | $G_{B_HI}$ | ADC_gain *((α * $G_{L_LO}$ + β )*$G_{B_HI}$ + γ ) |

The three gain point settings are defined by programming the DCOC_CAL_GAIN register. Here is an example:

**Table 64-33. Example DCOC_CAL_GAIN programming**

| DCOC_CAL | DCOC LNA CAL GAIN index | DCOC BBA CAL GAIN index |
|---|---|---|
| **DCOC[1]** | 3 (corresponds to $G_{L_LO}$ of 15dB) | 5 (corresponds to $G_{B_LO}$ of 15dB) |
| **DCOC[2]** | 6 (corresponds to $G_{L_HI}$ of 33dB) | 5 (corresponds to $G_{B_LO}$ of 15dB) |
| **DCOC[3]** | 3 (corresponds to $G_{L_LO}$ of 15dB) | 10 (corresponds to $G_{B_HI}$ of 30dB) |

Calculations of estimates for α , β and γ are carried out post ADC, i.e., ADC_gain is used implicitly in calculations.

Using [1] & [2]

$$\alpha\text{-hat} = \alpha * ADC_gain = (DCOC[2] - DCOC[1]) * [1.0((G_{L_HI} - G_{L_LO>}) * G_{B_LO})] \quad [4]$$

Using eq. [1] & eq. [3]

$$DCOC_tmp = (\alpha\text{-hat} * G_{L_LO} + \beta\text{-hat}) * ADC_gain$$
$$= (DCOC[3] - DCOC[1]) * [1.0/(G_{B_HI} - G_{B_LO})] \quad [5]$$

Normally, the TZA_DCOC_INIT bitfield is left at its default value of 0x80 in which case the TZA DCOC DAC applies an offset of 0V. In this case, using eq. [4] & eq. [5]

$$\beta\text{-hat} = \beta * ADC_gain = [5] - [4] * G_{L_LO} \quad [6]$$

To account for scenarios where TZA_DCOC_INIT is not 0x80, equation [6] can be generalized as shown below to subtract the contribution from the TZA DCOC DAC. In equation [6a], "tza_dac_sign" is +1 if DCOC_CTRL_0[TZA_CORR_POL] is clear, and -1 if it is set.

$$\beta\text{-hat} = \beta * ADC_gain = [5] - [4] * G_{L_LO} - (TZA_DCOC_INIT - 0x80) * TZA_dcoc_step_norm * tza_dac_sign \quad [6a]$$

Likewise, the BBA_DCOC_INIT bitfield is normally left at its default value of 0x20 in which case the BBA DCOC DAC applies an offset of 0V. In this case using eq. [5] & eq. [3]

$$\gamma\text{-hat} = \gamma * ADC_gain = [3] - [5] * G_{B_HI} \quad [7]$$

To account for where scenarios where BBA_DCOC_INIT is not 0x20, equation [7] can be generalized as shown below to subtract the contribution from the BBA DCOC DAC. In equation [7a], "bba_dac_sign" is +1 if DCOC_CTRL_0[BBA_CORR_POL] is clear, and -1 if it is set.

$$\gamma\text{-hat} = \gamma * ADC_gain = [3] - [5] * G_{B_HI} - (BBA_DCOC_INIT - 0x20) * BBA_dcoc_step * bba_dac_sign \quad [7a]$$

After calibration, α-hat, β-hat, and γ-hat can be read from the DCOC_CAL_ALPHA, DCOC_CAL_BETA, and DCOC_CAL_GAMMA registers, respectively for debug purposes.

To perform the α-hat/β-hat/γ-hat calculations, the DCOC also uses the following register bitfields:

- DCOC_CAL_RCP[ALPHA_CALC_RECIP]. This is the $1.0/((G_{L_HI}- G_{L_LO>})*G_{B_LO})$ factor from equation [4] above.
- DCOC_CAL_RCP[DCOC_TMP_CALC_RECIP]. This is the $1.0/(G_{B_HI}- G_{B_LO})$ factor from equation [5] above.
- LNA_GAIN_LIN_VAL_x (LNA_GAIN_LIN_VAL_x_y registers). The $G_{L_LO}$ and $G_{L_HI}$ gains are found from the appropriate index into this table.
- BBA_RES_TUNE_LIN_VAL_x (BBA_RES_TUNE_LIN_VAL_x_y registers). The $G_{B_LO}$ and $G_{B_HI}$ gains are found from the appropriate index into this table.
- DCOC_CTRL_0[TZA_CORR_POL] bit determines the polarity of the TZA DCOC DAC value applied during calibration.
- DCOC_CTRL_0[BBA_CORR_POL] bit determines the polarity of the BBA DCOC INIT DAC value applied during calibration.

#### *64.3.3.2.5.3.3 Calculation of DC Offset Table*

Once the estimates for α , β and γ , that is, α-hat, β-hat and γ-hat are computed, the calibrated DC offset correction values for the 27 entries in the AGC gain table (AGC_GAIN_TBL_xx_yy registers) can be calculated and stored (in the DCOC_OFFSET_n registers).

If DCOC_CTRL_0[DCOC_CAL_CHECK_EN] is set, the α-hat and β-hat estimates are checked against the thresholds in the DCOC_CAL_FAIL_TH and DCOC_CAL_PASS_TH registers as a condition of updating the DCOC_OFFSET_n registers:

- If the absolute value of all of the α-hat and β-hat estimates are less than the "pass" thresholds, the DCOC_OFFSET_n tables are updated.

- If the absolute value of any of the α-hat and β-hat estimates are greater than the "fail" thresholds, the DCOC_OFFSET_n tables are not updated.
- If neither a "pass" nor a "fail" condition is detected, the DCOC_OFFSET_n tables are only updated if a "pass" condition has not occurred since the last reset

In the cases where the DCOC_OFFSET_n table is not updated, the DCOC will use whatever values were previously written into the table to perform corrections during the burst.

The calculations are independently carried out for both the I/Q branches of the transceiver but only one branch is shown below. The computations are as follows for all k=26, 25, ... 0 entries of the AGC_GAIN_TBL, and where $TDAC_k$ and $BDAC_k$ represent the TZA and BBA DAC values:

```
DCOC_TZA_TOTAL = α-hat * G_Lk + β-hat
TDAC_k = round(DCOC_TZA_TOTAL*[1.0/TZA_dcoc_step_norm])
DCOC_TZA_RESIDUAL = DCOC_TZA_TOTAL - TDAC_k * TZA_dcoc_step_norm
DCOC_BBA_TOTAL = DCOC_TZA_residual *G_Bk + γ-hat
BDAC_k = round(DCOC_BBA_TOTAL*[1.0/BBA_dcoc_step])
```

| Information derived from the programmable AGC_GAIN_TBL_xx_yy registers | | Computed and stored in the DCOC_OFFSET_n registers | | | |
|---|---|---|---|---|---|
| **LNA linear gain (V/V)** | **BBA linear gain (V/V)** | **TZA-I DAC code** | **BBA-I DAC Code** | **TZA-Q DAC code** | **BBA-Q DAC Code** |
| $G_{L26}$ | $G_{B26}$ | $TDAC_{I26}$ | $BDAC_{I26}$ | $TDAC_{Q26}$ | $BDAC_{Q26}$ |
| … | … | … | … | … | … |
| $G_{L0}$ | $G_{B0}$ | $TDAC_{I0}$ | $BDAC_{I0}$ | $TDAC_{Q0}$ | $BDAC_{Q0}$ |

Note that the values stored in the DCOC_OFFSET_n table represent the DC offset at the DAC. Normally the DCOC will then apply the negative of this DC offset to the DAC when it applies the correction, but this depends on the DCOC_CTRL_0[TZA_CORR_POL] and DCOC_CTRL_0[BBA_CORR_POL] bits. Also, note that the DAC code values themselves (which are reflected in the DCOC_STAT register) include a bias offset of 0x80 for the TZA DACs and 0x20 for the BBA DACs.

To compute the TDAC and BDAC values for the DCOC_OFFSET_n registers, the DCOC uses the following register bitfields:

- DCOC_BBA_STEP[BBA_DCOC_STEP_RECIP]. This is the 1.0/BBA_dcoc_step referred to above.
- DCOC_TZA_STEP_0[DCOC_TZA_STEP_RCP]. This is the 1.0/ TZA_dcoc_step_norm referred to above.
- DCOC_TZA_STEP_0[DCOC_TZA_STEP]. This is the TZA_dcoc_step_norm referred to above.

- LNA_GAIN_LIN_VAL_x (LNA_GAIN_LIN_VAL_x_y registers). This is used for the $G_{Lk}$ referred to above
- BBA_RES_TUNE_LIN_VAL_x (BBA_RES_TUNE_LIN_VAL_x_y registers). This is used for the $G_{Bk}$ referred to above
- RF_TZA[RX_TZA_SPARE]. The RX_TZA_SPARE[2] bit specifies the allowed range of values for the TZA DCOC DAC.
- DCOC_CTRL_0[DCOC_CAL_CHECK_EN]. Determines whether calibration check logic is enabled or not. If it is, then the pass and fail threshold registers are used:
  - DCOC_CAL_PASS_TH[DCOC_CAL_ALPHA_P_TH]. Alpha pass threshold
  - DCOC_CAL_PASS_TH[DCOC_CAL_BETA_P_TH]. Beta pass threshold
  - DCOC_CAL_FAIL_TH[DCOC_CAL_ALPHA_F_TH]. Alpha fail threshold
  - DCOC_CAL_FAIL_TH[DCOC_CAL_BETA_F_TH]. Beta fail threshold

#### *64.3.3.2.5.3.4 Use of DC offset Table After Calibration*

After calibration, when the TSM RX_DIG_EN is asserted, if the DCOC_CTRL_0[DCOC_CORRECT_EN] bit is 1, the DCOC will apply the value stored in the DCOC_OFFSET_n table on each AGC gain change. As indicated previously, the value stored in the DCOC_OFFSET_n register represents the DAC vaue of the DC offset.

If the DCOC_CTRL_0[TZA_CORR_POL] bit is clear (normal polarity), the DCOC will subtract the TDAC value from the TZA DAC zero bias value of 0x80. If the TZA_CORR_POL bit is set, the DCOC will add the TDAC value to the TZA DAC zero bias value.

If the DCOC_CTRL_0[BBA_CORR_POL] bit is clear (normal polarity), the DCOC will subtract the BDAC value from the BBA DAC zero bias value of 0x20. If the BBA_CORR_POL bit is set, the DCOC will add the BDAC value to the BBA DAC zero bias value.

The DCOC also allows software to write to the DCOC_OFFSET_n table. In this case, the RX_DIG_CTRL[RX_DCOC_CAL_EN] bit should be cleared so that the DCOC will not overwrite the DCOC_OFFSET_n table, but DCOC_CTRL_0[DCOC_CORRECT_EN] bit should still be set.

### 64.3.3.2.5.4 DC Offset Tracking

The DCOC also supports a closed-loop DC tracking estimator which can be used with or without DC Offset calibration. The tracking estimator applies a DAC correction and also applies a digital correction to the downsampled output of the ADC.

The tracker is used if the DCOC_CTRL_0[DCOC_CORRECT_SRC] bit is programmed to 1 (RX_DIG_CTRL[RX_DCOC_EN] and DCOC_CTRL_0[DCOC_CORRECT_EN] should also be programmed to 1). The tracking estimator operation is as follows:

- On every AGC gain change, the DCOC will apply the DC offset correction to the DACs from the DCOC_OFFSET_n table, and resets the digital correction to zero.
- If the AGC gain table index is larger than or equal to DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX], then...
  - After the AGC gain settles (refer to AGC_CTRL_1[LNA_GAIN_SETTLE_TIME] and AGC_CTRL_2[BBA_GAIN_SETTLE_TIME] bitfields) , the tracker's estimate is reset to either 0 or the current I/Q sample, depending on programming of the DCOC_CTRL_0[TRACK_FROM_ZERO] bit.
  - The DCOC will periodically adjust the DACs and digital correction at intervals defined by the DCOC_CTRL_0[DCOC_CORR_DLY] bitfield until a timeout defined by the DCOC_CTRL_0[DCOC_CORR_HOLD_TIME] bitfield is reached.
  - The tracking estimator supports two set of parameters so that estimator can be gearshifted from a high bandwidth to a narrow bandwidth configuration after a number of update corrections defined by DCOC_CTRL_1[DCOC_TRK_EST_GS_CNT].
  - If the DCOC_CTRL_0[DCOC_MIDPWR_TRK_DIS] bit is set, the tracking corrections are disabled if either the TZA or BBA lo peak detector asserts. This is implemented by resetting the counters associated with DCOC_CORR_DLY and DCOC_HOLD_TIME, so if the lo peak deters stop asserting, then tracking corrections woud be resumed. The counter associated with DCOC_TRK_EST_GS_CNT is not affected by the DCOC_MIDPWR_TRK_DIS bit.
- If DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX_CFG] is set and the AGC gain table index is smaller than DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX], then...
  - After the AGC gain settles (refer to AGC_CTRL_1[LNA_GAIN_SETTLE_TIME] and AGC_CTRL_2[BBA_GAIN_SETTLE_TIME] bitfields) , the tracker's estimate is reset to either 0 or the current I/Q sample, depending on programming of the DCOC_CTRL_0[TRACK_FROM_ZERO] bit.
  - The DCOC will periodically adjust the DACs and digital correction at intervals defined by the DCOC_CTRL_2[DCOC_CORR_DLY_ALT] bitfield until a timeout defined by the DCOC_CTRL_2[DCOC_CORR_HOLD_TIME_ALT] bitfield is reached.
  - The tracking estimator supports two set of parameters so that estimator can be gearshifted from a high bandwidth to a narrow bandwidth configuration after a number of update corrections defined by DCOC_CTRL_1[DCOC_TRK_EST_GS_CNT].
  - If the DCOC_CTRL_0[DCOC_MIDPWR_TRK_DIS] bit is set, the tracking corrections are disabled if either the TZA or BBA lo peak detector asserts. This

is implemented by resetting the counters associated with DCOC_CORR_DLY and DCOC_HOLD_TIME, so if the lo peak deters stop asserting, then tracking corrections woud be resumed. The counter associated with DCOC_TRK_EST_GS_CNT is not affected by the DCOC_MIDPWR_TRK_DIS bit.

- If DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX_CFG] is clear and the AGC gain table index is smaller than DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX], then tracking is not activated for this gain setting.

The figure below illustrates the timing behavior during DC offset tracking correction after an AGC gain change. If the DCOC_CTRL_0[DCOC_CORRECT_SRC] bit were programmed to 0 instead of 1, only the DCOC DAC change at agc_gain_chng pulse would occur and the digital correction applies to the signal through the DCOC would remain at 0. The figure shows an example where DCOC_TRK_EST_GS_CNT=2, so the tracking estimator changes its parameters after 2 update correction periods.

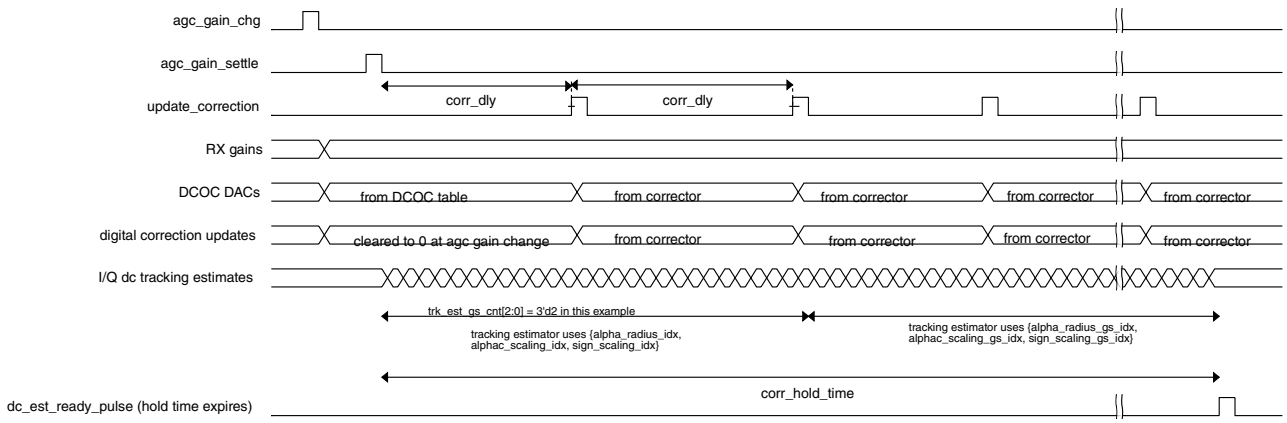



**Figure 64-51. DC Offset Tracking Correction Timing**

#### *64.3.3.2.5.4.1 DC Offset Estimation during Tracking Phase*

There are two distinct DC offset estimators: the one use during calibration described previously, and one used for DC tracking during normal receiver operation.

DC Offset estimation during normal receiver operation requires estimation of DC signal in the presence of a constant envelope modulation signal. A circle-fit type of algorithm is used for DC estimation in the presence of desired as well as interfering signals.

The circle-fit algorithm relies on the programmable bitfields shown in the list below. Note that there are two sets of alpha_radius, alphac_scale, and sign_scale parameters, which allows the algorithm to gearshift from a high-bandwidth configuration

{alpha_radius_idx, alphac_scale_idx, sign_scale_idx} to a low-bandwidth configuration {alpha_radius_gs_idx, alphac_scale_gs_idx, sign_scale_gs_idx} after the number of update corrections defined by DCOC_TRK_EST_GS_CNT.

- DCOC_CTRL_1[DCOC_ALPHA_RADIUS_IDX] and DCOC_CTRL_1[DCOC_ALPHA_RADIUS_GS_IDX]. Step size for radius, assumes values of {1, 1/2, 1/4, 1/8, 1/16, 1/32, or 1/64}. In pseudo-code below, this is referred to as "alpha_radius".
- DCOC_CTRL_1[DCOC_ALPHAC_SCALE_IDX] and DCOC_CTRL_1[DCOC_ALPHAC_SCALE_GS_IDX]. Step size for I/Q center, assumes values of {1/2, 1/4, 1/8, 1/16, 1/32 or 1/64}. In pseudo-code below, this value times the "alpha_radius" value is referred to as "alpha_center"
- DCOC_CTRL_1[DCOC_SIGN_SCALE_IDX] and DCOC_CTRL_1[DCOC_SIGN_SCALE_GS_IDX]. sign_scaling. Scaling factor for sign() based correction of the I/Q center, assumes values of {1/8, 1/16, 1/32, or 1/64}. In psuedo-code below, this is referred to as "sign_scaling".
- DCOC_CTRL_1[DCOC_TRK_EST_GS_CNT]. If this is 0, then the algorithm only uses the {alpha_radius_idx, alphac_scale_idx, sign_scale_idx} parameters above. Otherwise, the algorithm begins with the {alpha_radius_idx, alphac_scale_idx, sign_scale_idx} parameters after each AGC gain change, but then switches to the 2nd parameter set {alpha_radius_gs_idx, alphac_scale_gs_idx, sign_scale_gs_idx} after DCOC_TRK_EST_GS_CNT number of update correction periods.
- DCOC_CTRL_0[TRACK_FROM_ZERO]. This control bit determines the initial value used by the tracking estimator on each AGC gain change. If TRACK_FROM_ZERO=1, the tracking estimate starts from I/Qcenter=0; if TRACK_FROM_ZERO=0, the tracking estimator starts from I/Qcenter using the current I/Q sample. TRACK_FROM_ZERO=1 can be used in conjunction with calibration, whereas TRACK_FROM_ZERO=0 can be used with or without calibration and also works well at a wider range of RSSI levesl.

Pseudo-code for the circle-fit algorithm in the tracking estimator is as follows:

```
epsilon_I = I - Icenter;
epsilon_Q = Q - Qcenter;
i_abs = abs(epsilon_I);
q_abs = abs(epsilon_Q);
delta_radius = max(i_abs,q_abs)+min(i_abs,q_abs)*3/8;
delta_Icenter = I - Radius*sign(epsilon_I)*sign_scaling;
delta_Qcenter = Q - Radius*sign(epsilon_Q)*sign_scaling;
Icenter = alpha_center*delta_Icenter + (1 - alpha_center)*Icenter;
Qcenter = alpha_center*delta_Qcenter + (1 - alpha_center)*Qcenter;
Radius = alpha_radius*delta_radius + (1 - alpha_radius)*Radius;
```

For debug, the tracker's DC estimate (Icenter, Qcenter) can be read from the DCOC_DC_EST register.

The DC correction applied by the tracking algorithm makes use of the following register bitfields:

- DCOC_BBA_STEP[BBA_DCOC_STEP].
- DCOC_BBA_STEP[BBA_DCOC_STEP_RECIP].
- DCOC_TZA_STEPn[TZA_DCOC_STEP], where n=0..10.
- DCOC_TZA_STEPn[TZA_DCOC_STEP_RECIP], where n=0..10.
- DCOC_CTRL_0[TZA_CORR_POL]
- DCOC_CTRL_0[BBA_CORR_POL]
- RF_TZA[RX_TZA_SPARE]. The RX_TZA_SPARE[2] bit specifies the allowed range of values for the TZA DCOC DAC.

Normally, the tracking estimator is only enabled when it is needed by the corrector. However, by setting the DCOC_CTRL_0[TRK_EST_OVR] bit, the tracking estimator can be enabled continuously while the DCOC is active.

#### 64.3.3.2.5.5 DC Offset Manual Mode

For debug purposes, the DCOC supports a manual mode. This mode is enabled when the DCOC_CTRL_0[DCOC_MAN] bit is set, which overrides all other DCOC programmable control bits. In this mode, the DCOC DAC values are applied from DCOC_DAC_INIT[TZA_DCOC_INIT_Q], DCOC_DAC_INIT[TZA_DCOC_INIT_I], DCOC_DAC_INIT[BBA_DCOC_INIT_Q], and DCOC_DAC_INIT[BBA_DCOC_INIT_Q], and the digital correction is applied from DCOC_DIG_MAN[DIG_DCOC_INIT_Q] and DCOC_DIG_MAN[DIG_DCOC_INIT_I].

### 64.3.3.2.6 Channel Filter

The channel filter is a programmable, 24 tap symmetric FIR filter with 6-bit to 10-bit coefficients. The channel filter can also be configured to operate as a 16 tap filter by setting RX_DIG_CTRL[RX_CH_FILT_LEN], in which case only coefficients 4-11 are used. It operates at the decimated sample rate on the output from the DCOC. The coefficient values are programmable and should be configured for the mode of operation (BTLE, 802.15.4, Generic LL). The filter output is scaled by 1/256; the sum of the 12 coefficients programmed is expected to be <=256 in order to avoid saturation. A few example coefficient filter sets are given in the table below.

**Table 64-34. Channel Filter Coefficients**

| Link Layer | XO MHz | COEF 0 | COEF 1 | COEF 2 | COEF 3 | COEF 4 | COEF 5 | COEF 6 | COEF 7 | COEF 8 | COEF 9 | COEF 10 | COEF 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 802.15.4 | 32 | -2 | 0 | 8 | 2 | -19 | -10 | 34 | 31 | -56 | -87 | 71 | 282 |
| BLE | 32 | -2 | -5 | -3 | 7 | 19 | 14 | -16 | -51 | -49 | 16 | 122 | 204 |
| BLE[1] | 26 | -2 | -2 | 2 | 12 | 15 | 1 | -28 | -50 | -33 | 31 | 122 | 187 |

1. BLE at 26MHz configuration: OSR 12, SRC configured for ZOH 12/13 mode

#### 64.3.3.2.7 Sample Rate Conversion (SRC) Filter

The sample rate conversion block is used for a fractional clock rate conversion of the sampled decimated signal, when 26MHz reference clock is used. In this mode, the ADC is clocked at 2xFref or 52 MHz. For the 2.4GHz targeted modulations, the PHY data rate is always 4x the symbol rate (of 2Mbps, 1Mbps, 500kbps or 250kbps) for (G)FSK modes and 4x the 2Mchips/sec rate for IEEE 802.15.4. These desired frequencies are not related to the ADC clock rate by an integer factor, which necessitates the need for a fractional sample rate conversion that is performed by SRC. For a complete list of the fractional conversion modes refer to Table 64-21. The SRC is enabled by setting RX_DIG_CTRL[RX_SRC_EN] to 1, and the conversion factor is selected by RX_DIG_CTRL[RX_SRC_RATE] as described below.

##### 64.3.3.2.7.1 Sample Rate Conversion by a factor of 8/13

If RX_DIG_CTRL[RX_SRC_RATE] = 0, the SRC block is configured to perform a data rate conversion by a factor of 8/13, i.e., for every 13 input samples, the SRC block generates 8 output samples at a reduced data rate. A linearly interpolated first order hold (FOH) resampler is used for implementation as it provides significant better anti-aliasing as compared to a zero order hold (ZOH).

##### 64.3.3.2.7.2 Sample Rate Conversion by a factor of 12/13

If RX_DIG_CTRL[RX_SRC_RATE] = 1, the SRC block is configuration a zero-order hold sample rate conversion is used. This mode is realized by skipping one out of every 13 input samples.

#### 64.3.3.2.8 DC Residual Correction

This block is used to compensate for the DC offset which is still present after AGC and DCOC calibration/tracking has completed. It estimates the DC offset by measuring the minimum and maximum value and then applies an additive correction factor to the I and Q channels. This block does not change the value of the DCOC DACs.

The DC offset is compensated independently on I and Q channels. The correction loop determines the minimum and maximum values over a specified window of samples (*Nwin*) which are then averaged and used (with an update parameter *Alpha*) to create the DC estimate. This DC estimate is then subtracted from the input samples. The applied DC estimate continues to be updated every *NWIN* samples until *IT_FREEZE* of these windows have elapsed, at which time the DC estimate is frozen.

Multiple modes of operations can be identified for residual DC offset compensation block. They are presented in table below.

**Table 64-35. Residual DC offset compensation operational modes**

| Mode | Description | DC_RESID_EN | EXT_DC_EN | IT_FREEZE | DLY | NWIN, ALPHA |
|---|---|---|---|---|---|---|
| **Bypass** | The DC residual is disabled. The input I and Q samples to the DC Residual block are passed directly to the DC Redisdual output without any modifications.<br>. | 0 | - | - | - | - |
| **Mode 1** | In this mode, DC residual has the role of calculating and applying a DC offset without use of an initial estimate from the DCOC.<br>The DC residual processing is restarted at every AGC gain change<br>This mode is suitable for use when an AGC gain change occurs which is not followed by DCOC tracking. | 1 | 0 | >1 | >0 | program as needed |
| **Mode 2** | In this mode, DC residual has the role of refining the DC offset estimate provided by the DCOC at the end of the DCOC tracking after the last DCOC DAC update.<br>The DC residual processing is restarted whenever DCOC's tracking hold timer expires.<br>This mode is suitable after AGC gain change followed by DCOC tracking. | 1 | 1 | >1 | - | program as needed |
| **Mode 3** | In this mode, DC residual has the role of applying a correction based on the DC offset estimate provided by the DCOC at the end of the DCOC tracking after the last DCOC DAC update.<br>The DC residual processing is restarted whenever DCOC's tracking hold timer expires.<br>This mode is suitable after AGC gain change and DCOC tracking when no further refining is desired. | 1 | 1 | 1 | - | - |

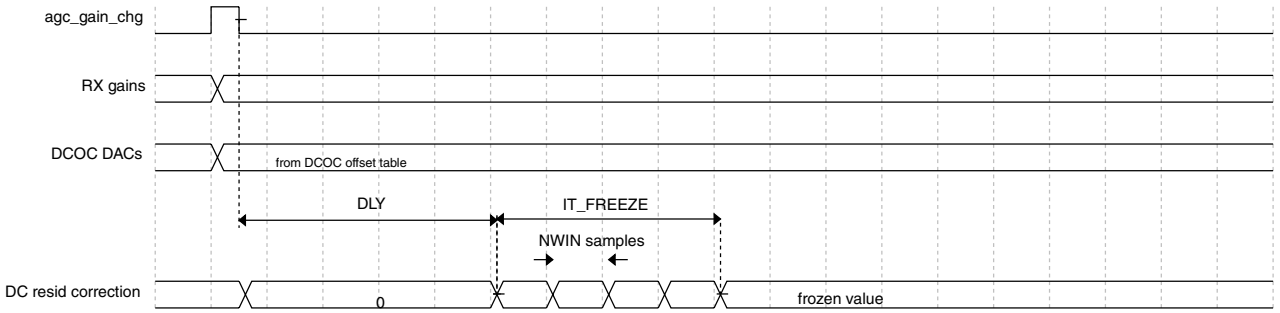



**Figure 64-52. DC residual mode 1 timing**

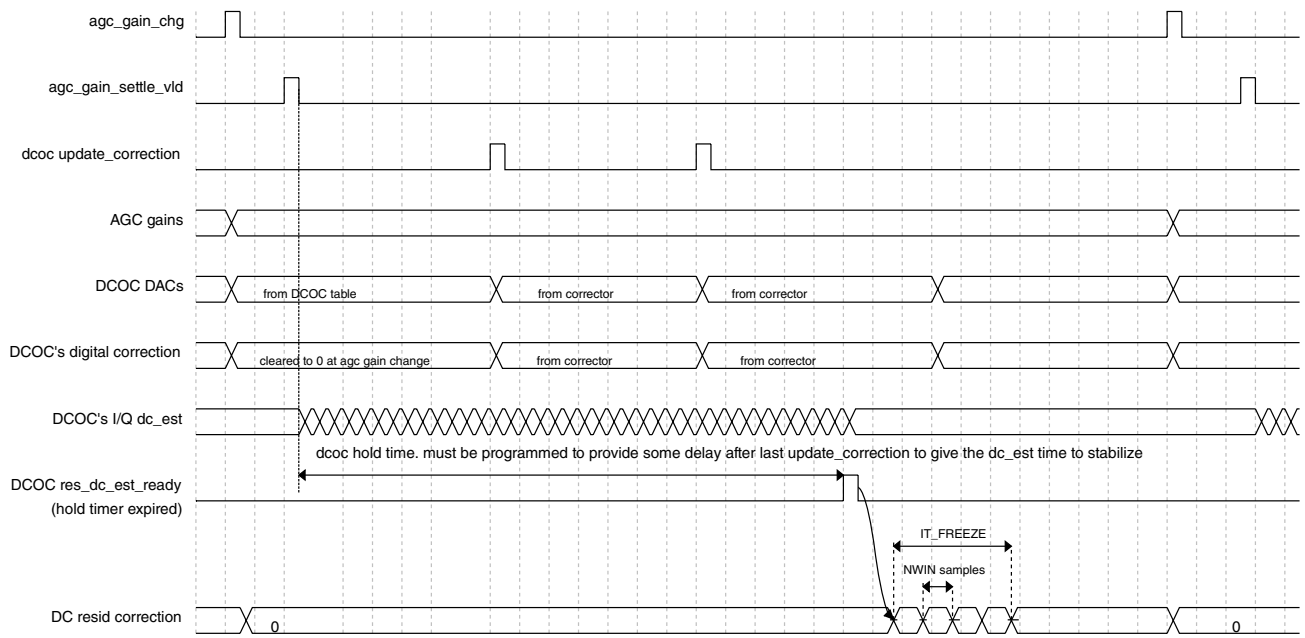



**Figure 64-53. DC residual mode 2 timing**

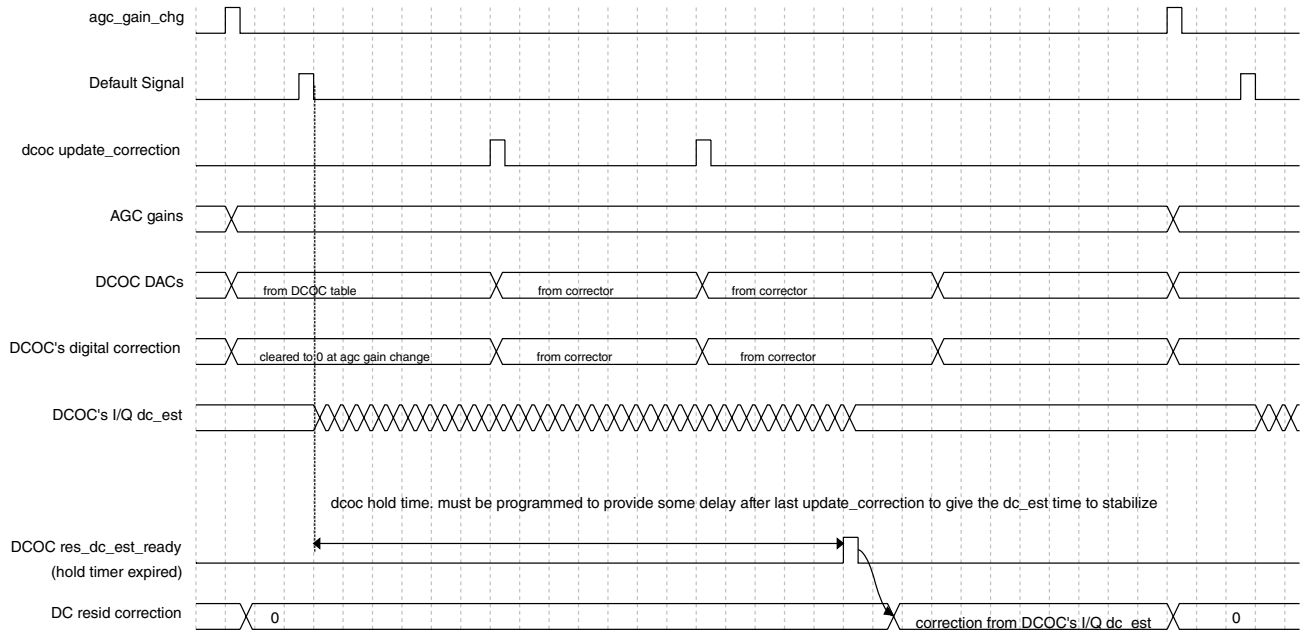



**Figure 64-54. DC residual mode 3 timing**

The behavior of the Resdual DC block is also dependent on the AGC index and the values programmed for DC_RESID's MIN_AGC_IDX and the DCOC's DCOC_CTRL_1[DCOC_TRK_MIN_AGC_IDX]. This behavior is defined in the table below, which assumes DC_RESID_EN=1.

**Table 64-36. Residual DC offset compensation operation dependency on AGC table index**

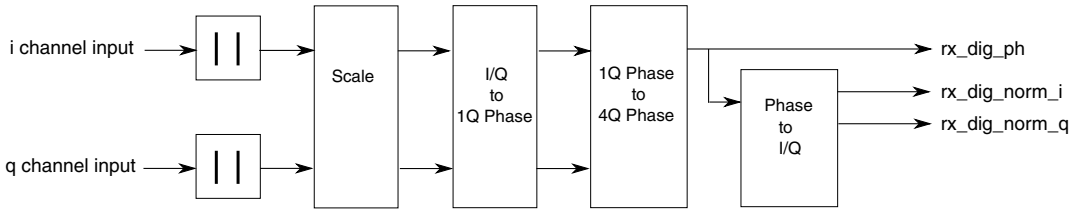
| EXT_DC_EN | MIN_AGC_IDX | DCOC_TRK_MIN_AGC_IDX | Behavior |
|---|---|---|---|
| - | AGC_IDX < MIN_AGC_IDX | - | DC residual is disabled |
| 0 | AGC_IDX >= MIN_AGC_IDX | - | DC residual operates in Mode 1 (it does not use an initial estimate from the DCOC) |
| 1 | | AGC_IDX >= DCOC_TRK_MIN_AGC_IDX | DC residual operates in Mode 2 or 3 (It uses the estimate from the DCOC) |
| | | AGC_IDX < DCOC_TRK_MIN_AGC_IDX | DC residual operates in Mode 1. it does not use the estimate from the DCOC, since the DCOC's tracker is disabled for this AGC gain index |

## 64.3.3.2.9 Normalizer

The normalizer block converts the 12-bit I/Q input signal into a normalized 5-bit I/Q output and a 5-bit phase output as shown in the block diagram below.

**NOTE**

The normalizer's 5-bit I/Q output is currently not used.




Normalizer Block Diagram

**Figure 64-55. Normalizer Block Diagram**

The absolute value of each component of the I/Q input is taken and shifted by a common factor to yield the largest 5-bit values which have a zero MSB for both I and Q. This is followed by rounding that yields I and Q values between 0 and 16 decimal. The scaled I/Q values are used as inputs to a one quadrant phase lookup table. The one quadrant table output is converted to a 4 quadrant 5-bit phase value based on the input sign bits. The 5-bit I/Q output is generated by another lookup table. The normalizer has a maximum quantization error of 8.9 degrees and a RMS error of 3.5 degrees for a random input.

Since the normalizer's phase output can be unreliable at very low signal levels, the normalizer includes a feature which, if enabled by setting RX_DIG_CTRL[RX_NORM_SUPP_EN], suppresses the rx_dig_ph output (sets it to 0) if none of the least last 6 I or Q input samples had an absolute value above a programmable threshold. The threshold is configured via RX_DIG_CTRL[RX_NORM_SUPP_TH].

### 64.3.3.2.10 AuxPLL Frequency Calibration

The AuxPLL Frequency Calibration module finds the best calibration code associated with the AuxPLL. Calibration is initiated using a TSM control signal, and it is expected that this calibration will be performed on each RX warmup. Features of the module include the following:

- Supports two different calibration run window lengths: 64 reference clock cycles, and 128 reference clock cycles. For a 32Mhz reference clock, the overall calibration time is <17us for a run window of 64 reference clock cycles, and <31us for a run window of 128 reference clock cycles.
- Calibration logic can be disabled and a manual calibration value can be applied
- The clk cycle at which the fcal_count[9:0] input from the analog is sampled is programmable.

The module uses a binary search to find the best 7bit calibration code (auxpll_fco_dac_cal_adjust[6:0]). At the end of each calibration stage, the auxpll_fcal_count[9:0] value output from the analog (which counts the number of positive and negative edges of the auxpll clock during the run window) is sampled and compared against the expected count (256 for a run window length of 64 clk cycles, 512 for a run window length of 128 clk cycles), and a decision is made whether the appropriate auxpll_vco_dac_cal_adjust[6:0] bit should be set or cleared. The decision logic is programmable:

- AUXPLL_FCAL_CTRL[FCAL_COMP_INV]=1'b0 (default behavior): if the fcal_count is larger than or equal to the expected count, the cal_adjust bit remains set.
- AUXPLL_FCAL_CTRL[FCAL_COMP_INV]=1'b1: if the fcal_count is less than or equal to the expected count, the cal_adjust bit remains set.

After each of the seven calibration cycles, the current count error is compared to the best (smallest absolute) count error found so far, and after the binary search is complete, the module outputs the calibration code associated with the smallest absolute count error which was found.

Debug features include the following:

- Count value for each of the several calibration stages are saved to registers
- The best absolute count error found during calibration is saved to a register
- Calibration logic can be disabled and a manual calibration value can be applied

The figure below shows the beginning of the AuxPLL calibration timing. For a run window length of 64 clk cycles (which is the configuration expected to be used) each of the 7 calibration stages requires ~76clks, with total calibration time of ~532 clks, which is <17us at 32MHz. Only the 1st (and the beinning of the 2nd) of the 7 calibration stages are shown in the figure.

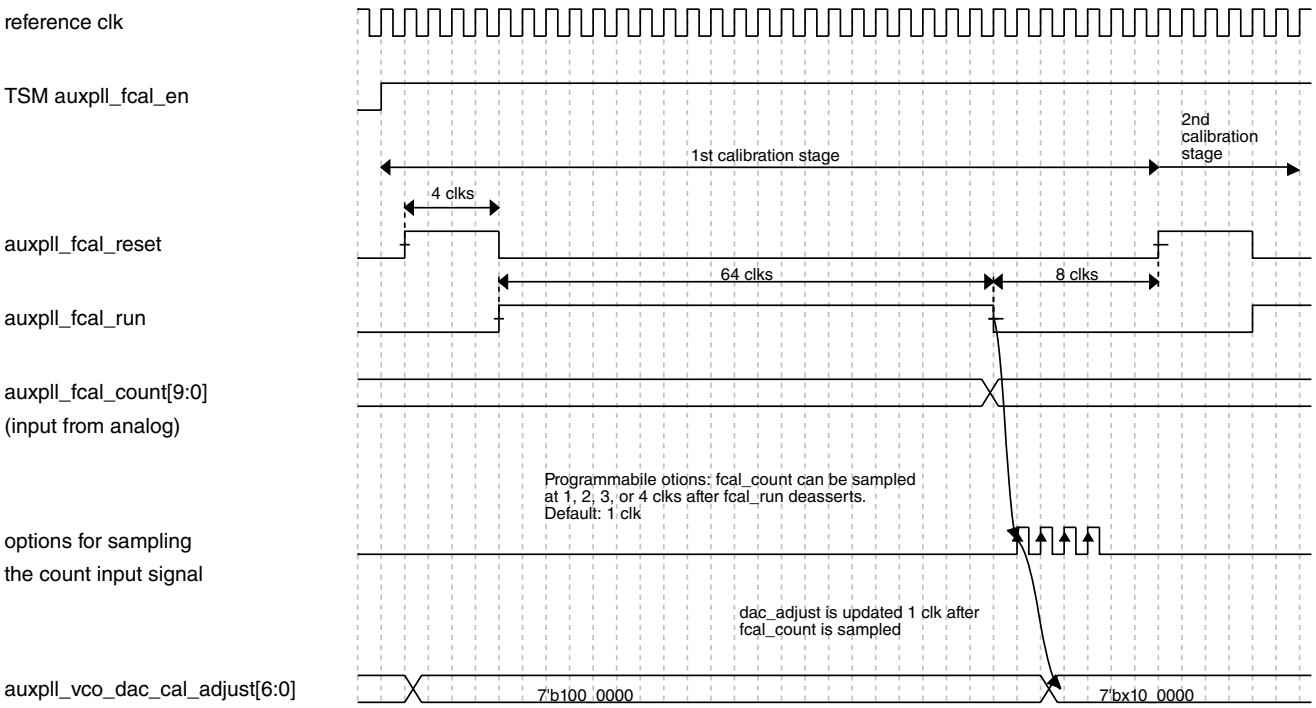



**Figure 64-56. RX AuxPLL FCAL timing**

## 64.3.3.2.11 RX RC Calibration

The RX RC Calibration module finds the best 5bit calibration code for the RC calibration circuit in the analog. From the resulting RC calibration code, separate 5bit codes are derived for each of the following analog circuits: TZA, BBA, BBA2, and ADC. The calibration sequence is initiated by a TSM control signal, and it is expected that this calibration will be performed on each RX warmup.

Features of the module include the following:

- The run window length for each of the five calibration stages is fixed at 4us. The total calibration time is 1us + 5*(4us + 10clks + ipr_rccal_smpl_dly[1:0] clks) - 1 clk.
  - For a 32MHz clk, this is <=23us for all values of ipr_rccal_smpl_dly[1:0]
  - For a 26MHz, this is <=23us for ipr_rccal_smpl_dly[1:0] = 2'b0, and <=24us for other values.
- The four (TZA, BBA, BBA2, and ADC) calibration codes are derived from the RC calibration code by adding signed 4bit programmable offsets. Each of TZA, BBA, BBA2 and ADC have their own programmable offset with saturation detection.
- The comparator output (rccal_comp_out in figure below) from the analog can be programmably inverted.

- The clk cycle at which the comp_out input from the analog is sampled is programmable.
- The output for each of the individual calibration codes (TZA, BBA, BBA2, and ADC) can be applied manually from a register

The module uses a binary search to find the best 5bit calibration code (rccal_code[4:0]). At the end of each calibration stage, the comp_out value output from the analog is sampled, optionally inverted, and used to determine whether the appropriate rccal_code[4:0] bit should be set or cleared.

The figure below illustrates the RC calibration timing. There is a 1us initialization after the calibration circuit is enabled, and then each of the 5 calibration stages requires 5us, for a total of 26us for the entire calibration sequence. Only the inialization 1us and the 1st calibration stage are shown in the figure. The rccal_comp_out signal level at end the end of the calibration stage is used to determine whether the appropriate bit of rccal_code[4:0] will be set or cleared.

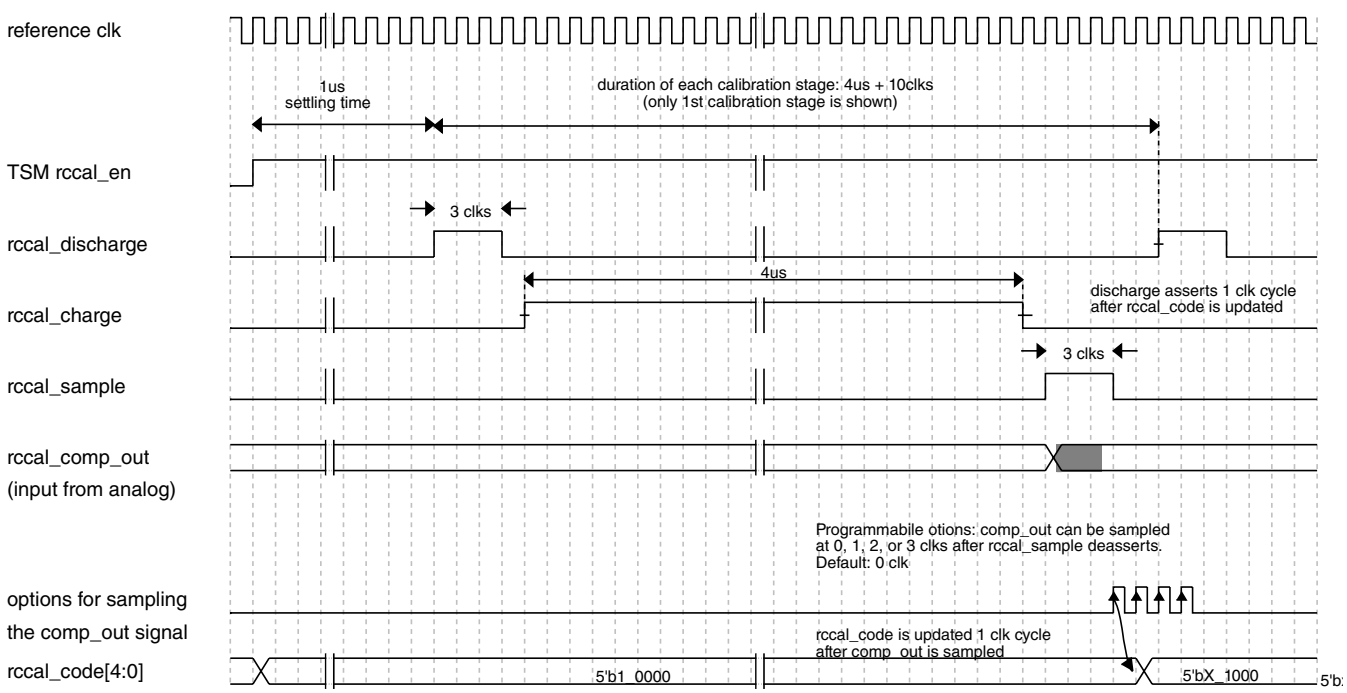



**Figure 64-57. RX RCCAL timing**

## 64.3.3.3 Memory Map and register definition

The rx_dig memory map and description of the rx_dig registers is included in the following register section.

## 64.3.3.3.1 XCVR_RX_DIG register descriptions

### 64.3.3.3.1.1 XCVR_RX_DIG_ADDR Memory map

XCVR_RX_DIG base address: 4103_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | RX Digital Control (RX_DIG_CTRL) | 32 | RW | 0000_0000h |
| 4h | AGC Control 0 (AGC_CTRL_0) | 32 | RW | 0000_0000h |
| 8h | AGC Control 1 (AGC_CTRL_1) | 32 | RW | 0000_0000h |
| Ch | AGC Control 2 (AGC_CTRL_2) | 32 | RW | 00A6_9000h |
| 10h | AGC Control 3 (AGC_CTRL_3) | 32 | RW | 0000_0000h |
| 14h | AGC Status (AGC_STAT) | 32 | RO | 0000_0000h |
| 18h | RSSI Control 0 (RSSI_CTRL_0) | 32 | RW | 0030_0000h |
| 1Ch | RSSI Control 1 (RSSI_CTRL_1) | 32 | RW | 0000_0000h |
| 24h | DCOC Control 0 (DCOC_CTRL_0) | 32 | RW | 0000_0000h |
| 28h | DCOC Control 1 (DCOC_CTRL_1) | 32 | RW | 0000_0000h |
| 2Ch | DCOC DAC Initialization (DCOC_DAC_INIT) | 32 | RW | 8080_2020h |
| 30h | DCOC Digital Correction Manual Override (DCOC_DIG_MAN) | 32 | RW | 0000_0000h |
| 34h | DCOC Calibration Gain (DCOC_CAL_GAIN) | 32 | RW | 0000_0000h |
| 38h | DCOC Status (DCOC_STAT) | 32 | RO | 8080_2020h |
| 3Ch | DCOC DC Estimate (DCOC_DC_EST) | 32 | RO | 0000_0000h |
| 40h | DCOC Calibration Reciprocals (DCOC_CAL_RCP) | 32 | RW | 0000_0000h |
| 44h | DCOC Control 2 (DCOC_CTRL_2) | 32 | RW | 0000_0000h |
| 48h | IQMC Control (IQMC_CTRL) | 32 | RW | 0400_8000h |
| 4Ch | IQMC Calibration (IQMC_CAL) | 32 | RW | 0000_0400h |
| 50h | LNA_GAIN Step Values 3..0 (LNA_GAIN_VAL_3_0) | 32 | RW | 3809_321Dh |
| 54h | LNA_GAIN Step Values 7..4 (LNA_GAIN_VAL_7_4) | 32 | RW | 8B74_5D4Fh |
| 58h | LNA_GAIN Step Values 8 (LNA_GAIN_VAL_8) | 32 | RW | 0000_B6A1h |
| 5Ch | BBA Resistor Tune Values 7..0 (BBA_RES_TUNE_VAL_7_0) | 32 | RW | 0000_0000h |
| 60h | BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_VAL_10_8) | 32 | RW | 0000_0000h |
| 64h | LNA Linear Gain Values 2..0 (LNA_GAIN_LIN_VAL_2_0) | 32 | RW | 0000_0000h |
| 68h | LNA Linear Gain Values 5..3 (LNA_GAIN_LIN_VAL_5_3) | 32 | RW | 0000_0000h |
| 6Ch | LNA Linear Gain Values 8..6 (LNA_GAIN_LIN_VAL_8_6) | 32 | RW | 0000_0000h |
| 70h | LNA Linear Gain Values 9 (LNA_GAIN_LIN_VAL_9) | 32 | RW | 0000_0000h |
| 74h | BBA Resistor Tune Values 3..0 (BBA_RES_TUNE_LIN_VAL_3_0) | 32 | RW | 0000_0000h |
| 78h | BBA Resistor Tune Values 7..4 (BBA_RES_TUNE_LIN_VAL_7_4) | 32 | RW | 0000_0000h |
| 7Ch | BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_LIN_VAL_10_8) | 32 | RW | 0000_0000h |
| 80h | AGC Gain Tables Step 03..00 (AGC_GAIN_TBL_03_00) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 84h | AGC Gain Tables Step 07..04 (AGC_GAIN_TBL_07_04) | 32 | RW | 0000_0000h |
| 88h | AGC Gain Tables Step 11..08 (AGC_GAIN_TBL_11_08) | 32 | RW | 0000_0000h |
| 8Ch | AGC Gain Tables Step 15..12 (AGC_GAIN_TBL_15_12) | 32 | RW | 0000_0000h |
| 90h | AGC Gain Tables Step 19..16 (AGC_GAIN_TBL_19_16) | 32 | RW | 0000_0000h |
| 94h | AGC Gain Tables Step 23..20 (AGC_GAIN_TBL_23_20) | 32 | RW | 0000_0000h |
| 98h | AGC Gain Tables Step 26..24 (AGC_GAIN_TBL_26_24) | 32 | RW | 0000_0000h |
| A0h - 108h | DCOC Offset (DCOC_OFFSET_0 - DCOC_OFFSET_26) | 32 | RW | 0000_0000h |
| 10Ch | DCOC BBA DAC Step (DCOC_BBA_STEP) | 32 | RW | 0000_0000h |
| 110h | DCOC TZA DAC Step 0 (DCOC_TZA_STEP_0) | 32 | RW | 0000_0000h |
| 114h | DCOC TZA DAC Step 1 (DCOC_TZA_STEP_1) | 32 | RW | 0000_0000h |
| 118h | DCOC TZA DAC Step 2 (DCOC_TZA_STEP_2) | 32 | RW | 0000_0000h |
| 11Ch | DCOC TZA DAC Step 3 (DCOC_TZA_STEP_3) | 32 | RW | 0000_0000h |
| 120h | DCOC TZA DAC Step 4 (DCOC_TZA_STEP_4) | 32 | RW | 0000_0000h |
| 124h | DCOC TZA DAC Step 5 (DCOC_TZA_STEP_5) | 32 | RW | 0000_0000h |
| 128h | DCOC TZA DAC Step 6 (DCOC_TZA_STEP_6) | 32 | RW | 0000_0000h |
| 12Ch | DCOC TZA DAC Step 7 (DCOC_TZA_STEP_7) | 32 | RW | 0000_0000h |
| 130h | DCOC TZA DAC Step 5 (DCOC_TZA_STEP_8) | 32 | RW | 0000_0000h |
| 134h | DCOC TZA DAC Step 9 (DCOC_TZA_STEP_9) | 32 | RW | 0000_0000h |
| 138h | DCOC TZA DAC Step 10 (DCOC_TZA_STEP_10) | 32 | RW | 0000_0000h |
| 160h | DCOC Calibration Fail Thresholds (DCOC_CAL_FAIL_TH) | 32 | RW | 0000_0000h |
| 164h | DCOC Calibration Pass Thresholds (DCOC_CAL_PASS_TH) | 32 | RW | 0000_0000h |
| 168h | DCOC Calibration Alpha (DCOC_CAL_ALPHA) | 32 | RO | 0000_0000h |
| 16Ch | DCOC Calibration Beta Q (DCOC_CAL_BETA_Q) | 32 | RO | 0000_0000h |
| 170h | DCOC Calibration Beta I (DCOC_CAL_BETA_I) | 32 | RO | 0000_0000h |
| 174h | DCOC Calibration Gamma (DCOC_CAL_GAMMA) | 32 | RO | 0000_0000h |
| 178h | DCOC Calibration IIR (DCOC_CAL_IIR) | 32 | RW | 0000_0000h |
| 180h - 188h | DCOC Calibration Result (DCOC_CAL1 - DCOC_CAL3) | 32 | RO | 0000_0000h |
| 190h | RX_DIG CCA ED LQI Control Register 0 (CCA_ED_LQI_CTRL_0) | 32 | RW | 0000_0000h |
| 194h | RX_DIG CCA ED LQI Control Register 1 (CCA_ED_LQI_CTRL_1) | 32 | RW | 0000_0000h |
| 198h | RX_DIG CCA ED LQI Status Register 0 (CCA_ED_LQI_STAT_0) | 32 | RO | 0000_0000h |
| 1A0h | Receive Channel Filter Coefficient 0 (RX_CHF_COEF_0) | 32 | RW | 0000_0000h |
| 1A4h | Receive Channel Filter Coefficient 1 (RX_CHF_COEF_1) | 32 | RW | 0000_0000h |
| 1A8h | Receive Channel Filter Coefficient 2 (RX_CHF_COEF_2) | 32 | RW | 0000_0000h |
| 1ACh | Receive Channel Filter Coefficient 3 (RX_CHF_COEF_3) | 32 | RW | 0000_0000h |
| 1B0h | Receive Channel Filter Coefficient 4 (RX_CHF_COEF_4) | 32 | RW | 0000_0000h |
| 1B4h | Receive Channel Filter Coefficient 5 (RX_CHF_COEF_5) | 32 | RW | 0000_0000h |
| 1B8h | Receive Channel Filter Coefficient 6 (RX_CHF_COEF_6) | 32 | RW | 0000_0000h |
| 1BCh | Receive Channel Filter Coefficient 7 (RX_CHF_COEF_7) | 32 | RW | 0000_0000h |
| 1C0h | Receive Channel Filter Coefficient 8 (RX_CHF_COEF_8) | 32 | RW | 0000_0000h |
| 1C4h | Receive Channel Filter Coefficient 9 (RX_CHF_COEF_9) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

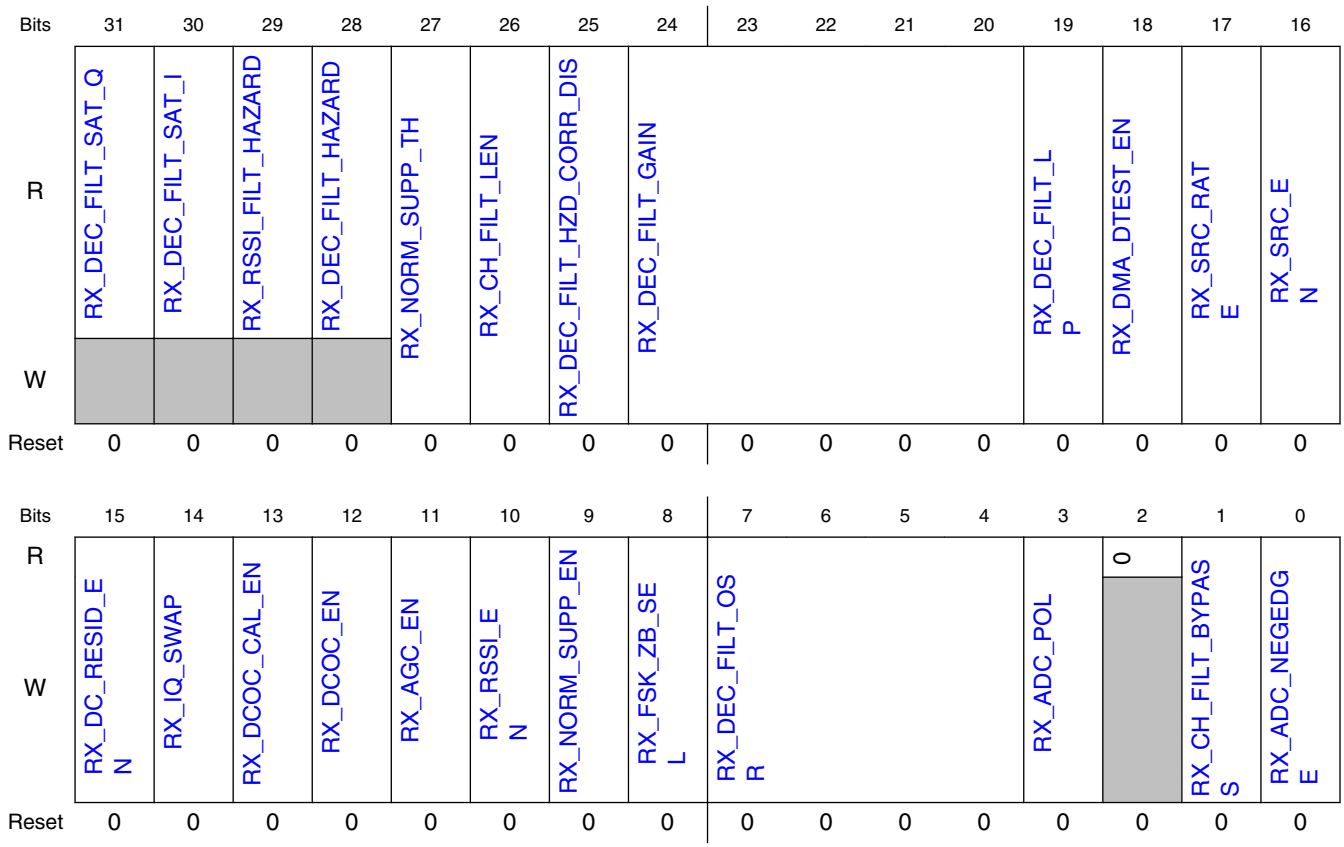
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1C8h | Receive Channel Filter Coefficient 10 (RX_CHF_COEF_10) | 32 | RW | 0000_0000h |
| 1CCh | Receive Channel Filter Coefficient 11 (RX_CHF_COEF_11) | 32 | RW | 0000_0000h |
| 1D0h | AGC Manual AGC Index (AGC_MAN_AGC_IDX) | 32 | RW | 0000_0000h |
| 1D4h | DC Residual Control (DC_RESID_CTRL) | 32 | RW | 0000_0000h |
| 1D8h | DC Residual Estimate (DC_RESID_EST) | 32 | RO | 0000_0000h |
| 1DCh | RX RC Calibration Control0 (RX_RCCAL_CTRL0) | 32 | RW | 0000_0000h |
| 1E0h | RX RC Calibration Control1 (RX_RCCAL_CTRL1) | 32 | RW | 0000_0000h |
| 1E4h | RX RC Calibration Status (RX_RCCAL_STAT) | 32 | RO | 0210_4210h |
| 1E8h | Aux PLL Frequency Calibration Control (AUXPLL_FCAL_CTRL) | 32 | RW | 0040_0000h |
| 1ECh | Aux PLL Frequency Calibration Count 6 (AUXPLL_FCAL_CNT6) | 32 | RO | 0000_0000h |
| 1F0h | Aux PLL Frequency Calibration Count 5 and 4 (AUXPLL_FCAL_CNT5_4) | 32 | RO | 0000_0000h |
| 1F4h | Aux PLL Frequency Calibration Count 3 and 2 (AUXPLL_FCAL_CNT3_2) | 32 | RO | 0000_0000h |
| 1F8h | Aux PLL Frequency Calibration Count 1 and 0 (AUXPLL_FCAL_CNT1_0) | 32 | RO | 0000_0000h |

## 64.3.3.3.1.2 RX Digital Control (RX_DIG_CTRL)

### *64.3.3.3.1.2.1 Offset*

| Register | Offset |
|---|---|
| RX_DIG_CTRL | 0h |

### 64.3.3.3.1.2.2 Diagram




### 64.3.3.3.1.2.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_DEC_FILT_SAT_Q | Decimator output, saturation detected for Q channel<br>This bit will be set if a saturation condition is detected in the decimator filter Q channel. This bit is cleared by tsm_rx_dcoc_init and rx_init.<br>0b - A saturation condition has not occurred.<br>1b - A saturation condition has occurred. |
| 30<br>RX_DEC_FILT_SAT_I | Decimator output, saturation detected for I channel<br>This bit will be set if a saturation condition is detected in the decimator filter I channel. This bit is cleared by tsm_rx_dcoc_init and rx_init.<br>0b - A saturation condition has not occurred.<br>1b - A saturation condition has occurred. |
| 29<br>RX_RSSI_FILT_HAZARD | Decimator output for RSSI, hazard condition detected<br>This bit will be set if a hazard condition is detected in either the I or Q decimator filter related to the wideband RSSI measurement. This does not indicate that a data corruption has occurred, only that the conditions associated with data corruption were detected. This bit is cleared by rx_init.<br>0b - A hazard condition has not been detected<br>1b - A hazard condition has been detected |
| 28 | Decimator output, hazard condition detected |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RX_DEC_FILT_HAZARD | This bit will be set if a hazard condition is detected in either the I or Q decimator filter. This does not indicate that a data corruption has occurred, only that the conditions associated with data corruption were detected. This bit is cleared by rx_init.<br>0b - A hazard condition has not been detected<br>1b - A hazard condition has been detected |
| 27<br>RX_NORM_SUPP_TH | Normalizer Suppression Threshold<br>Configures the threshold for suppressing the normalizer's phase output when RX_NORM_SUPP_EN is set.<br>0b - Normalizer suppression threshold is 12'd7.<br>1b - Normalizer suppression threshold is 12'd15. |
| 26<br>RX_CH_FILT_LEN | RX Channel Filter Length<br>0b - Channel filter length is 24.<br>1b - Channel filter length is 16. Only RX_CHF_COEF_4 - RX_CHF_COEF_11 are used in this mode. |
| 25<br>RX_DEC_FILT_HZD_CORR_DIS | Decimator filter hazard correction disable<br>The decimator uses an asynchronous FIFO between the data coming out of the ADC (in the ADC clock domain) and the decimator output (in the ref osc clock domain, with the rest of the radio digital logic). The hazard detection logic checks for a hazard condition when the ADC clock domain logic may be about to write into the same FIFO location where the ref osc clock domain logic is about to read. If hazard correction is not disabled, the FIFO pointers will be adjusted on detection of a hazard condition.<br>0b - Hazard correction is enabled<br>1b - Hazard correction is disabled |
| 24-20<br>RX_DEC_FILT_GAIN | Decimation Filter Fractional Gain<br>Defines the fractional gain which is applied at the decimator output. The format is u0.5. The gain applied is 1 + RX_DEC_FILT_GAIN/32, so e.g. if RX_DEC_FILT_GAIN is 5'b10110=5'd22, the gain is 1+22/32=1.6875. The nominal gain through the ADC is 2048codes/1.7V; the combined gain through the ADC and decimator would therefore be 2048codes/1.7e3mV*1.6875=2.03codes/mV for the example RX_DEC_FILT_GAIN=5'b10110. |
| 19<br>RX_DEC_FILT_LP | RX Decimator Low Power<br>This bit can be set to reduce the power in the decimator.<br>0b - Decimator operates in normal mode.<br>1b - Decimator operates in low power mode. |
| 18<br>RX_DMA_DTEST_EN | RX DMA and DTEST enable<br>This bit should be set to ensure that all of the rx_dig outputs related to DMA or DTEST are enabled. In mission mode this bit is intended to be cleared to reduce switching power. |
| 17<br>RX_SRC_RATE | RX Sample Rate Converter Rate Selections<br>0b - SRC is configured for a First Order Hold rate of 8/13.<br>1b - SRC is configured for a Zero Order Hold rate of 12/13. |
| 16<br>RX_SRC_EN | RX Sample Rate Converter Enable<br>0b - SRC is disabled.<br>1b - SRC is enabled. |
| 15<br>RX_DC_RESID_EN | DC Residual Enable<br>Enables DC Residual block<br>0b - DC Residual block is disabled.<br>1b - DC Residual block is enabled. |
| 14<br>RX_IQ_SWAP | RX IQ Swap<br>Enable swap of I/Q channels (does not affect ADC raw mode).<br>0b - IQ swap is disabled.<br>1b - IQ swap is enabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13<br>RX_DCOC_CAL_EN | DCOC Calibration Enable<br>Enable DCOC warm-up calibration in receiver.<br>0b - DCOC calibration is disabled.<br>1b - DCOC calibration is enabled. |
| 12<br>RX_DCOC_EN | DCOC Enable<br>Enables DCO calculation and application of corrections.<br>0b - DCOC is disabled.<br>1b - DCOC is enabled. |
| 11<br>RX_AGC_EN | AGC Global Enable<br>Does NOT affect user gains (user gain programming has priority).<br>0b - AGC is disabled.<br>1b - AGC is enabled. |
| 10<br>RX_RSSI_EN | RSSI Measurement Enable<br>0b - RSSI measurement is disabled.<br>1b - RSSI measurement is enabled. |
| 9<br>RX_NORM_SUPP_EN | Normalizer Suppression Enable<br>0b - Normalizer suppression is disabled.<br>1b - Normalizer suppression is enabled. |
| 8<br>RX_FSK_ZB_SEL | FSK / 802.15.4 demodulator select<br>Select between FSK and 802.15.4 demodulator. This is used in the RSSI and AGC to select trigger source input signals.<br>0b - FSK demodulator.<br>1b - 802.15.4 demodulator. |
| 7-4<br>RX_DEC_FILT_OSR | Decimation Filter Oversampling<br>**NOTE:** All undocumented values are Reserved.<br>0000b - OSR 4<br>0001b - OSR 8<br>0010b - OSR 16<br>0011b - OSR 6<br>0100b - OSR 32<br>0101b - OSR 12<br>0110b - OSR 24<br>0111b - OSR 48<br>1000b - OSR 64 |
| 3<br>RX_ADC_POL | Receive ADC Polarity<br>Selects polarity of the ADC data<br>0b - ADC output of 1'b0 maps to -1, 1'b1 maps to +1 (default)<br>1b - ADC output of 1'b0 maps to +1, 1'b1 maps to -1 |
| 2<br>— | Reserved. |
| 1<br>RX_CH_FILT_BYPASS | Receive Channel Filter Bypass<br>Selects whether to disable and bypass channel filter.<br>0b - Channel filter is enabled.<br>1b - Disable and bypass channel filter. |
| 0<br>RX_ADC_NEGEDGE | Receive ADC Negative Edge Selection<br>Selects which edge of the clock the ADC data is registered.<br>0b - Register ADC data on positive edge of clock<br>1b - Register ADC data on negative edge of clock |

## 64.3.3.3.1.3 AGC Control 0 (AGC_CTRL_0)

### *64.3.3.3.1.3.1 Offset*

| Register | Offset |
|---|---|
| AGC_CTRL_0 | 4h |

### *64.3.3.3.1.3.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AGC_DOWN_RSSI_THRESH | | | | | | | | AGC_UP_RSSI_THRESH | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AGC_DOWN_LNA_STEP_SZ | | | | AGC_DOWN_BBA_STEP_SZ | | | | AGC_UP_SRC | AGC_UP_EN | AGC_FREEZE_SRC | | AGC_FREEZE_EN | SLOW_AGC_SRC | | SLOW_AGC_EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.3.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>AGC_DOWN_RSSI_THRESH | AGC DOWN RSSI Threshold<br>ADC RSSI threshold to take downward step (AGC slow). If the ADC RSSI measurment is higher than this threshold, a downward gain step can be taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 23-16<br>AGC_UP_RSSI_THRESH | AGC UP RSSI Threshold<br>ADC RSSI threshold to take upward step (AGC slow). If the ADC RSSI measurment is below this threshold, an upward gain step can be taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 15-12 | AGC_DOWN_LNA_STEP_SZ<br>Number of table steps for downward step (LNA) in AGC fast. |

*Table continues on the next page...*

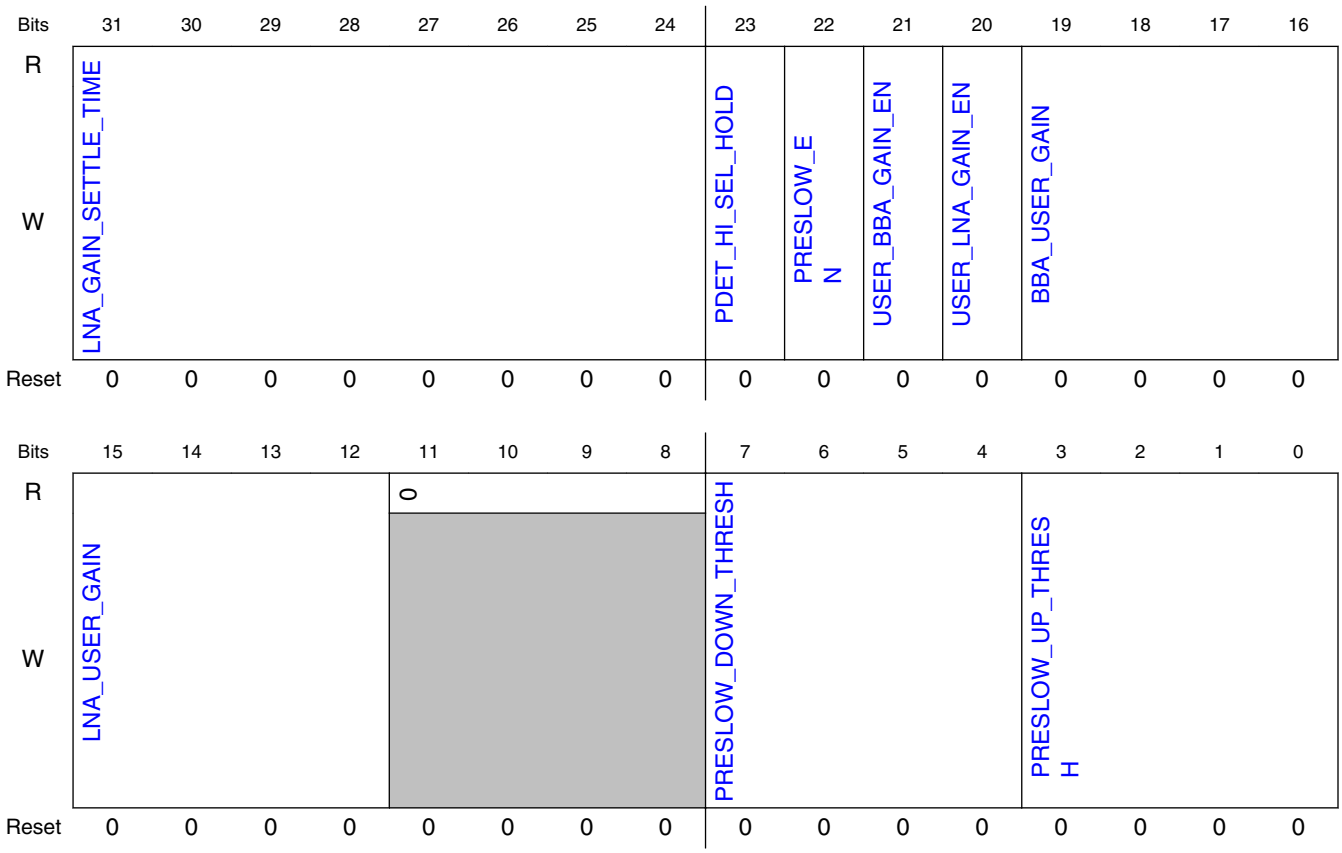
| Field | Function |
|---|---|
| AGC_DOWN_LNA_STEP_SZ | |
| 11-8<br>AGC_DOWN_BBA_STEP_SZ | AGC_DOWN_BBA_STEP_SZ<br>Number of table steps for downward step (BBA) in AGC fast. |
| 7<br>AGC_UP_SRC | AGC Up Source<br>Criterion to use for upward AGC steps in SLOW state. For pdet_lo, a timing window is observed for no assertions of the LO peak detectors and then an upward step is made. If RSSI is selected, the current RSSI measurement is compared with the UP threshold to determine if an upward step is due.<br>0b - PDET LO<br>1b - RSSI |
| 6<br>AGC_UP_EN | AGC Up Enable<br>Allow AGC to take upward steps in the gain table in slow mode. |
| 5-4<br>AGC_FREEZE_SRC | AGC Freeze Source Selection<br>Select trigger source for entering freeze AGC (HOLD state).<br>00b - Access Address match (for active protocol)<br>01b - Preamble Detect (for active protocol)<br>10b - PD confirmation / Access Address match (for active protocol) |
| 3<br>AGC_FREEZE_EN | AGC Freeze Enable<br>Allow AGC to freeze (ie enter HOLD state). AGC can still go to hold mode if timer expires (same as fast expire) from slow mode. |
| 2-1<br>SLOW_AGC_SRC | Slow AGC Source Selection<br>Select trigger source for entering slow AGC. For address match and preamble detect, the trigger is generated by the PHY. If the Fast AGC expire timer is selected, when the timer expires, the AGC state machine will transition into SLOW.<br>00b - Access Address match (for active protocol)<br>01b - Preamble Detect (for active protocol)<br>10b - Fast AGC expire timer<br>11b - Reserved |
| 0<br>SLOW_AGC_EN | Slow AGC Enable<br>Allow AGC to enter into slow mode. |

## 64.3.3.3.1.4 AGC Control 1 (AGC_CTRL_1)

### *64.3.3.3.1.4.1 Offset*

| Register | Offset |
|---|---|
| AGC_CTRL_1 | 8h |

### 64.3.3.3.1.4.2 Diagram




### 64.3.3.3.1.4.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LNA_GAIN_SETTLE_TIME | LNA_GAIN_SETTLE_TIME<br>At LNA gain change, number of clocks to assert TZA peak detector reset (for automatic control). Should be programmed greater than zero. |
| 23<br>PDET_HI_SEL_HOLD | AGC HOLD hysteresis<br>When enabled, and the AGC state machine is in HOLD, the BBA clip detector threshold is raised by 1 count.<br>0b - Disabled.<br>1b - Enabled. |
| 22<br>PRESLOW_EN | Pre-slow Enable<br>Enable the pre-slow state where signal headroom is checked before entering SLOW.<br>0b - Pre-slow is disabled.<br>1b - Pre-slow is enabled. |
| 21<br>USER_BBA_GAIN_EN | User BBA Gain Enable<br>Enable user defined BBA gain (no AGC). |
| 20 | User LNA Gain Enable |

*Table continues on the next page...*

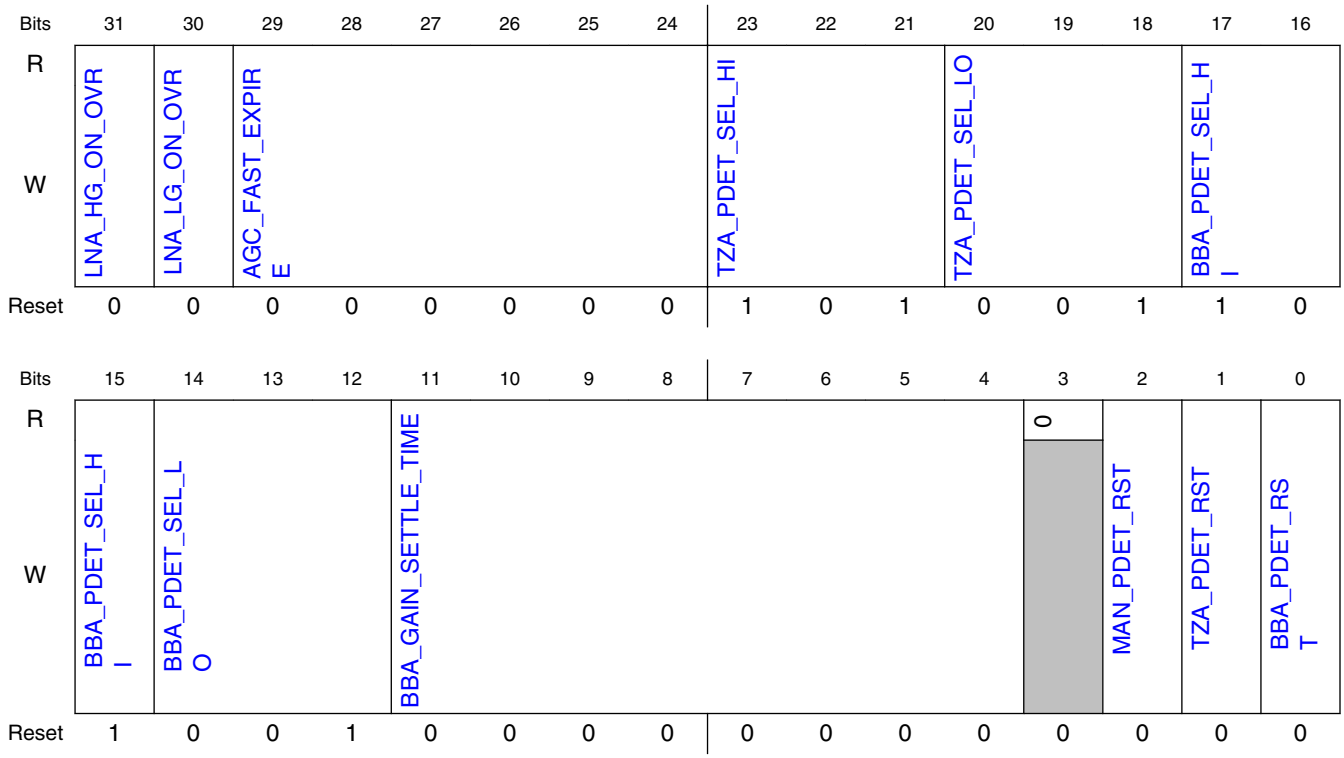
| Field | Function |
|---|---|
| USER_LNA_GAIN_EN | Enable user defined LNA gain (no AGC). |
| 19-16<br>BBA_USER_GAIN | BBA_USER_GAIN<br>User defined BBA gain index if user_bba_gain_en =1 |
| 15-12<br>LNA_USER_GAIN | LNA_USER_GAIN<br>User defined LNA gain index if user_lna_gain_en =1. |
| 11-8<br>— | Reserved. |
| 7-4<br>PRESLOW_DOWN_THRESH | PRESLOW_DOWN_THRESH<br>ADC RSSI threshold to take downward step in (AGC PRESLOW state). The actual threshold is computed as the programmed value minus 10, so the threshold assumes values in the range [-10,5]. If the ADC RSSI measurment is above this threshold, a downward gain step of 1 is taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 3-0<br>PRESLOW_UP_THRESH | PRESLOW_UP_THRESH<br>ADC RSSI threshold to take upward step in (AGC PRESLOW state). The actual threshold is computed as the programmed value minus 15, so the threshold assumes values in the range [-15,0]. If the ADC RSSI measurment is below this threshold, an upward gain step of 1 is taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |

## 64.3.3.3.1.5 AGC Control 2 (AGC_CTRL_2)

### *64.3.3.3.1.5.1 Offset*

| Register | Offset |
|---|---|
| AGC_CTRL_2 | Ch |

### 64.3.3.3.1.5.2 Diagram




### 64.3.3.3.1.5.3 Fields

| Field | Function |
|---|---|
| 31<br>LNA_HG_ON_OVR | LNA_HG_ON override<br>If set, the lna high gain "on" signal is always asserted. Otherwise, this signal is asserted automatically based on the LNA_GAIN code. |
| 30<br>LNA_LG_ON_OVR | LNA_LG_ON override<br>If set, the lna low gain "on" signal is always asserted. Otherwise, this signal is asserted automatically based on the LNA_GAIN code. |
| 29-24<br>AGC_FAST_EXPIRE | AGC Fast Expire<br>Expire time (uS) for fast AGC (1-63uS). |
| 23-21<br>TZA_PDET_SEL_HI | TZA PDET Threshold High<br>TZA peak detect HI threshold.<br>000b - 0.600V<br>001b - 0.645V<br>010b - 0.705V<br>011b - 0.750V<br>100b - 0.795V<br>101b - 0.855V<br>110b - 0.900V<br>111b - 0.945V |
| 20-18 | TZA PDET Threshold Low |

*Table continues on the next page...*

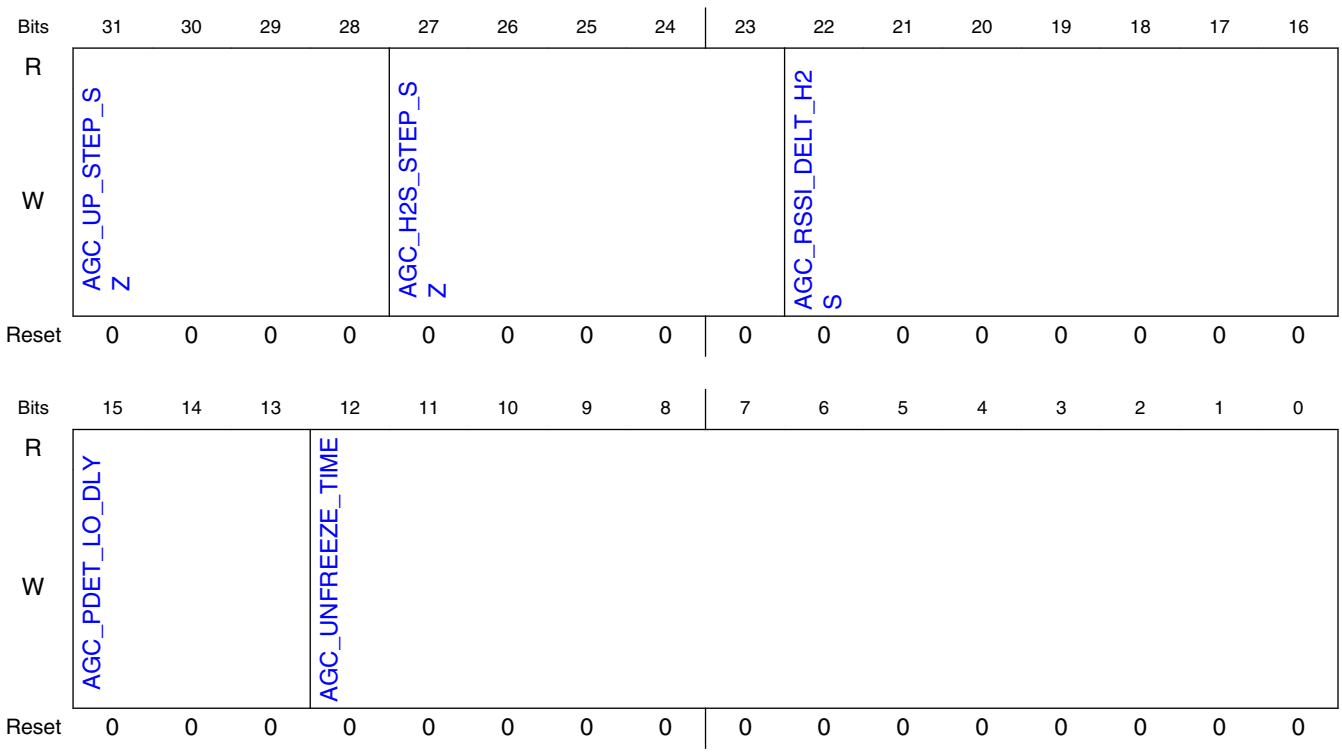
| Field | Function |
|---|---|
| TZA_PDET_SEL_LO | TZA peak detect LO threshold.<br>000b - 0.600V<br>001b - 0.615V<br>010b - 0.630V<br>011b - 0.645V<br>100b - 0.660V<br>101b - 0.675V<br>110b - 0.690V<br>111b - 0.705V |
| 17-15<br>BBA_PDET_SEL_HI | BBA PDET Threshold High<br>BBA peak detect HI threshold.<br>000b - 0.600V<br>001b - 0.795V<br>010b - 0.900V<br>011b - 0.945V<br>100b - 1.005V<br>101b - 1.050V<br>110b - 1.095V<br>111b - 1.155V |
| 14-12<br>BBA_PDET_SEL_LO | BBA PDET Threshold Low<br>BBA peak detect LO threshold.<br>000b - 0.600V<br>001b - 0.615V<br>010b - 0.630V<br>011b - 0.645V<br>100b - 0.660V<br>101b - 0.675V<br>110b - 0.690V<br>111b - 0.705V |
| 11-4<br>BBA_GAIN_SETTLE_TIME | BBA Gain Settle Time<br>Number of clocks to assert BBA peak detector reset (for automatic control). Should be programmed greater than zero. |
| 3<br>— | Reserved. |
| 2<br>MAN_PDET_RST | MAN PDET Reset<br>0b - The peak detector reset signals are controlled automatically by the AGC.<br>1b - The BBA_PDET_RST and TZA_PDET_RST are used to manually control the peak detector reset signals. |
| 1<br>TZA_PDET_RST | TZA PDET Reset<br>TZA peak detector reset, manual control. |
| 0<br>BBA_PDET_RST | BBA PDET Reset<br>BBA peak detector reset, manual control. |

## 64.3.3.3.1.6 AGC Control 3 (AGC_CTRL_3)

#### 64.3.3.3.1.6.1 Offset

| Register | Offset |
|---|---|
| AGC_CTRL_3 | 10h |

#### 64.3.3.3.1.6.2 Diagram




#### 64.3.3.3.1.6.3 Fields

| Field | Function |
|---|---|
| 31-28<br>AGC_UP_STEP_SZ | AGC Up Step Size<br>Number of AGC gain table steps for upward step |
| 27-23<br>AGC_H2S_STEP_SZ | AGC_H2S_STEP_SZ<br>AGC gain table step size for hold to slow jump. |
| 22-16<br>AGC_RSSI_DELT_H2S | AGC_RSSI_DELT_H2S<br>RSSI delta that causes hold to slow transition. This delta is observed from a previous RSSI measurement to a current measurement. |
| 15-13 | AGC Peak Detect Low Delay<br>Time (uS) to wait for pdet low to assert (1-7uS). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| AGC_PDET_LO_DLY | |
| 12-0<br>AGC_UNFREEZE_TIME | AGC Unfreeze Time<br>Time (uS) for AGC to unfreeze (1-8191uS) from HOLD and re-enter SLOW state. |

## 64.3.3.3.1.7 AGC Status (AGC_STAT)

### *64.3.3.3.1.7.1 Offset*

| Register | Offset |
|---|---|
| AGC_STAT | 14h |

### *64.3.3.3.1.7.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RSSI_ADC_RAW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | AGC_IDX_AA_MATCH | | | | | AGC_FROZEN | CURR_AGC_IDX | | | | | TZA_PDET_HI_STAT | TZA_PDET_LO_STAT | BBA_PDET_HI_STAT | BBA_PDET_LO_STAT |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.7.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-16 | ADC RAW RSSI Reading |

*Table continues on the next page...*

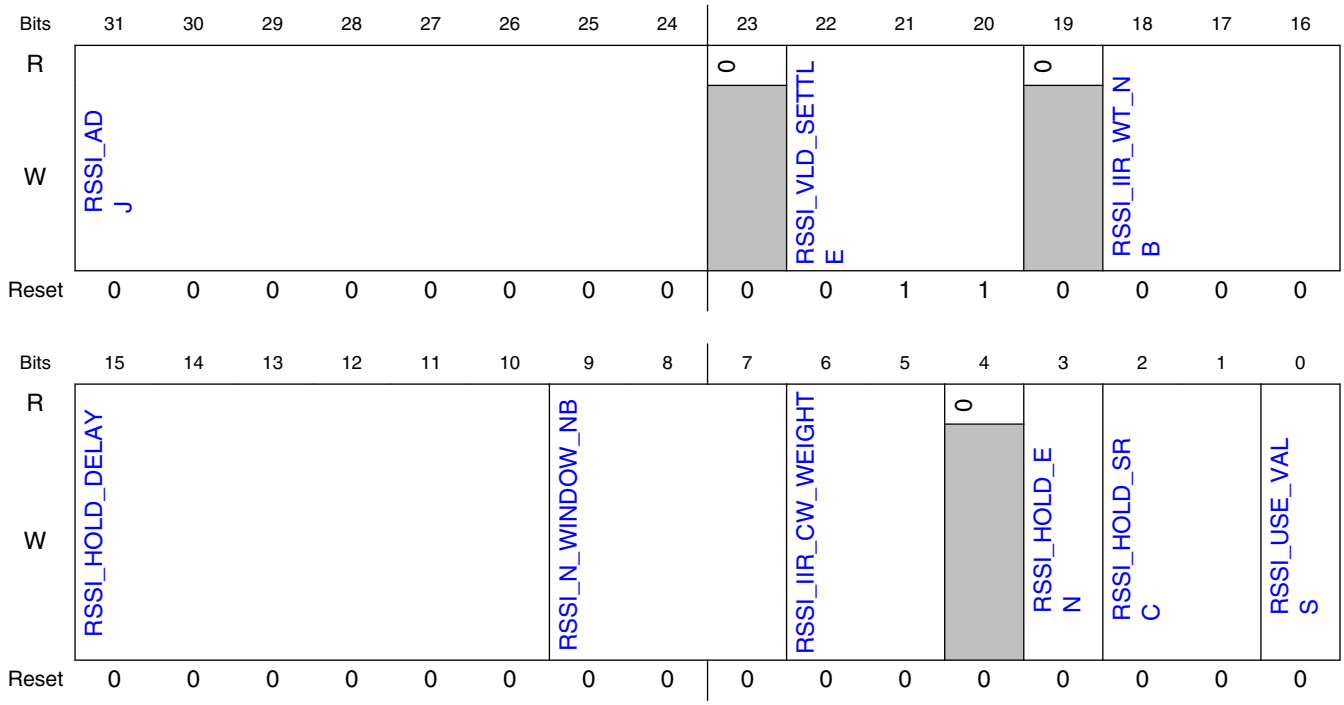
| Field | Function |
|---|---|
| RSSI_ADC_RAW | Reading of ADC rssi (before adjustments) |
| 15<br>— | Reserved. |
| 14-10<br>AGC_IDX_AA_MATCH | AGC Gain Index at AA Match<br>AGC Gain Index captured at AA Match. This value remains static so it can be read by software after the end of the RX burst. The value is cleared to 0 at TSM rx_init and radio reset conditions. |
| 9<br>AGC_FROZEN | AGC Frozen Status<br>Status of AGC freeze.<br>0b - AGC is not frozen.<br>1b - AGC is frozen. |
| 8-4<br>CURR_AGC_IDX | Current AGC Gain Index<br>Current AGC gain table index |
| 3<br>TZA_PDET_HI_STAT | TZA Peak Detector High Status<br>Status of TZA peak detector HI flag (1=set) |
| 2<br>TZA_PDET_LO_STAT | TZA Peak Detector Low Status<br>Status of TZA peak detector LO flag (1=set) |
| 1<br>BBA_PDET_HI_STAT | BBA Peak Detector High Status<br>Status of BBA peak detector HI flag (1=set) |
| 0<br>BBA_PDET_LO_STAT | BBA Peak Detector Low Status<br>Status of BBA peak detector LO flag (1=set) |

## 64.3.3.3.1.8 RSSI Control 0 (RSSI_CTRL_0)

### *64.3.3.3.1.8.1 Offset*

| Register | Offset |
|---|---|
| RSSI_CTRL_0 | 18h |

### 64.3.3.3.1.8.2 Diagram




### 64.3.3.3.1.8.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RSSI_ADJ | RSSI Adjustment<br>RSSI calculation adjustment (8-bit signed 1/4 dB). |
| 23<br>— | Reserved. |
| 22-20<br>RSSI_VLD_SETTLE | RSSI Valid Settle<br>Sets number of us (times 8) that RSSI will be considered invalid after certain events. |
| 19<br>— | Reserved. |
| 18-16<br>RSSI_IIR_WT_NB | RSSI IIR Weighting Narrowband<br>IIR filter weight for RSSI filtering for Narrowband mode.<br>**NOTE:** All undocumented values are Reserved.<br>000b - Bypass<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32 |
| 15-10 | RSSI Hold Delay<br>Sets number of us (times 8) that RSSI will run after a hold event before the value is frozen. |

*Table continues on the next page...*

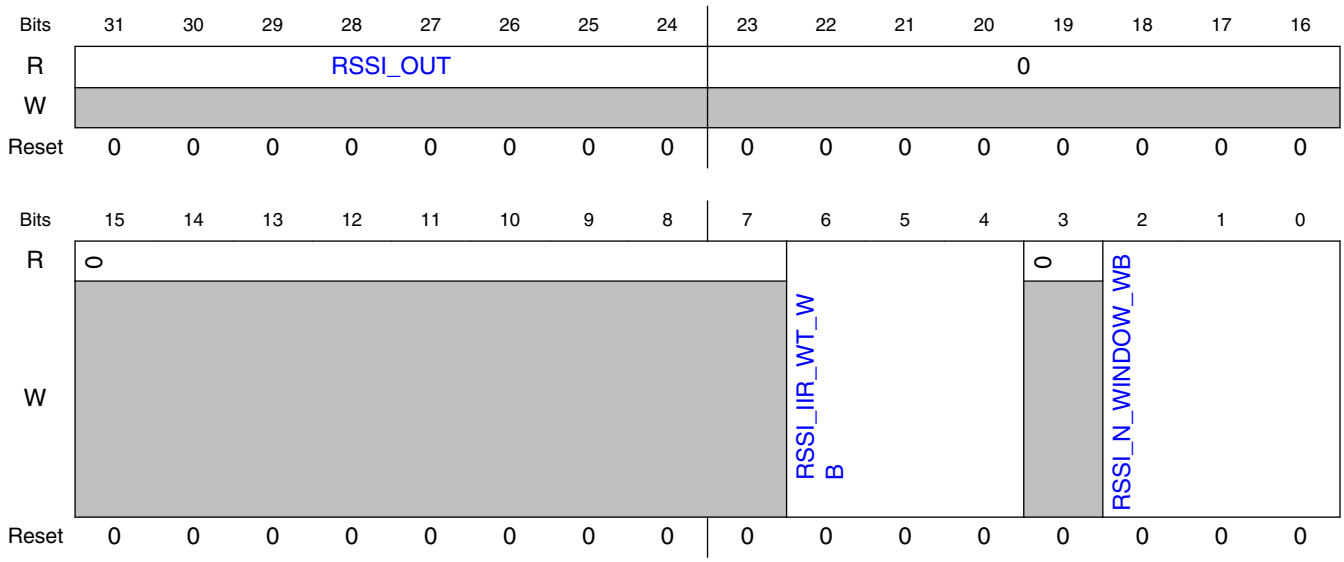
| Field | Function |
|---|---|
| RSSI_HOLD_DELAY | |
| 9-7<br>RSSI_N_WINDOW_NB | RSSI N Window Average Narrowband<br>Selects Averaging window length for RSSI in Narrowband mode.<br>000b - No averaging<br>001b - Averaging window length is 2 samples<br>010b - Averaging window length is 4 samples<br>011b - Averaging window length is 8 samples<br>100b - Averaging window length is 16 samples<br>101b - Averaging window length is 32 samples |
| 6-5<br>RSSI_IIR_CW_WEIGHT | RSSI IIR CW Weighting<br>IIR filter weight for RSSI filtering of a CW input.<br>00b - Bypass<br>01b - 1/8<br>10b - 1/16<br>11b - 1/32 |
| 4<br>— | Reserved. |
| 3<br>RSSI_HOLD_EN | RSSI Hold Enable<br>Enable RSSI to freeze after hold criterion met. RSSI will still be briefly held when a gain change occurs. |
| 2-1<br>RSSI_HOLD_SRC | RSSI Hold Source Selection<br>Select trigger source for freezing RSSI measurement.<br>00b - Access Address match<br>01b - Preamble Detect<br>10b - Reserved<br>11b - 802.15.4 LQI done (1=freeze, 0=run AGC) |
| 0<br>RSSI_USE_VALS | RSSI Values Selection<br>Enable use of LNA and BBA gain values programmed in registers for calculation. |

## 64.3.3.3.1.9 RSSI Control 1 (RSSI_CTRL_1)

### *64.3.3.3.1.9.1 Offset*

| Register | Offset |
|---|---|
| RSSI_CTRL_1 | 1Ch |

### 64.3.3.3.1.9.2 Diagram




### 64.3.3.3.1.9.3 Fields

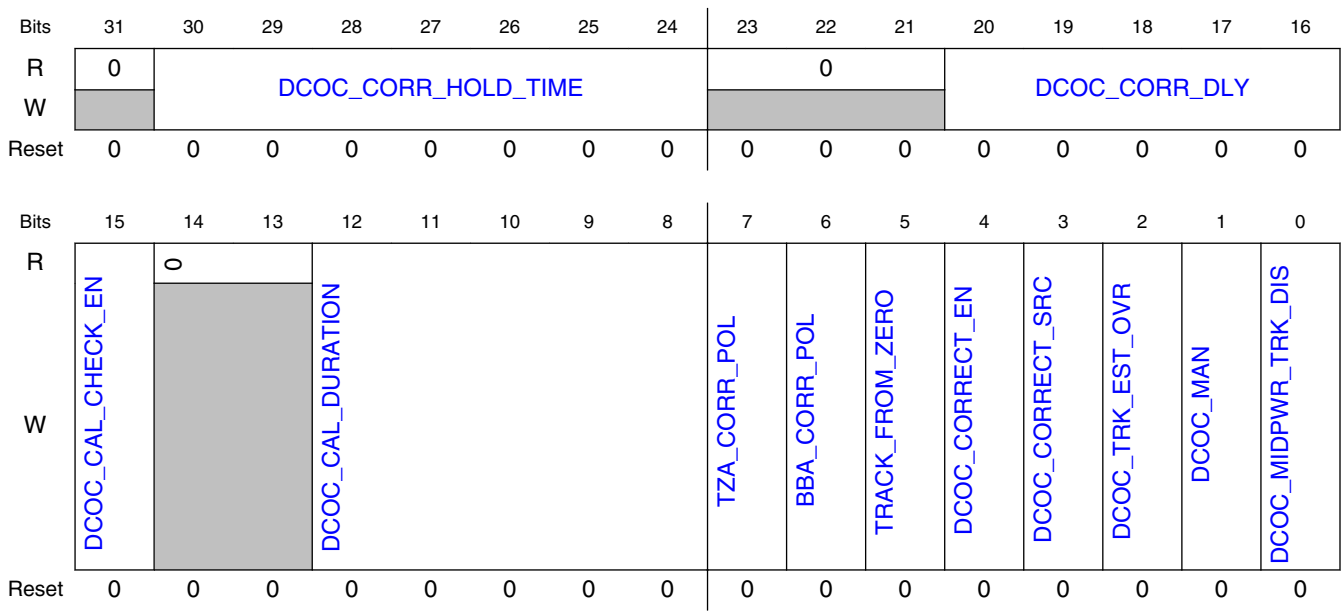
| Field | Function |
|---|---|
| 31-24<br>RSSI_OUT | RSSI Reading<br>RSSI output (8-bit signed). |
| 23-7<br>— | Reserved. |
| 6-4<br>RSSI_IIR_WT_WB | RSSI IIR Weighting Wideband<br>IIR filter weight for RSSI filtering in Wideband mode.<br>**NOTE:** All undocumented values are Reserved.<br>000b - Bypass<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32 |
| 3<br>— | Reserved. |
| 2-0<br>RSSI_N_WINDOW_WB | RSSI N Window Average Wideband<br>Selects Averaging window length for RSSI in Wideband mode.<br>000b - No averaging<br>001b - Averaging window length is 2 samples<br>010b - Averaging window length is 4 samples<br>011b - Averaging window length is 8 samples<br>100b - Averaging window length is 16 samples<br>101b - Averaging window length is 32 samples |

## 64.3.3.3.1.10 DCOC Control 0 (DCOC_CTRL_0)

### *64.3.3.3.1.10.1 Offset*

| Register | Offset |
|---|---|
| DCOC_CTRL_0 | 24h |

### *64.3.3.3.1.10.2 Diagram*




### *64.3.3.3.1.10.3 Fields*

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>DCOC_CORR_HOLD_TIME | DCOC Correction Hold Time<br>Delay from last gain change to freezing the DC correction.<br>0000000b - Reserved<br>0000001-1111110b - For a 32MHz reference clock, this is the delay in microseconds; for other reference clock frequencies, the delay is scaled accordingly.<br>1111111b - The DC correction is not frozen. |
| 23-21<br>— | Reserved. |
| 20-16<br>DCOC_CORR_DLY | DCOC Correction Delay<br>Wait time between corrections.<br>00000b - Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 00001-11111b - For a 32MHz reference clock, this is the wait time in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 15<br>DCOC_CAL_CHECK_EN | DCOC Calibration Check Enable<br>0b - Calibration checking disabled. The DCOC_OFFSET_n registers are always updated during calibration.<br>1b - Calibration checking enabled. The DCOC_OFFSET_n registers are updated conditionally depending on the outcome of the pass/fail threshold checks performed on the alpha-hat and beta-hat estimates during calibration. |
| 14-13<br>— | Reserved. |
| 12-8<br>DCOC_CAL_DURATION | DCOC Calibration Duration<br>00000b - Reserved<br>00001-11111b - For a 32MHz reference clock, this is the calibration duration in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 7<br>TZA_CORR_POL | TZA Correction Polarity<br>Selects polarity of TZA corrections.<br>0b - Normal polarity.<br>1b - Negative polarity. This should be set if the ADC output is inverted, or if the TZA DACs were implemented with negative polarity. |
| 6<br>BBA_CORR_POL | BBA Correction Polarity<br>Selects polarity of BBA corrections.<br>0b - Normal polarity.<br>1b - Negative polarity. This should be set if the ADC output is inverted, or if the BBA DACs were implemented with negative polarity. |
| 5<br>TRACK_FROM_ZERO | Track from Zero<br>Selects whether the tracking estimator resets its DC estimate on every AGC gain change to zero or uses the current I/Q sample.<br>0b - Track from current I/Q sample.<br>1b - Track from zero. |
| 4<br>DCOC_CORRECT_EN | DCOC Correction Enable<br>0b - Correction disabled. The DCOC will not correct the DC offset.<br>1b - Correction enabled. The DCOC will use the TZA and BBA DACs, and apply digital corrections (if DCOC_CORRECT_SRC=1) to correct the DC offset. |
| 3<br>DCOC_CORRECT_SRC | DCOC Corrector Source<br>If not set, the corrector uses only the DCOC calibration table to apply corrections to the DCOC DACs.<br>0b - If correction is enabled, the DCOC will use only the DCOC calibration table to correct the DC offset.<br>1b - If correction is enabled, the DCOC will use the DCOC calibration table and then the tracking estimator to correct the DC offset. |
| 2<br>DCOC_TRK_EST_OVR | Override for the DCOC tracking estimator<br>0b - The tracking estimator is enabled only as needed by the corrector<br>1b - The tracking estimator remains enabled whenever the DCOC is active |
| 1<br>DCOC_MAN | DCOC Manual Override<br>If the manual override bit is set, it forces the DCOC to use the DAC and digital correction values from registers DCOC_DAC_INIT and DCOC_DIG_MAN, respectively. |
| 0<br>DCOC_MIDPWR_TRK_DIS | DCOC Mid Power Tracking Disable<br>Disables tracking correction at mid power levels, as indicated by the TZA and BBA lo peak detectors. This is implemented by reseting the counters associated with DCOC_CORR_DLY and DCOC_CORR_HOLD_TIME, so if the lo peak detectors do not continue to assert, tracking corrections |

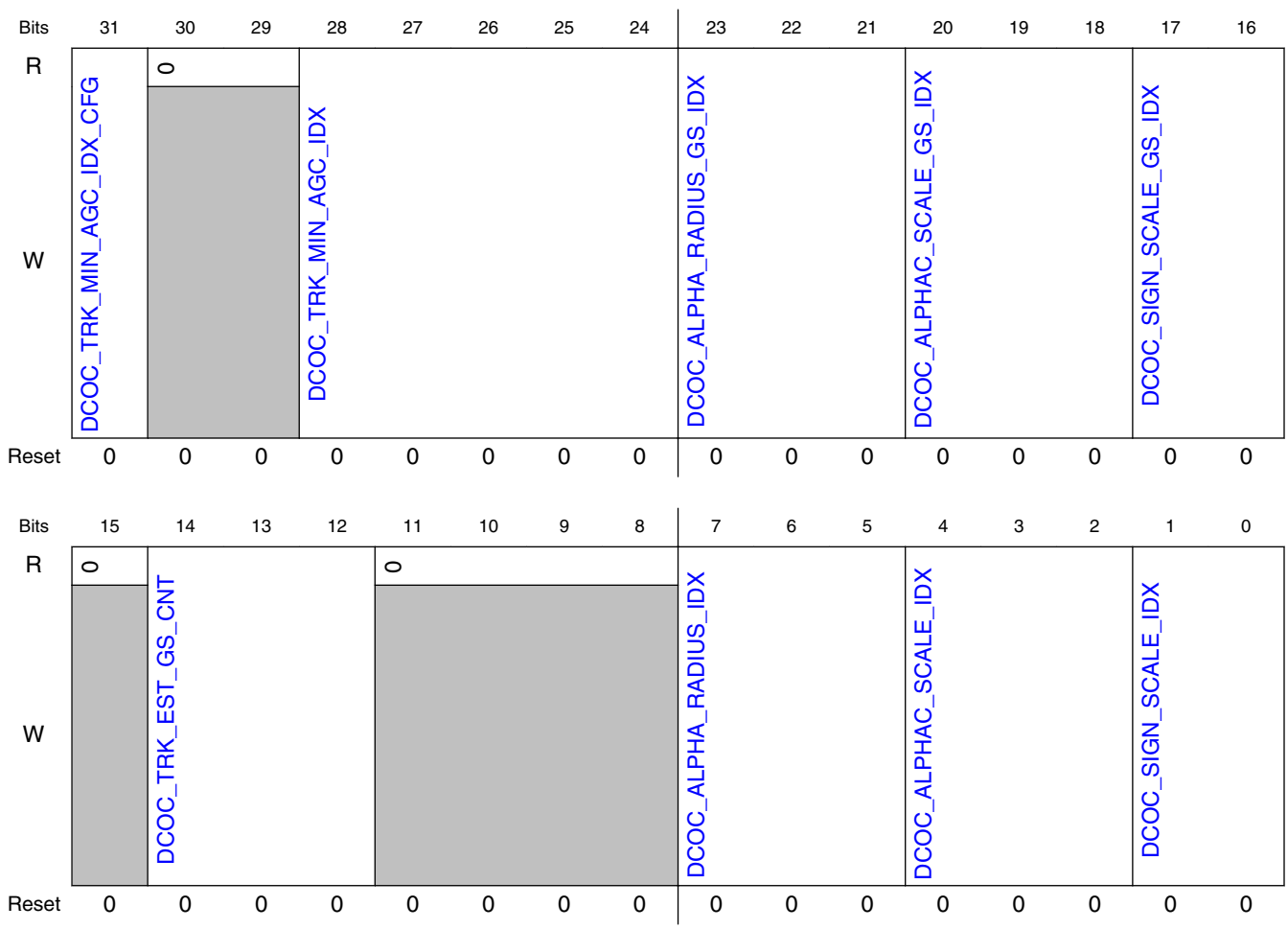
| Field | Function |
|---|---|
| | would resume operation. The tracking estimator is not disabled and the counter associated with DCOC_TRK_EST_GS_CNT is not reset when this condition occurs.<br>0b - Tracking corrections are enabled as determined by DCOC_CORRECT_SRC and DCOC_TRK_MIN_AGC_IDX.<br>1b - Tracking corrections are disabled when either the TZA or BBA lo peak detector asserts. |

## 64.3.3.3.1.11 DCOC Control 1 (DCOC_CTRL_1)

### *64.3.3.3.1.11.1 Offset*

| Register | Offset |
|---|---|
| DCOC_CTRL_1 | 28h |

### *64.3.3.3.1.11.2 Diagram*



### *64.3.3.3.1.11.3 Fields*

| Field | Function |
|---|---|
| 31<br>DCOC_TRK_MIN_AGC_IDX_CFG | DCOC_TRK_MIN_AGC_IDX Configuration<br>0b - Tracking is disabled when the AGC index is less than DCOC_TRK_MIN_AGC_IDX<br>1b - Tracking is enabled when AGC index is less than DCOC_TRK_MIN_AGC_IDX, but DCOC_CORR_DLY_ALT and DCOC_CORR_HOLD_TIME_ALT are used instead of DCOC_CORR_DLY and DCOC_CORR_HOLD_TIME |
| 30-29<br>— | Reserved. |
| 28-24<br>DCOC_TRK_MIN_AGC_IDX | DCOC Tracking Minimum AGC Table Index<br>Specifies the minimum AGC table index value at which tracking is enabled. E.g., if this is 5'd0, then tracking is enabled for all AGC gain table indexes (assuming DCOC_CORRECT_SRC is set) if this is 5'd20, then tracking is enabled for AGC gain table indexes 20-26 (assuming DCOC_CORRECT_SRC is set). |
| 23-21<br>DCOC_ALPHA_RADIUS_GS_IDX | Alpha-R Scaling for Gearshift<br>DCOC Alpha-R Scaling for Gearshift. Radius stepsize used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>000b - 1<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32<br>110b - 1/64<br>111b - Reserved |
| 20-18<br>DCOC_ALPHAC_SCALE_GS_IDX | DCOC Alpha-C Scaling for Gearshift<br>DCOC Alpha-C Scaling for Gearshift. I/Q center stepsize used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>000b - 1/2<br>001b - 1/4<br>010b - 1/8<br>011b - 1/16<br>100b - 1/32<br>101b - 1/64<br>110b - Reserved<br>111b - Reserved |
| 17-16<br>DCOC_SIGN_SCALE_GS_IDX | DCOC Sign Scaling for Gearshift<br>DCOC Sign Scaling for Gearshift. Sign()-based scaling factor used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>00b - 1/8<br>01b - 1/16<br>10b - 1/32<br>11b - 1/64 |
| 15<br>— | Reserved. |
| 14-12<br>DCOC_TRK_EST_GS_CNT | DCOC Tracking Estimator Gearshift Count<br>Specifies the number of corrections (periods of DCOC_CORR_DLY) before the tracking estimator switches from using the set of parameters {DCOC_ALPHA_RADIUS_IDX, DCOC_ALPHAC_SCALE_IDX, DCOC_SIGN_SCALE_IDX} to the set of gearshift parameters |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | {DCOC_ALPHA_RADIUS_GS_IDX, DCOC_ALPHAC_SCALE_GS_IDX, DCOC_SIGN_SCALE_GS_IDX}. If the value is 0, the set of gearshift parameters are not used. |
| 11-8<br>— | Reserved. |
| 7-5<br>DCOC_ALPHA_RADIUS_IDX | Alpha-R Scaling<br>DCOC Alpha-R Scaling. Radius stepsize used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>000b - 1<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32<br>110b - 1/64<br>111b - Reserved |
| 4-2<br>DCOC_ALPHAC_SCALE_IDX | DCOC Alpha-C Scaling<br>DCOC Alpha-C Scaling. I/Q center stepsize used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>000b - 1/2<br>001b - 1/4<br>010b - 1/8<br>011b - 1/16<br>100b - 1/32<br>101b - 1/64<br>110b - Reserved<br>111b - Reserved |
| 1-0<br>DCOC_SIGN_SCALE_IDX | DCOC Sign Scaling<br>DCOC Sign Scaling. Sign()-based scaling factor used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>00b - 1/8<br>01b - 1/16<br>10b - 1/32<br>11b - 1/64 |

## 64.3.3.3.1.12 DCOC DAC Initialization (DCOC_DAC_INIT)

### *64.3.3.3.1.12.1 Offset*

| Register | Offset |
|---|---|
| DCOC_DAC_INIT | 2Ch |

#### 64.3.3.3.1.12.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TZA_DCOC_INIT_Q | | | | | | | | TZA_DCOC_INIT_I | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | BBA_DCOC_INIT_Q | | | | | | 0 | | BBA_DCOC_INIT_I | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.12.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TZA_DCOC_INIT_Q | DCOC TZA Init Q<br>Value used for DCOC TZA Q channel DAC during calibration and for manual override. |
| 23-16<br>TZA_DCOC_INIT_I | DCOC TZA Init I<br>Value used for DCOC TZA I channel DAC during calibration and for manual override. |
| 15-14<br>— | Reserved. |
| 13-8<br>BBA_DCOC_INIT_Q | DCOC BBA Init Q<br>Value used for DCOC BBA Q channel DAC during calibration and for manual override. |
| 7-6<br>— | Reserved. |
| 5-0<br>BBA_DCOC_INIT_I | DCOC BBA Init I<br>Value used for DCOC BBA I channel DAC during calibration and for manual override. |

## 64.3.3.3.1.13 DCOC Digital Correction Manual Override (DCOC_DIG_MAN)

#### 64.3.3.3.1.13.1 Offset

| Register | Offset |
|---|---|
| DCOC_DIG_MAN | 30h |

#### 64.3.3.3.1.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DIG_DCOC_INIT_Q | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DIG_DCOC_INIT_I | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.13.3 Fields

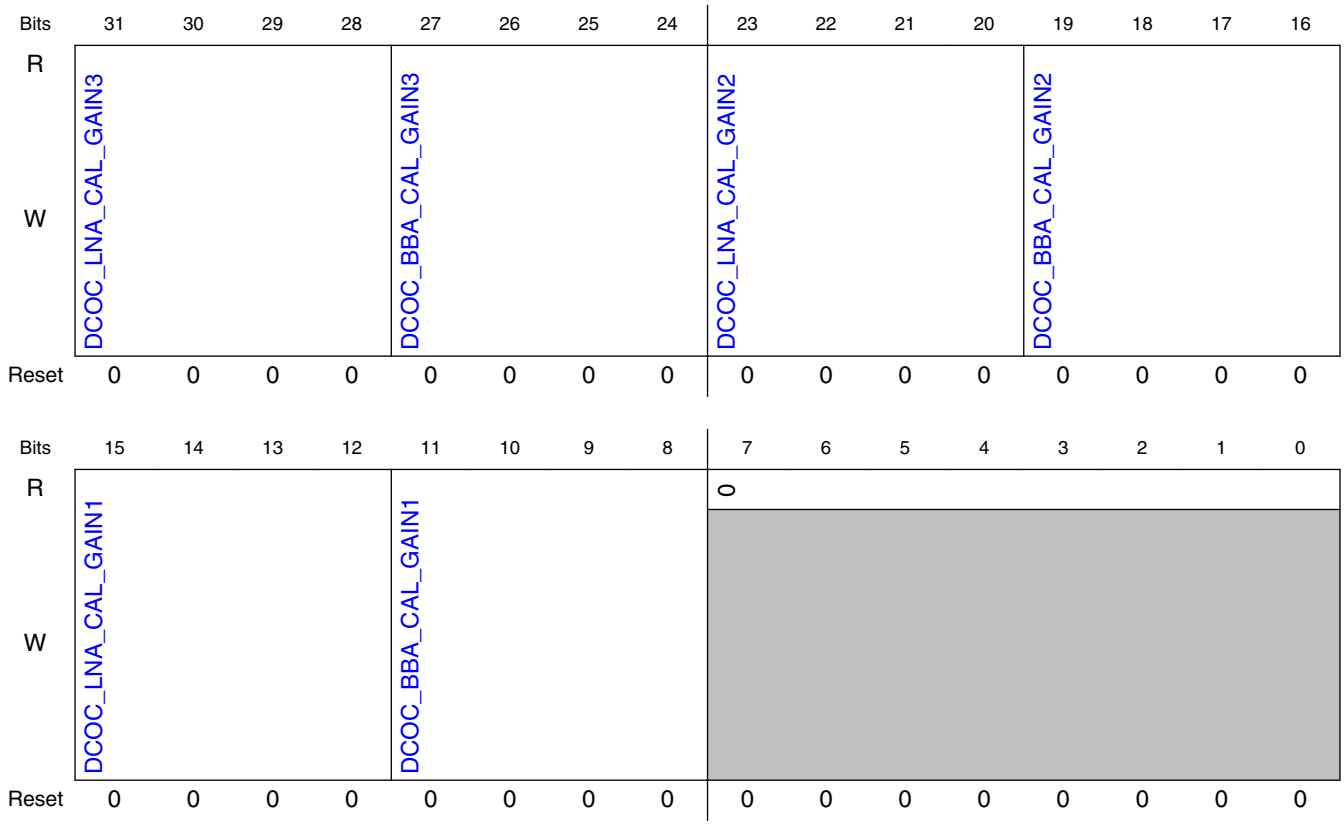
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DIG_DCOC_INIT_Q | DCOC DIG Init Q<br>Manual override for DCOC DIG Q channel correction. Value to be subtracted from downsampled Q channel. Used when DCOC_MAN=1. |
| 15-12<br>— | Reserved. |
| 11-0<br>DIG_DCOC_INIT_I | DCOC DIG Init I<br>Manual override for DCOC DIG I channel correction. Value to be subtracted from downsampled I channel. Used when DCOC_MAN=1. |

### 64.3.3.3.1.14 DCOC Calibration Gain (DCOC_CAL_GAIN)

#### 64.3.3.3.1.14.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_GAIN | 34h |

### 64.3.3.3.1.14.2 Diagram




### 64.3.3.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-28<br>DCOC_LNA_CAL_GAIN3 | DCOC LNA Calibration Gain 3<br>The LNA gain index used for the 3rd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 27-24<br>DCOC_BBA_CAL_GAIN3 | DCOC BBA Calibration Gain 3<br>The BBA gain index used for the 3rd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 23-20<br>DCOC_LNA_CAL_GAIN2 | DCOC LNA Calibration Gain 2<br>The LNA gain index used for the 2nd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 19-16<br>DCOC_BBA_CAL_GAIN2 | DCOC BBA Calibration Gain 2<br>The BBA gain index used for the 2nd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 15-12<br>DCOC_LNA_CAL_GAIN1 | DCOC LNA Calibration Gain 1<br>The LNA gain index used for the 1st DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 11-8 | DCOC BBA Calibration Gain 1 |

*Table continues on the next page...*

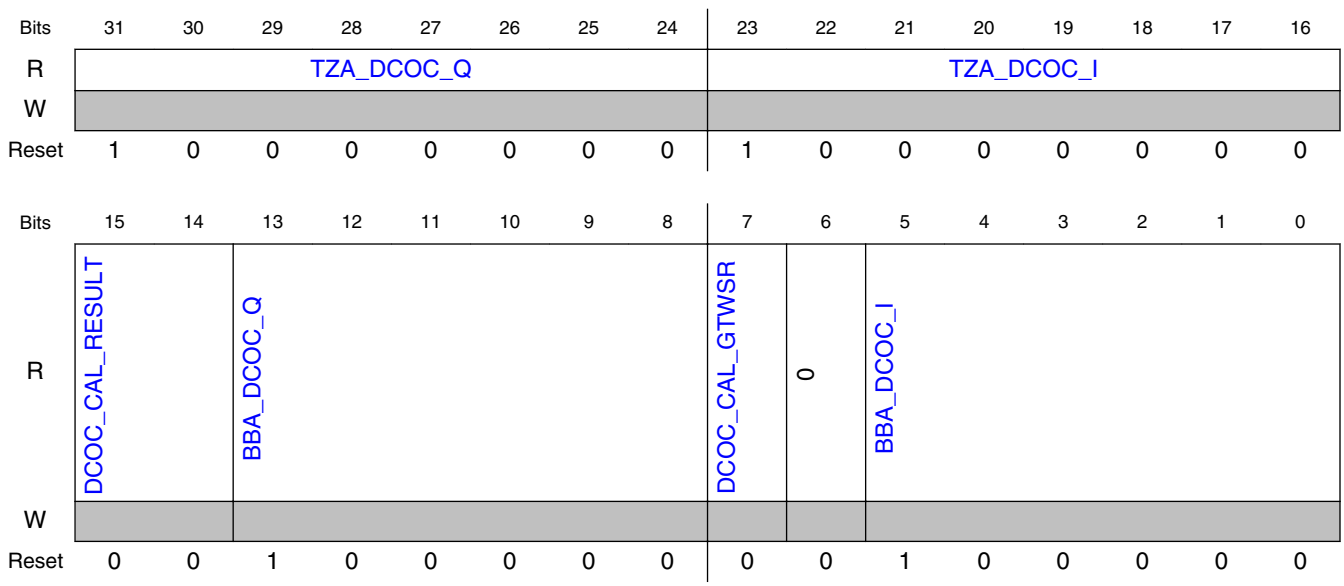
| Field | Function |
|---|---|
| DCOC_BBA_CAL_GAIN1 | The BBA gain index used for the 1st DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 7-0<br>— | Reserved. |

## 64.3.3.3.1.15 DCOC Status (DCOC_STAT)

### *64.3.3.3.1.15.1 Offset*

| Register | Offset |
|---|---|
| DCOC_STAT | 38h |

### *64.3.3.3.1.15.2 Diagram*




### *64.3.3.3.1.15.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>TZA_DCOC_Q | DCOC TZA DAC Q<br>Current TZA DAC setting for Q channel. Note that the TZA DACs have a bias of 0x80; 0x0 represents the most negative DC offset, 0x80 represents a DC offset of 0, and 0xFF represents the most postiive DC offset. This is provided for debug and characterization purposes only. |
| 23-16 | DCOC TZA DAC I |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TZA_DCOC_I | Current TZA DAC setting for I channel. Note that the TZA DACs have a bias of 0x80; 0x0 represents the most negative DC offset, 0x80 represents a DC offset of 0, and 0xFF represents the most postiive DC offset. This is provided for debug and characterization purposes only. |
| 15-14<br>DCOC_CAL_RESULT | DCOC_CAL_RESULT<br>Indicates the calibration result. Only applicable when DCOC_CTRL_0[DCOC_CAL_CHECK_EN] is set.<br>00b - Calibration checks failed. DCOC_OFFSET_n tables not updated.<br>01b - Calibration checks neither passed nor failed, DCOC_OFFSET_n tables not updated.<br>10b - Calibration checks neither passed nor failed, DCOC_OFFSET_n tables updated since no previous Pass condition has occurred since the last radio reset.<br>11b - Calibration checks passed. DCOC_OFFSET_n tables updated |
| 13-8<br>BBA_DCOC_Q | DCOC BBA DAC Q<br>Current BBA DAC setting for Q channel. Note that the BBA DACs have a bias of 0x20; 0x0 represents the most negative DC offset, 0x20 represents a DC offset of 0, and 0x3F represents the most postiive DC offset. This is provided for debug and characterization purposes only. |
| 7<br>DCOC_CAL_GTWSR | DCOC calibration Good Table Written Since Reset<br>Indicates whether a Passing calibration check has occurred since the last reset. Only applicable when DCOC_CTRL_0[DCOC_CAL_CHECK_EN] is set.<br>0b - A Passing calibration result has not occurred since the last radio reset.<br>1b - A Passing calibration result has occurred since the last radio reset. |
| 6<br>— | Reserved. |
| 5-0<br>BBA_DCOC_I | DCOC BBA DAC I<br>Current BBA DAC setting for I channel. Note that the BBA DACs have a bias of 0x20; 0x0 represents the most negative DC offset, 0x20 represents a DC offset of 0, and 0x3F represents the most postiive DC offset. This is provided for debug and characterization purposes only. |

## 64.3.3.3.1.16 DCOC DC Estimate (DCOC_DC_EST)

### *64.3.3.3.1.16.1 Offset*

| Register | Offset |
|---|---|
| DCOC_DC_EST | 3Ch |

#### 64.3.3.3.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DC_EST_Q | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DC_EST_I | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.16.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DC_EST_Q | DCOC DC Estimate Q<br>Reflects the current DCOC DC tracking estimate for Q channel. Format s11. Used when DCOC_CORRECT_SRC=1. This is provided for debug and characterization purposes only. |
| 15-12<br>— | Reserved. |
| 11-0<br>DC_EST_I | DCOC DC Estimate I<br>Reflects the current DCOC DC tracking estimate for I channel. Format s11. Used when DCOC_CORRECT_SRC=1. This is provided for debug and characterization purposes only. |

### 64.3.3.3.1.17 DCOC Calibration Reciprocals (DCOC_CAL_RCP)

#### 64.3.3.3.1.17.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_RCP | 40h |

#### 64.3.3.3.1.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ALPHA_CALC_RECIP | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_TMP_CALC_RECIP | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.17.3 Fields

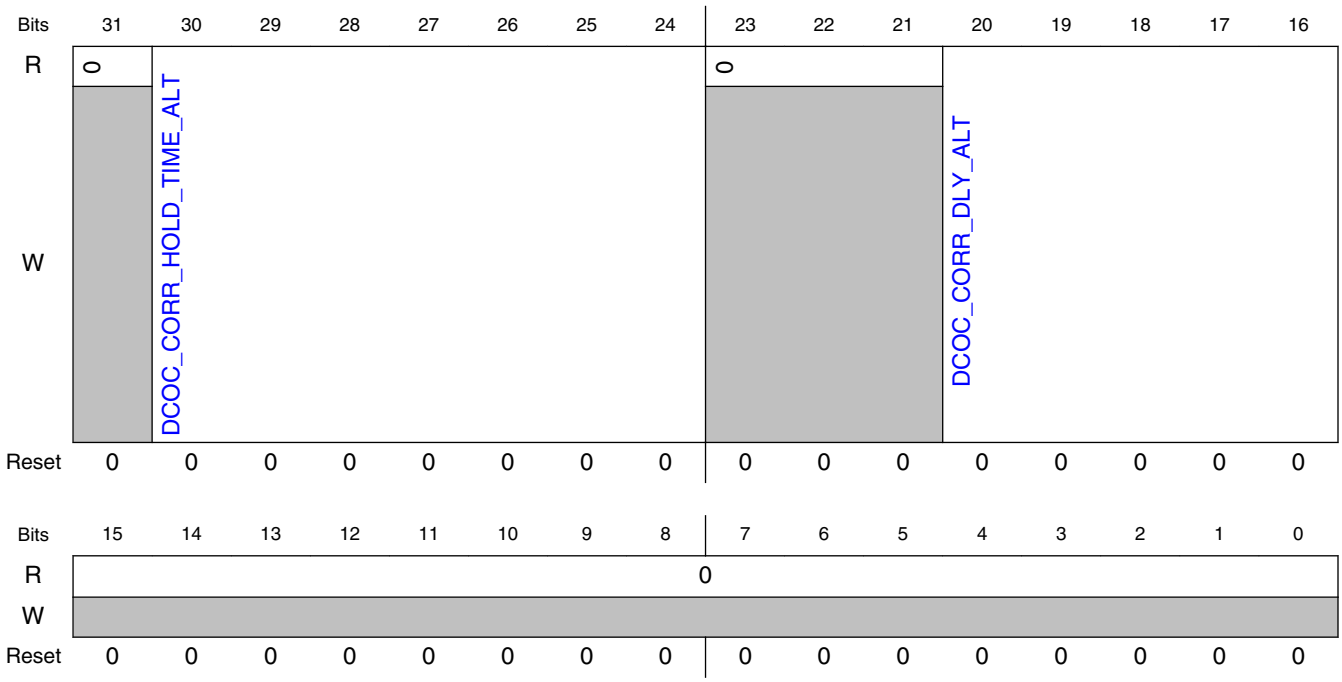
| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>ALPHA_CALC_RECIP | Alpha Calculation Reciprocal<br>DCOC Alpha calculation reciprocal (format: u.11). This is used in DCOC calibration calculation of the alpha DC component. It is defined as:<br>$1.0/((G_{L_HI} - G_{L_LO})*G_{B_LO})$<br>This is stored as with 11 fractional bits, so program the value<br>$round([1.0/((G_{L_HI} - G_{L_LO})*G_{B_LO})]*2^{11})$. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_TMP_CALC_RECIP | DCOC Calculation Reciprocal<br>DCOC_tmp calculation reciprocal (format: u1.10). This is used in DCDC calibration calculation. It is defined as<br>$1.0/(G_{B_HI} - G_{B_LO})$<br>This is stored with 10 fractional bits, so program the value $round([1.0/(G_{B_HI} - G_{B_LO})]*2^{10})$. |

## 64.3.3.3.1.18 DCOC Control 2 (DCOC_CTRL_2)

#### 64.3.3.3.1.18.1 Offset

| Register | Offset |
|---|---|
| DCOC_CTRL_2 | 44h |

### 64.3.3.3.1.18.2 Diagram




### 64.3.3.3.1.18.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>DCOC_CORR_HOLD_TIME_ALT | DCOC Correction Hold Time Alternate<br>Delay from last gain change to freezing the DC correction. This is used instead of DCOC_CORR_HOLD_TIME depending on programming of DCOC_TRK_MIN_AGC_IDX_CFG and the current AGC index.<br>0000000b - Reserved<br>0000001-1111110b - For a 32MHz reference clock, this is the delay in microseconds; for other reference clock frequencies, the delay is scaled accordingly.<br>1111111b - The DC correction is not frozen. |
| 23-21<br>— | Reserved. |
| 20-16<br>DCOC_CORR_DLY_ALT | DCOC Correction Delay Alternate<br>Wait time between corrections. This is used instead of DCOC_CORR_DLY depending on programming of DCOC_TRK_MIN_AGC_IDX_CFG and the current AGC index.<br>00000b - Reserved<br>00001-11111b - For a 32MHz reference clock, this is the wait time in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 15-0<br>— | Reserved. |

## 64.3.3.3.1.19 IQMC Control (IQMC_CTRL)

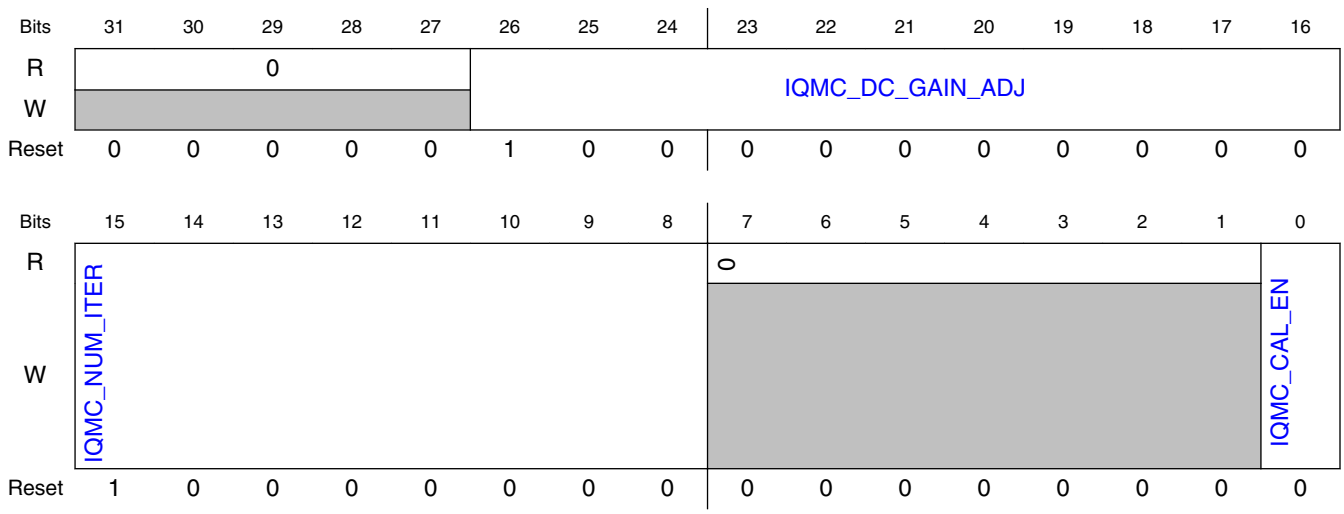
### *64.3.3.3.1.19.1 Offset*

| Register | Offset |
|---|---|
| IQMC_CTRL | 48h |

### *64.3.3.3.1.19.2 Function*

IQMC Control register. This register can only be accessed when the radio oscillator clock is active.

### *64.3.3.3.1.19.3 Diagram*




### *64.3.3.3.1.19.4 Fields*

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>IQMC_DC_GAIN_ADJ | IQ Mismatch Correction DC Gain Coeff<br>I/Q mismatch correction DC gain coefficient. This gain value is used only during DCOC calibration. It is not calculated during the IQMC calibration sequence, so it must be written by software. The format is u1.10; the reset value of 0x400 corresponds to a DC gain coefficient of 1.0. |
| 15-8<br>IQMC_NUM_ITER | IQ Mismatch Cal Num Iter<br>Number of iterations for IQ Mismatch Calibration. |
| 7-1 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 0<br>IQMC_CAL_EN | IQ Mismatch Cal Enable<br>Enables IQ mismatch calibration. This bit is self-clearing; it will clear automatically at the end of the calibration sequence. |

## 64.3.3.3.1.20 IQMC Calibration (IQMC_CAL)

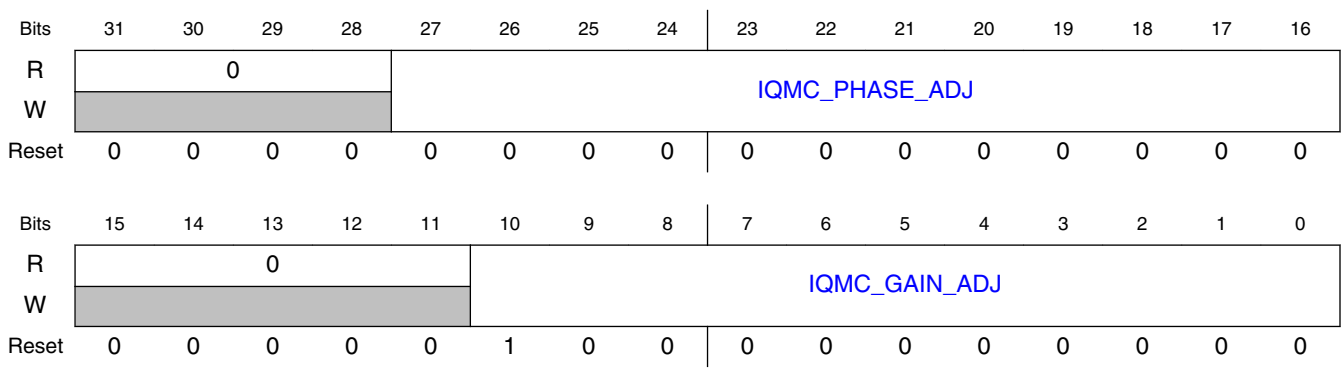
### *64.3.3.3.1.20.1 Offset*

| Register | Offset |
|---|---|
| IQMC_CAL | 4Ch |

### *64.3.3.3.1.20.2 Function*

IQMC Calibration register. This register can only be accessed when the radio oscillator clock is active.

### *64.3.3.3.1.20.3 Diagram*




### *64.3.3.3.1.20.4 Fields*

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>IQMC_PHASE_ADJ | IQ Mismatch Correction Phase Coeff<br>I/Q mismatch correction phase coefficient. This value is updated automatically at the end of the IQMC calibration sequence. It can also be written by software. The format is signed, with the maximum positive |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | value 0x7ff corresponding to a phase coefficient of 0.25, and the maximum negative value 0x800 corresponding to -0.25. The reset value of 0x000 corresponds to a phase coefficient of 0. |
| 15-11<br>— | Reserved. |
| 10-0<br>IQMC_GAIN_ADJ | IQ Mismatch Correction Gain Coeff<br>I/Q mismatch correction gain coefficient. This value is updated automatically at the end of the IQMC calibration sequence. It can also be written by software. The format is u1.10. The reset value of 0x400 corresponds to a gain coefficient of 1.0. |

## 64.3.3.3.1.21 LNA_GAIN Step Values 3..0 (LNA_GAIN_VAL_3_0)

### *64.3.3.3.1.21.1 Offset*

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_3_0 | 50h |

### *64.3.3.3.1.21.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_3 | | | | | | | | LNA_GAIN_VAL_2 | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_1 | | | | | | | | LNA_GAIN_VAL_0 | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

### *64.3.3.3.1.21.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>LNA_GAIN_VAL_3 | LNA_GAIN step 3<br>Gain for LNA gain Step 3, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 23-16<br>LNA_GAIN_VAL_2 | LNA_GAIN step 2<br>Gain for LNA gain Step 2, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 15-8 | LNA_GAIN step 1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LNA_GAIN_VAL_1 | Gain for LNA gain Step 1, used in RSSI calculation. Unsigned, 4*(measured_gain_dB+16). |
| 7-0<br>LNA_GAIN_VAL_0 | LNA_GAIN step 0<br>Gain for LNA gain Step 0, used in RSSI calculation. Unsigned, 4*(measured_gain_dB+16). |

## 64.3.3.3.1.22 LNA_GAIN Step Values 7..4 (LNA_GAIN_VAL_7_4)

### *64.3.3.3.1.22.1 Offset*

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_7_4 | 54h |

### *64.3.3.3.1.22.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_7 | | | | | | | | LNA_GAIN_VAL_6 | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_5 | | | | | | | | LNA_GAIN_VAL_4 | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

### *64.3.3.3.1.22.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>LNA_GAIN_VAL_7 | LNA_GAIN step 7<br>Gain for LNA gain Step 7, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 23-16<br>LNA_GAIN_VAL_6 | LNA_GAIN step 6<br>Gain for LNA gain Step 6, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 15-8<br>LNA_GAIN_VAL_5 | LNA_GAIN step 5<br>Gain for LNA gain Step 5, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 7-0 | LNA_GAIN step 4 |

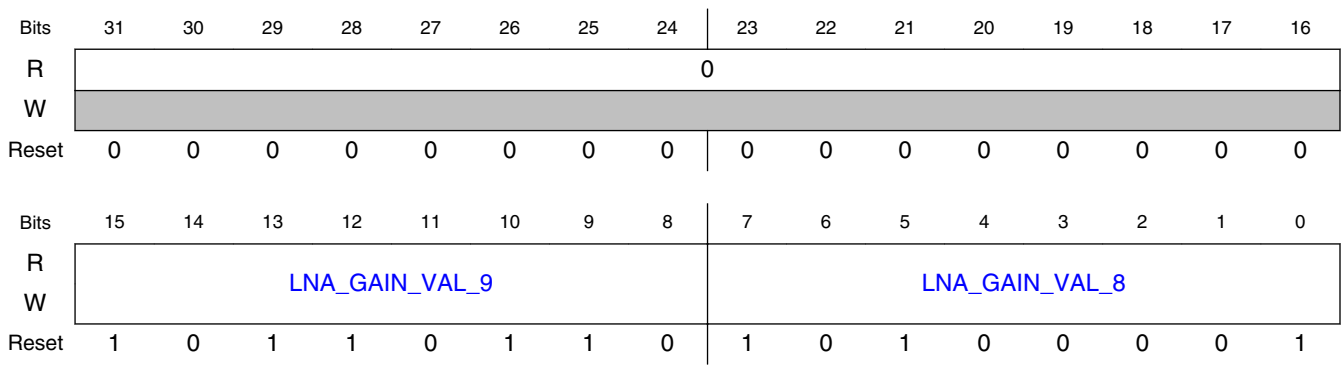
| Field | Function |
|---|---|
| LNA_GAIN_VAL_4 | Gain for LNA gain Step 4, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

## 64.3.3.3.1.23 LNA_GAIN Step Values 8 (LNA_GAIN_VAL_8)

### *64.3.3.3.1.23.1 Offset*

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_8 | 58h |

### *64.3.3.3.1.23.2 Diagram*




### *64.3.3.3.1.23.3 Fields*

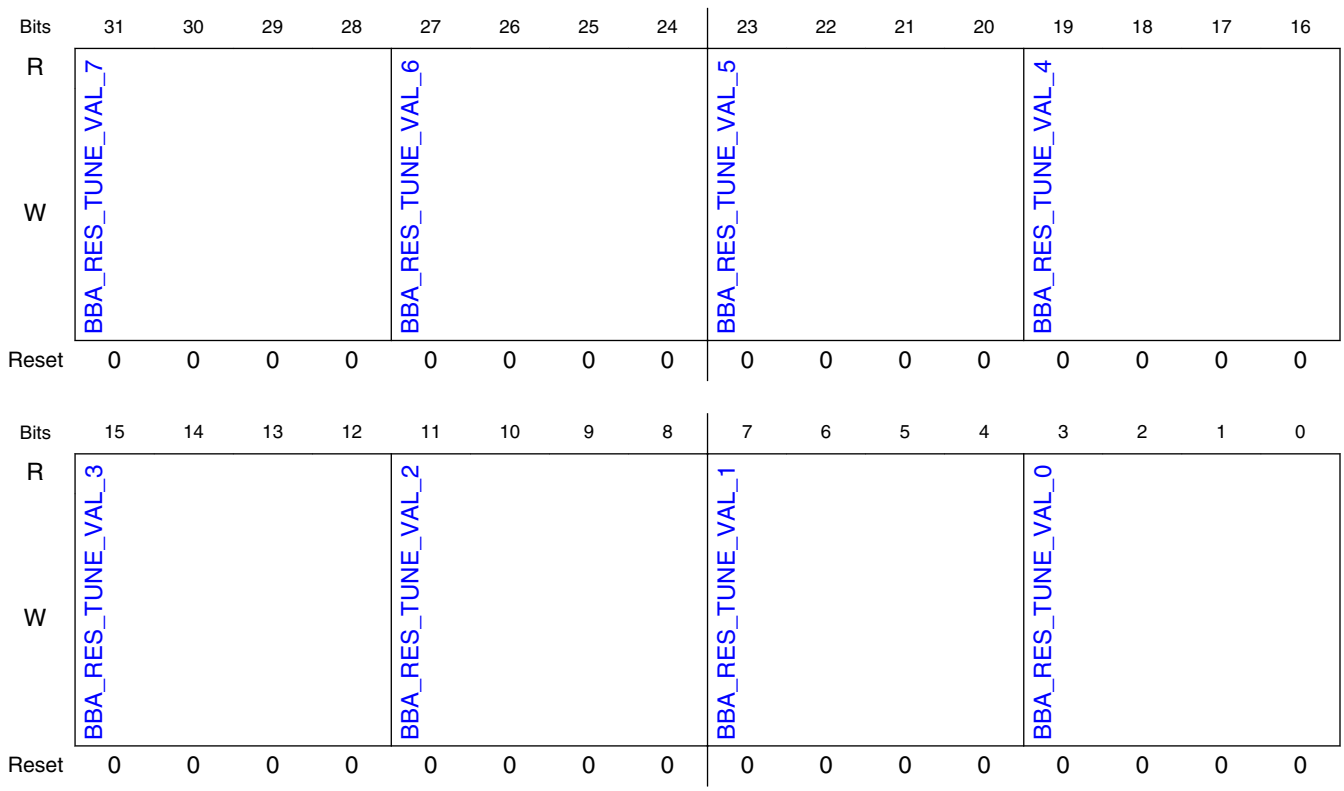
| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8<br>LNA_GAIN_VAL_9 | LNA_GAIN step 9<br>Gain for LNA gain Step 9, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 7-0<br>LNA_GAIN_VAL_8 | LNA_GAIN step 8<br>Gain for LNA gain Step 8, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

## 64.3.3.3.1.24 BBA Resistor Tune Values 7..0 (BBA_RES_TUNE_VAL_7_0)

### 64.3.3.3.1.24.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_VAL_7_0 | 5Ch |

### 64.3.3.3.1.24.2 Diagram




### 64.3.3.3.1.24.3 Fields

| Field | Function |
|---|---|
| 31-28 | BBA Resistor Tune Step 7 |
| BBA_RES_TUNE_VAL_7 | Gain offset for BBA gain Step 7, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 27-24 | BBA Resistor Tune Step 6 |
| BBA_RES_TUNE_VAL_6 | Gain offset for BBA gain Step 6, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 23-20 | BBA Resistor Tune Step 5 |
| BBA_RES_TUNE_VAL_5 | Gain offset for BBA gain Step 5, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 19-16 | BBA Resistor Tune Step 4 |

*Table continues on the next page...*

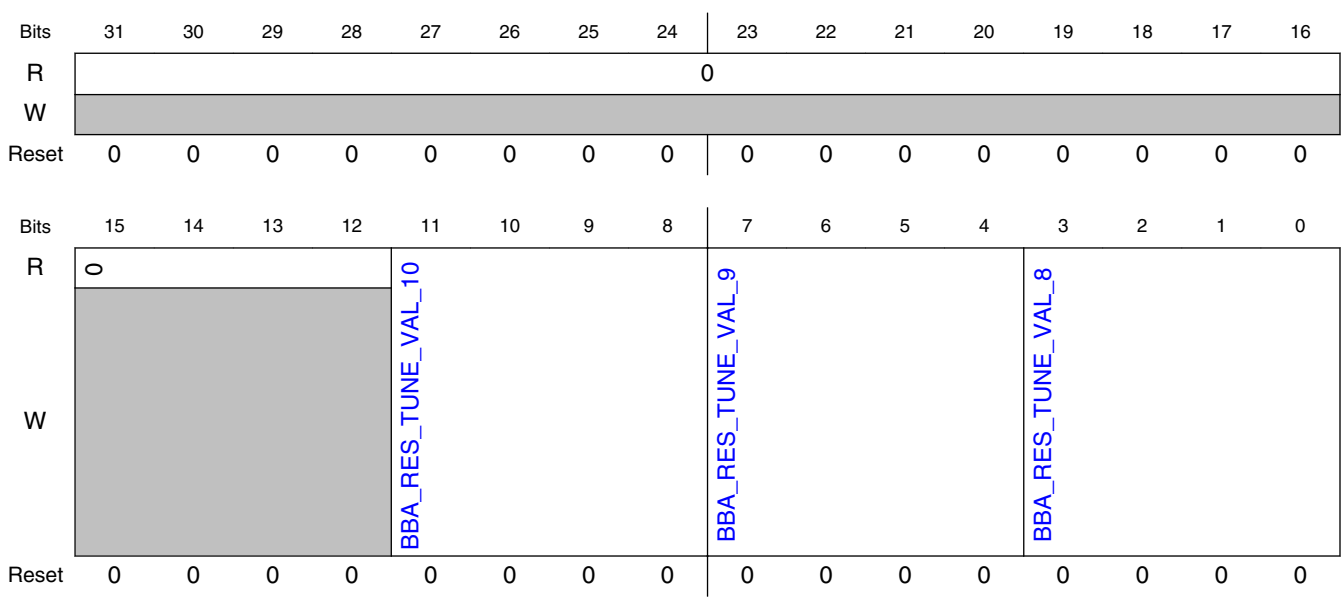
| Field | Function |
|---|---|
| BBA_RES_TUNE_VAL_4 | Gain offset for BBA gain Step 4, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 15-12<br>BBA_RES_TUNE_VAL_3 | BBA Resistor Tune Step 3<br>Gain offset for BBA gain Step 3, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 11-8<br>BBA_RES_TUNE_VAL_2 | BBA Resistor Tune Step 2<br>Gain offset for BBA gain Step 2, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 7-4<br>BBA_RES_TUNE_VAL_1 | BBA Resistor Tune Step 1<br>Gain offset for BBA gain Step 1, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 3-0<br>BBA_RES_TUNE_VAL_0 | BBA Resistor Tune Step 0<br>Gain offset for BBA gain Step 0, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |

## 64.3.3.3.1.25 BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_VAL_10_8)

### *64.3.3.3.1.25.1 Offset*

| Register | Offset |
|---|---|
| BBA_RES_TUNE_VAL_10_8 | 60h |

### *64.3.3.3.1.25.2 Diagram*



#### 64.3.3.3.1.25.3 Fields

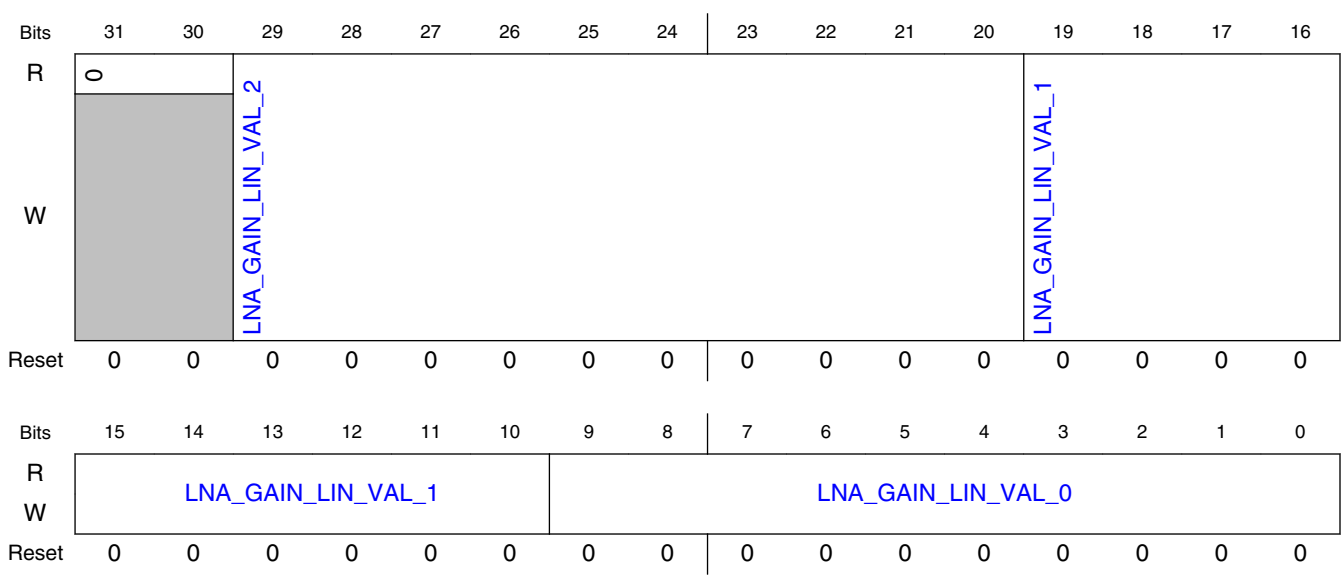
| Field | Function |
|---|---|
| 31-12<br>— | Reserved. |
| 11-8<br>BBA_RES_TUNE_VAL_10 | BBA Resistor Tune Step 10<br>Gain offset for BBA gain Step 10, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 7-4<br>BBA_RES_TUNE_VAL_9 | BBA Resistor Tune Step 9<br>Gain offset for BBA gain Step 9, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 3-0<br>BBA_RES_TUNE_VAL_8 | BBA Resistor Tune Step 8<br>Gain offset for BBA gain Step 8, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |

## 64.3.3.3.1.26 LNA Linear Gain Values 2..0 (LNA_GAIN_LIN_VAL_2_0)

#### 64.3.3.3.1.26.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_2_0 | 64h |

#### 64.3.3.3.1.26.2 Diagram



#### 64.3.3.3.1.26.3 Fields

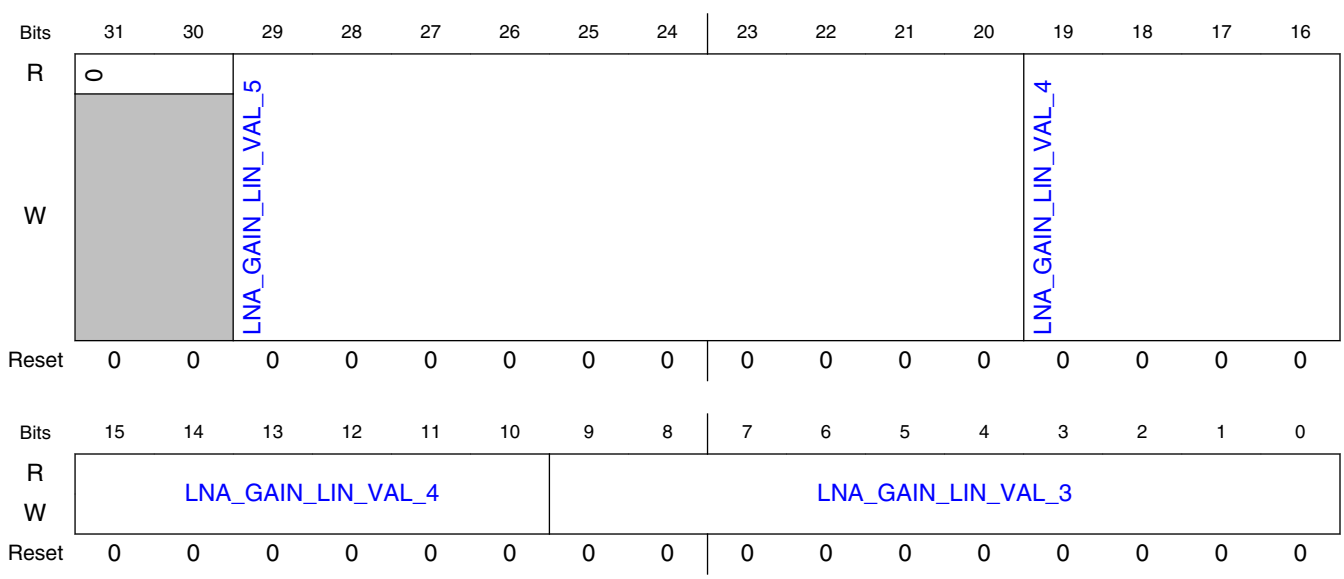
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_2 | LNA Linear Gain Step 2<br>LNA linear gain value for index 2, e.g. nominal value is 10^(2.2/20). Stored with 2 fractional bits, e.g. round([10^(2.2/20)]*2^2) = 5decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_1 | LNA Linear Gain Step 1<br>LNA linear gain value for index 1, e.g. nominal value is 10^(-3.5/20). Stored with 2 fractional bits, e.g. round([10^(-3.5/20)]*2^2) = 3decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_0 | LNA Linear Gain Step 0<br>LNA linear gain value for index 0, e.g. nominal value is 10^(-8.6/20). Stored with 2 fractional bits, e.g. round([10^(-8.6/20)]*2^2) = 1decimal. Format (8.2). |

## 64.3.3.3.1.27 LNA Linear Gain Values 5..3 (LNA_GAIN_LIN_VAL_5_3)

#### 64.3.3.3.1.27.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_5_3 | 68h |

#### 64.3.3.3.1.27.2 Diagram



#### *64.3.3.3.1.27.3 Fields*

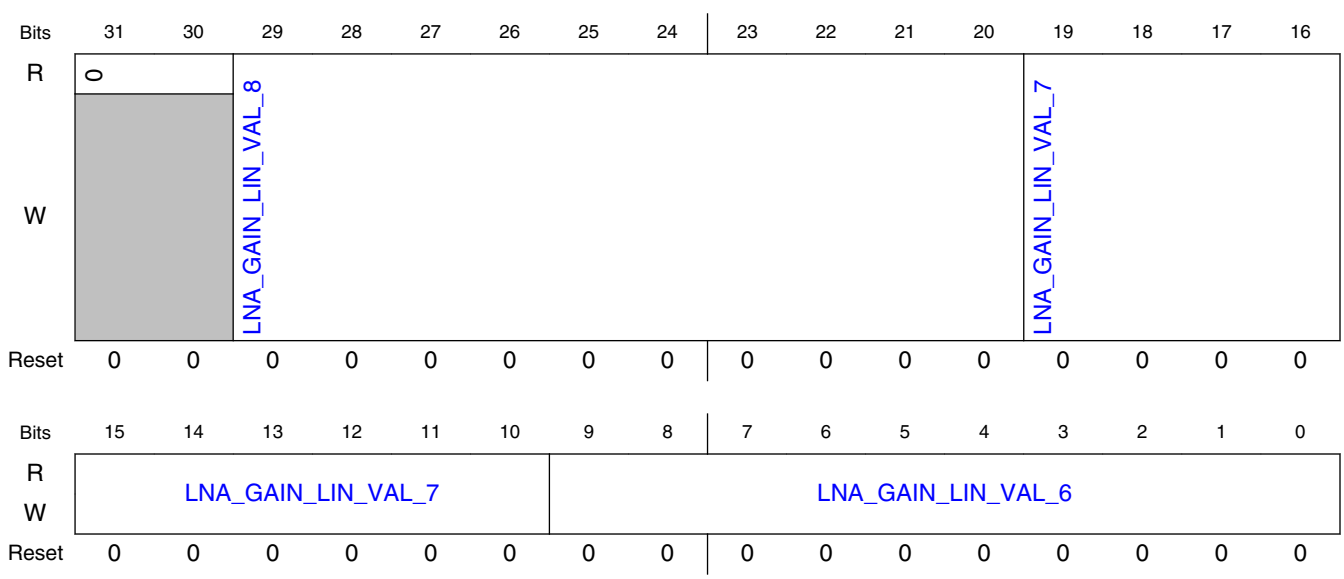
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_5 | LNA Linear Gain Step 5<br>LNA linear gain value for index 5, e.g. nominal value is 10^(22.7/20). Stored with 2 fractional bits, e.g. round([10^(22.7/20)]*2^2) = 55decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_4 | LNA Linear Gain Step 4<br>LNA linear gain value for index 4, e.g. nominal value is 10^(19.8/20). Stored with 2 fractional bits, e.g. round([10^(19.8/20)]*2^2) = 39decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_3 | LNA Linear Gain Step 3<br>LNA linear gain value for index 3, e.g. nominal value is 10^(13.9/20). Stored with 2 fractional bits, e.g. round([10^(13.9/20)]*2^2) = 20decimal. Format (8.2). |

## 64.3.3.3.1.28 LNA Linear Gain Values 8..6 (LNA_GAIN_LIN_VAL_8_6)

#### *64.3.3.3.1.28.1 Offset*

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_8_6 | 6Ch |

#### *64.3.3.3.1.28.2 Diagram*



#### 64.3.3.3.1.28.3 Fields

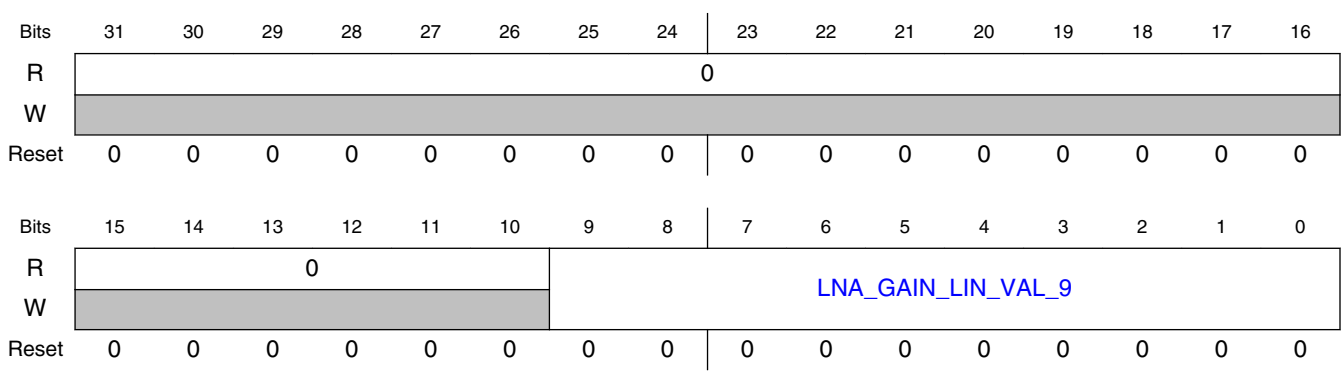
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_8 | LNA Linear Gain Step 8<br>LNA linear gain value for index 8, e.g. nominal value is 10^(39.9/20). Stored with 2 fractional bits, e.g. round([10^(39.9/20)]*2^2) = 396decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_7 | LNA Linear Gain Step 7<br>LNA linear gain value for index 7, e.g. nominal value is 10^(34.4/20). Stored with 2 fractional bits, e.g. round([10^(34.4/20)]*2^2) = 210decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_6 | LNA Linear Gain Step 6<br>LNA linear gain value for index 6, e.g. nominal value is 10^(28.6/20). Stored with 2 fractional bits, e.g. round([10^(28.6/20)]*2^2) = 108decimal. Format (8.2). |

## 64.3.3.3.1.29 LNA Linear Gain Values 9 (LNA_GAIN_LIN_VAL_9)

#### 64.3.3.3.1.29.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_9 | 70h |

#### 64.3.3.3.1.29.2 Diagram




#### 64.3.3.3.1.29.3 Fields

| Field | Function |
|---|---|
| 31-10 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 9-0<br>LNA_GAIN_LIN_VAL_9 | LNA Linear Gain Step 9<br>LNA linear gain value for index 9, e.g. nominal value is 10^(45.4/20). Stored with 2 fractional bits, e.g. round([10^(45.4/20)]*2^2) = 744decimal. Format (8.2). |

## 64.3.3.3.1.30 BBA Resistor Tune Values 3..0 (BBA_RES_TUNE_LIN_VAL_3_0)

### *64.3.3.3.1.30.1 Offset*

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_3_0 | 74h |

### *64.3.3.3.1.30.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_3 | | | | | | | | BBA_RES_TUNE_LIN_VAL_2 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_1 | | | | | | | | BBA_RES_TUNE_LIN_VAL_0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.30.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>BBA_RES_TUNE_LIN_VAL_3 | BBA Resistor Tune Linear Gain Step 3<br>BBA linear gain value for index 3 (format: u5.3). Nominal value is 10^(9/20). Stored with 3 fractional bits, e.g. round([10^(9/20)]*2^3) = 23decimal |
| 23-16<br>BBA_RES_TUNE_LIN_VAL_2 | BBA Resistor Tune Linear Gain Step 2<br>BBA linear gain value for index 2 (format: u5.3). Nominal value is 10^(6/20). Stored with 3 fractional bits, e.g. round([10^(6/20)]*2^3) = 16decimal |
| 15-8<br>BBA_RES_TUNE_LIN_VAL_1 | BBA Resistor Tune Linear Gain Step 1<br>BBA linear gain value for index 1 (format: u5.3). Nominal value is 10^(3/20). Stored with 3 fractional bits, e.g. round([10^(3/20)]*2^3) = 11decimal |
| 7-0 | BBA Resistor Tune Linear Gain Step 0 |

| Field | Function |
|---|---|
| BBA_RES_TUNE_LIN_VAL_0 | BBA linear gain value for index 0 (format: u5.3). Nominal value is 10^(0/20). Stored with 3 fractional bits, e.g. round([10^(0/20)]*2^3) = 8decimal |

## 64.3.3.3.1.31 BBA Resistor Tune Values 7..4 (BBA_RES_TUNE_LIN_VAL_7_4)

### *64.3.3.3.1.31.1 Offset*

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_7_4 | 78h |

### *64.3.3.3.1.31.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_7 | | | | | | | | BBA_RES_TUNE_LIN_VAL_6 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_5 | | | | | | | | BBA_RES_TUNE_LIN_VAL_4 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.31.3 Fields*

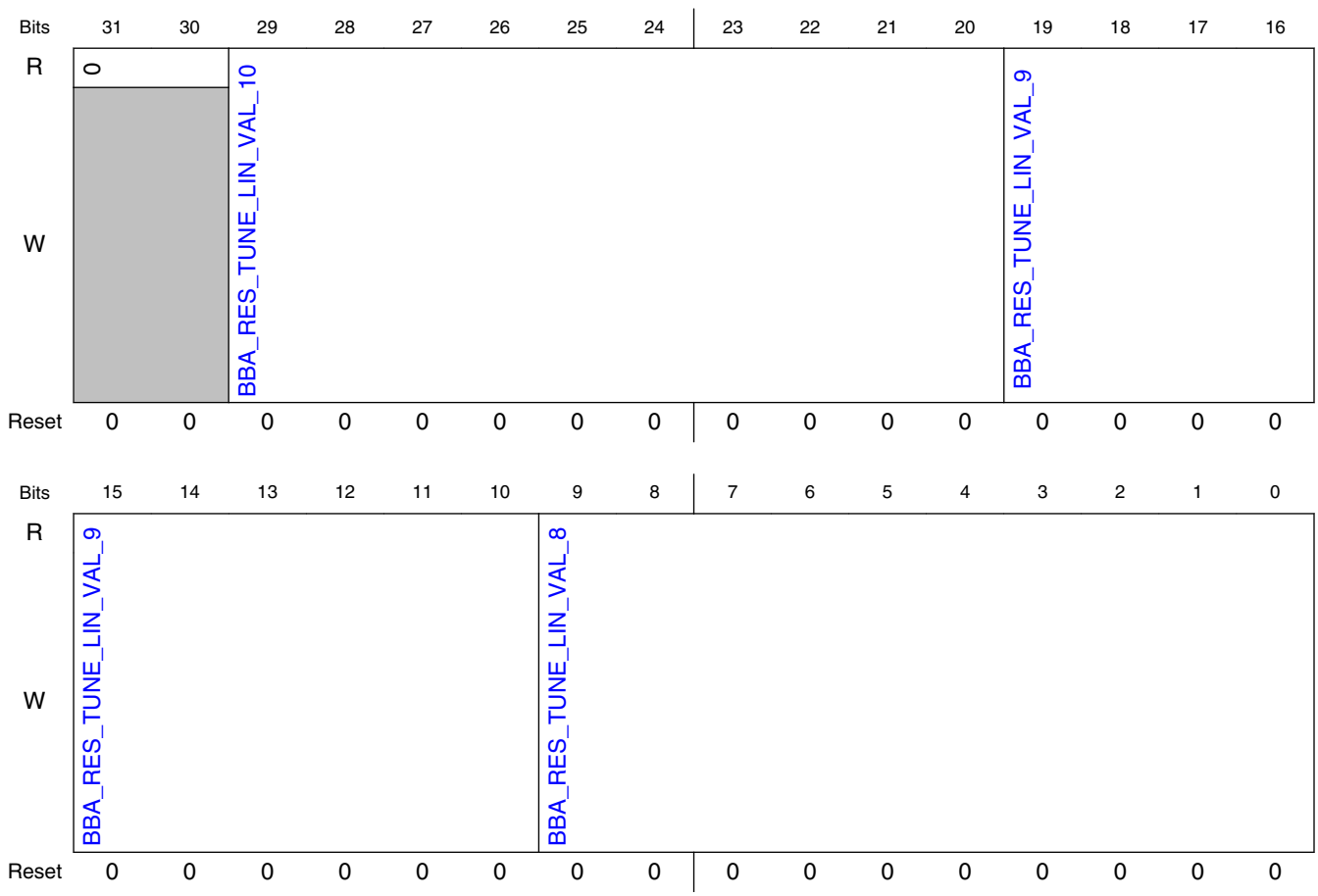
| Field | Function |
|---|---|
| 31-24 | BBA Resistor Tune Linear Gain Step 7 |
| BBA_RES_TUNE_LIN_VAL_7 | BBA linear gain value for index 7 (format: u6.2). Nominal value is 10^(21/20). Stored with 2 fractional bits, e.g. round([10^(21/20)]*2^2) = 45 decimal |
| 23-16 | BBA Resistor Tune Linear Gain Step 6 |
| BBA_RES_TUNE_LIN_VAL_6 | BBA linear gain value for index 6 (format: u5.3). Nominal value is 10^(18/20). Stored with 3 fractional bits, e.g. round([10^(18/20)]*2^3) = 64decimal |
| 15-8 | BBA Resistor Tune Linear Gain Step 5 |
| BBA_RES_TUNE_LIN_VAL_5 | BBA linear gain value for index 5 (format: u5.3). Nominal value is 10^(15/20). Stored with 3 fractional bits, e.g. round([10^(15/20)]*2^3) = 45decimal |
| 7-0 | BBA Resistor Tune Linear Gain Step 4 |
| BBA_RES_TUNE_LIN_VAL_4 | BBA linear gain value for index 4 (format: u5.3). Nominal value is 10^(12/20). Stored with 3 fractional bits, e.g. round([10^(12/20)]*2^3) = 32decimal |

## 64.3.3.3.1.32 BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_LIN_VAL_10_8)

### *64.3.3.3.1.32.1 Offset*

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_10_8 | 7Ch |

### *64.3.3.3.1.32.2 Diagram*




### *64.3.3.3.1.32.3 Fields*

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20 | BBA Resistor Tune Linear Gain Step 10 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BBA_RES_TUNE_LIN_VAL_10 | BBA linear gain value for index 10 (format: u7.3). Nominal value is 10^(30/20). Stored with 3 fractional bits, e.g. round([10^(30/20)]*2^3) = 253decimal |
| 19-10<br>BBA_RES_TUNE_LIN_VAL_9 | BBA Resistor Tune Linear Gain Step 9<br>BBA linear gain value for index 9 (format: u7.3). Nominal value is 10^(27/20). Stored with 3 fractional bits, e.g. round([10^(27/20)]*2^3) = 179decimal |
| 9-0<br>BBA_RES_TUNE_LIN_VAL_8 | BBA Resistor Tune Linear Gain Step 8<br>BBA linear gain value for index 8 (format: u7.3). Nominal value is 10^(24/20). Stored with 3 fractional bits, e.g. round([10^(24/20)]*2^3) = 127decimal |

## 64.3.3.3.1.33 AGC Gain Tables Step 03..00 (AGC_GAIN_TBL_03_00)

### *64.3.3.3.1.33.1 Offset*

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_03_00 | 80h |

### *64.3.3.3.1.33.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_03 | | | | BBA_GAIN_03 | | | | LNA_GAIN_02 | | | | BBA_GAIN_02 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_01 | | | | BBA_GAIN_01 | | | | LNA_GAIN_00 | | | | BBA_GAIN_00 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.33.3 Fields*

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_03 | LNA Gain 03<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_03 | BBA Gain 03<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_02 | LNA Gain 02<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19-16<br>BBA_GAIN_02 | BBA Gain 02<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_01 | LNA Gain 01<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_01 | BBA Gain 01<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_00 | LNA Gain 00<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_00 | BBA Gain 00<br>BBA GAIN. |

## 64.3.3.3.1.34 AGC Gain Tables Step 07..04 (AGC_GAIN_TBL_07_04)

### *64.3.3.3.1.34.1 Offset*

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_07_04 | 84h |

### *64.3.3.3.1.34.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_07 | | | | BBA_GAIN_07 | | | | LNA_GAIN_06 | | | | BBA_GAIN_06 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_05 | | | | BBA_GAIN_05 | | | | LNA_GAIN_04 | | | | BBA_GAIN_04 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.34.3 Fields*

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_07 | LNA Gain 07<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24 | BBA Gain 07 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BBA_GAIN_07 | BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_06 | LNA Gain 06<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_06 | BBA Gain 06<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_05 | LNA Gain 05<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_05 | BBA Gain 05<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_04 | LNA Gain 04<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_04 | BBA Gain 04<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## 64.3.3.3.1.35 AGC Gain Tables Step 11..08 (AGC_GAIN_TBL_11_08)

### *64.3.3.3.1.35.1 Offset*

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_11_08 | 88h |

### *64.3.3.3.1.35.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_11 | | | | BBA_GAIN_11 | | | | LNA_GAIN_10 | | | | BBA_GAIN_10 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_09 | | | | BBA_GAIN_09 | | | | LNA_GAIN_08 | | | | BBA_GAIN_08 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.35.3 Fields*

| Field | Function |
|---|---|
| 31-28 | LNA Gain 11 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LNA_GAIN_11 | LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_11 | BBA Gain 11<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_10 | LNA Gain 10<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_10 | BBA Gain 10<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_09 | LNA Gain 09<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_09 | BBA Gain 09<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_08 | LNA Gain 08<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_08 | BBA Gain 08<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## 64.3.3.3.1.36 AGC Gain Tables Step 15..12 (AGC_GAIN_TBL_15_12)

### *64.3.3.3.1.36.1 Offset*

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_15_12 | 8Ch |

### *64.3.3.3.1.36.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_15 | | | | BBA_GAIN_15 | | | | LNA_GAIN_14 | | | | BBA_GAIN_14 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_13 | | | | BBA_GAIN_13 | | | | LNA_GAIN_12 | | | | BBA_GAIN_12 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.36.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_15 | LNA Gain 15<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_15 | BBA Gain 15<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_14 | LNA Gain 14<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_14 | BBA Gain 14<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_13 | LNA Gain 13<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_13 | BBA Gain 13<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_12 | LNA Gain 12<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_12 | BBA Gain 12<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## 64.3.3.3.1.37 AGC Gain Tables Step 19..16 (AGC_GAIN_TBL_19_16)

#### 64.3.3.3.1.37.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_19_16 | 90h |

#### 64.3.3.3.1.37.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_19 | | | | BBA_GAIN_19 | | | | LNA_GAIN_18 | | | | BBA_GAIN_18 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_17 | | | | BBA_GAIN_17 | | | | LNA_GAIN_16 | | | | BBA_GAIN_16 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.3.3.1.37.3 Fields*

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_19 | LNA Gain 19<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_19 | BBA Gain 193<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_18 | LNA Gain 18<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_18 | BBA Gain 18<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_17 | LNA Gain 17<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_17 | BBA Gain 17<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_16 | LNA Gain 16<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_16 | BBA Gain 16<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## 64.3.3.3.1.38 AGC Gain Tables Step 23..20 (AGC_GAIN_TBL_23_20)

#### *64.3.3.3.1.38.1 Offset*

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_23_20 | 94h |

#### *64.3.3.3.1.38.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_23 | | | | BBA_GAIN_23 | | | | LNA_GAIN_22 | | | | BBA_GAIN_22 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_21 | | | | BBA_GAIN_21 | | | | LNA_GAIN_20 | | | | BBA_GAIN_20 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.38.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_23 | LNA Gain 23<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_23 | BBA Gain 23<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_22 | LNA Gain 22<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_22 | BBA Gain 22<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_21 | LNA Gain 21<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_21 | BBA Gain 21<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_20 | LNA Gain 20<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_20 | BBA Gain 20<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## 64.3.3.3.1.39 AGC Gain Tables Step 26..24 (AGC_GAIN_TBL_26_24)

#### 64.3.3.3.1.39.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_26_24 | 98h |

#### 64.3.3.3.1.39.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | LNA_GAIN_26 | | | | BBA_GAIN_26 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LNA_GAIN_25 | | | | BBA_GAIN_25 | | | | LNA_GAIN_24 | | | | BBA_GAIN_24 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.3.3.1.39.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-20<br>LNA_GAIN_26 | LNA Gain 26<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_26 | BBA Gain 26<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_25 | LNA Gain 25<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_25 | BBA Gain 25<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_24 | LNA Gain 24<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_24 | BBA Gain 24<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

### 64.3.3.3.1.40 DCOC Offset (DCOC_OFFSET_0 - DCOC_OFFSET_26)

#### *64.3.3.3.1.40.1 Offset*

For a = 0 to 26:

| Register | Offset |
|---|---|
| DCOC_OFFSET_a | A0h + (a × 4h) |

#### *64.3.3.3.1.40.2 Function*

DCOC Offset Registers. These registers can only be accessed when the radio oscillator clock is active.

#### 64.3.3.3.1.40.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCOC_TZA_OFFSET_Q | | | | | | | | DCOC_TZA_OFFSET_I | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCOC_BBA_OFFSET_Q | | | | | | 0 | | DCOC_BBA_OFFSET_I | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.40.4 Fields

| Field | Function |
|---|---|
| 31-24<br>DCOC_TZA_OFFSET_Q | DCOC TZA Q-channel offset<br>DCOC TZA Q-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 23-16<br>DCOC_TZA_OFFSET_I | DCOC TZA I-channel offset<br>DCOC TZA I-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 15-14<br>— | Reserved. |
| 13-8<br>DCOC_BBA_OFFSET_Q | DCOC BBA Q-channel offset<br>DCOC BBA Q-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 7-6<br>— | Reserved. |
| 5-0<br>DCOC_BBA_OFFSET_I | DCOC BBA I-channel offset<br>DCOC BBA I-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |

## 64.3.3.3.1.41 DCOC BBA DAC Step (DCOC_BBA_STEP)

#### 64.3.3.3.1.41.1 Offset

| Register | Offset |
|---|---|
| DCOC_BBA_STEP | 10Ch |

#### 64.3.3.3.1.41.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | BBA_DCOC_STEP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | BBA_DCOC_STEP_RECIP | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.41.3 Fields

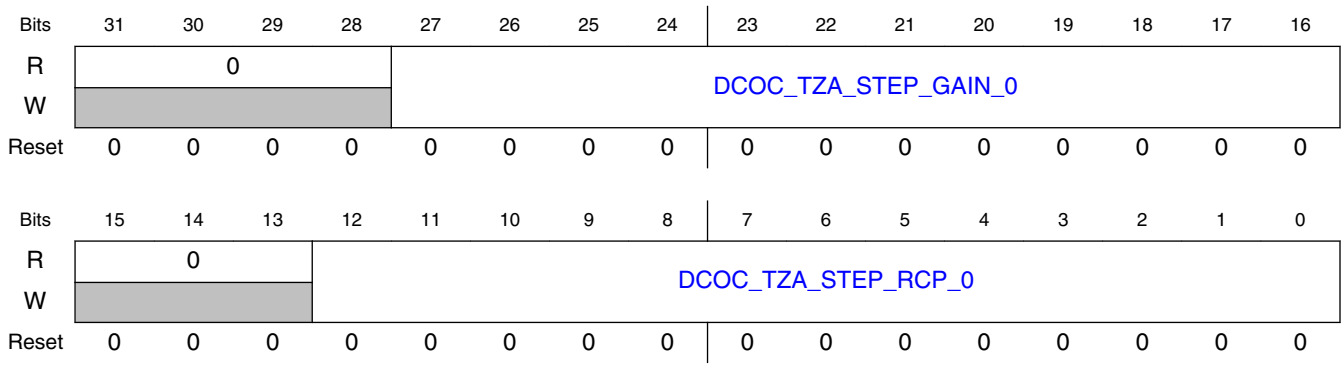
| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-16<br>BBA_DCOC_STEP | DCOC BBA Step Size<br>DCOC BBA Step Size (format: u6.3). This is the BBA DAC resolution in mV (1.2e3/32*63.44/120 = 19.83mV) times the ADC/decimator gain. This value is stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>BBA_DCOC_STEP_RECIP | DCOC BBA Reciprocal of Step Size<br>DCOC BBA Reciprocal of Step Size (format: u.[00]13). This the reciprocal of the BBA DCOC STEP value. This value is stored as a 15 bit fraction (though only 13 bits are programmed). |

## 64.3.3.3.1.42 DCOC TZA DAC Step 0 (DCOC_TZA_STEP_0)

#### 64.3.3.3.1.42.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_0 | 110h |

#### 64.3.3.3.1.42.2 Diagram




#### 64.3.3.3.1.42.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_0 | DCOC_TZA_STEP_GAIN_0<br>DCOC TZA Step Size 0 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 0. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_0 | DCOC_TZA_STEP_RCP_0<br>DCOC TZA Reciprocal of Step Size 0, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_0. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.43 DCOC TZA DAC Step 1 (DCOC_TZA_STEP_1)

#### 64.3.3.3.1.43.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_1 | 114h |

#### 64.3.3.3.1.43.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_TZA_STEP_GAIN_1 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_1 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.43.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_1 | DCOC_TZA_STEP_GAIN_1<br>DCOC TZA Step Size 1 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 1. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_1 | DCOC_TZA_STEP_RCP_1<br>DCOC TZA Reciprocal of Step Size 1, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_1. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.44 DCOC TZA DAC Step 2 (DCOC_TZA_STEP_2)

#### 64.3.3.3.1.44.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_2 | 118h |

#### 64.3.3.3.1.44.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_TZA_STEP_GAIN_2 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_2 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.44.3 Fields

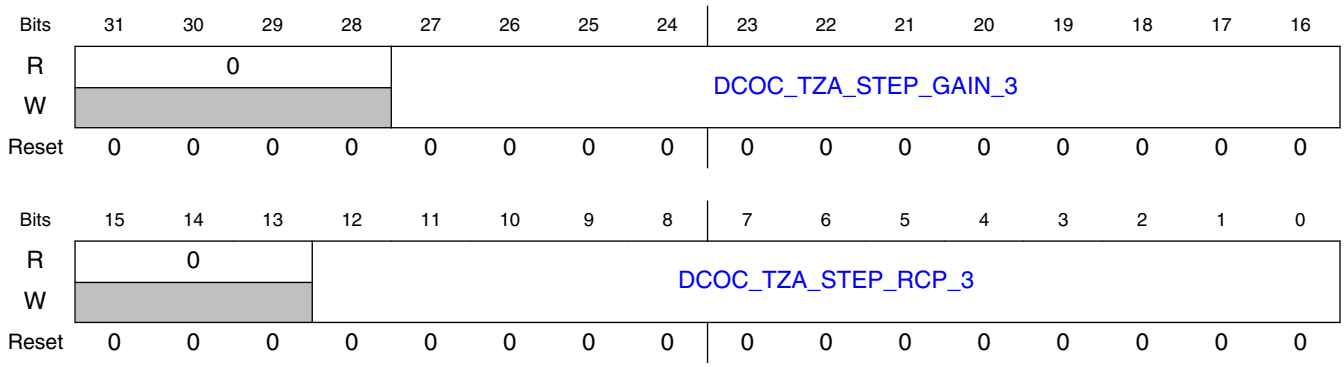
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_2 | DCOC_TZA_STEP_GAIN_2<br>DCOC TZA Step Size 2 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 2. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_2 | DCOC_TZA_STEP_RCP_2<br>DCOC TZA Reciprocal of Step Size 2, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_2. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.45 DCOC TZA DAC Step 3 (DCOC_TZA_STEP_3)

#### 64.3.3.3.1.45.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_3 | 11Ch |

#### 64.3.3.3.1.45.2 Diagram




#### 64.3.3.3.1.45.3 Fields

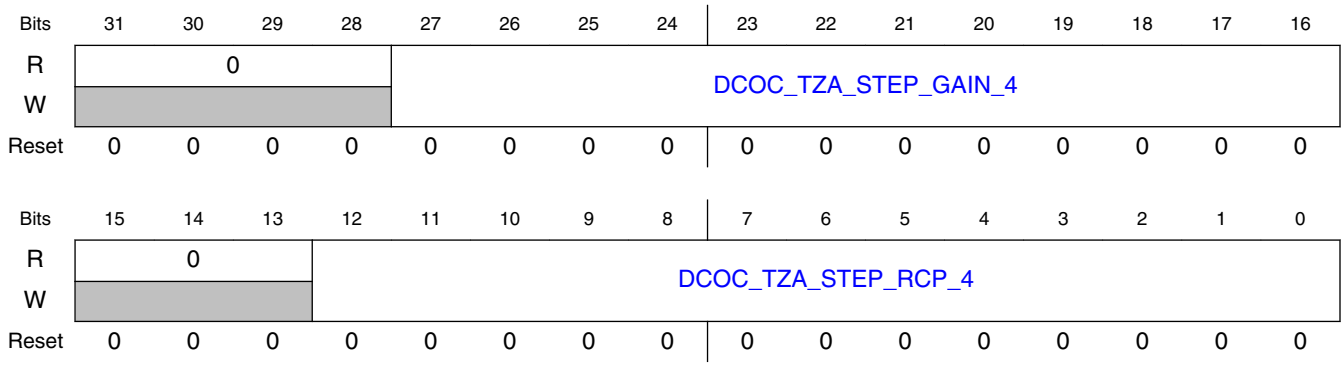
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_3 | DCOC_TZA_STEP_GAIN_3<br>DCOC TZA Step Size 3 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 3. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_3 | DCOC_TZA_STEP_RCP_3<br>DCOC TZA Reciprocal of Step Size 3, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_3. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.46 DCOC TZA DAC Step 4 (DCOC_TZA_STEP_4)

#### 64.3.3.3.1.46.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_4 | 120h |

### 64.3.3.3.1.46.2 Diagram




### 64.3.3.3.1.46.3 Fields

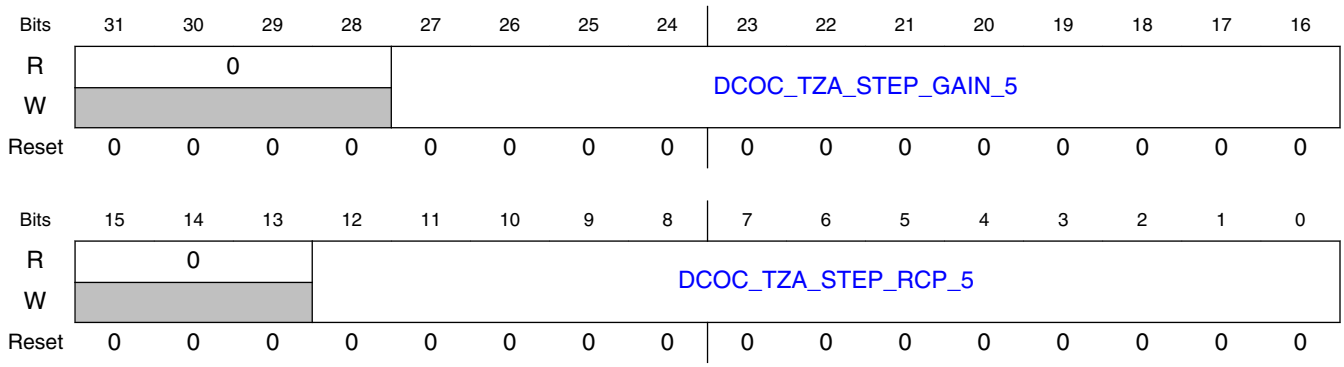
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_4 | DCOC_TZA_STEP_GAIN_4<br>DCOC TZA Step Size 4 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 4. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_4 | DCOC_TZA_STEP_RCP_4<br>DCOC TZA Reciprocal of Step Size 4, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_4. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.47 DCOC TZA DAC Step 5 (DCOC_TZA_STEP_5)

### 64.3.3.3.1.47.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_5 | 124h |

#### 64.3.3.3.1.47.2 Diagram




#### 64.3.3.3.1.47.3 Fields

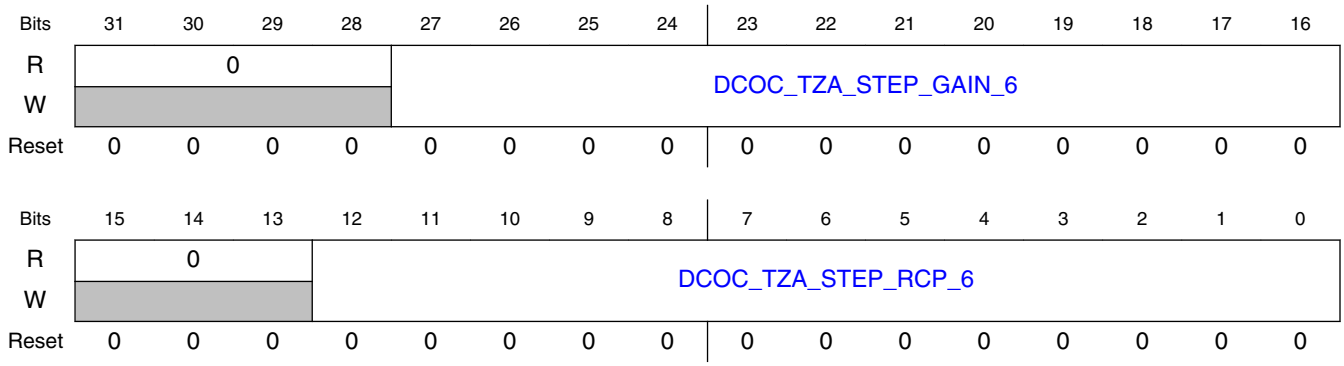
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_5 | DCOC_TZA_STEP_GAIN_5<br>DCOC TZA Step Size 5 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 5. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_5 | DCOC_TZA_STEP_RCP_5<br>DCOC TZA Reciprocal of Step Size 5, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_5. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.48 DCOC TZA DAC Step 6 (DCOC_TZA_STEP_6)

#### 64.3.3.3.1.48.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_6 | 128h |

#### 64.3.3.3.1.48.2 Diagram




#### 64.3.3.3.1.48.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_6 | DCOC_TZA_STEP_GAIN_6<br>DCOC TZA Step Size 6 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 6. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_6 | DCOC_TZA_STEP_RCP_6<br>DCOC TZA Reciprocal of Step Size 6, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_6. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.49 DCOC TZA DAC Step 7 (DCOC_TZA_STEP_7)

#### 64.3.3.3.1.49.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_7 | 12Ch |

#### 64.3.3.3.1.49.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_GAIN_7 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_7 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.49.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DCOC_TZA_STEP_GAIN_7 | DCOC_TZA_STEP_GAIN_7<br>DCOC TZA Step Size 7 with gain (format u10.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 7. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_7 | DCOC_TZA_STEP_RCP_7<br>DCOC TZA Reciprocal of Step Size 7, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_7. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.50 DCOC TZA DAC Step 5 (DCOC_TZA_STEP_8)

#### 64.3.3.3.1.50.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_8 | 130h |

#### 64.3.3.3.1.50.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_GAIN_8 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_8 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.50.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DCOC_TZA_STEP_GAIN_8 | DCOC_TZA_STEP_GAIN_8<br>DCOC TZA Step Size 8 with gain (format u10.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 8. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_8 | DCOC_TZA_STEP_RCP_8<br>DCOC TZA Reciprocal of Step Size 8, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_8. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.51 DCOC TZA DAC Step 9 (DCOC_TZA_STEP_9)

#### 64.3.3.3.1.51.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_9 | 134h |

#### 64.3.3.3.1.51.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCOC_TZA_STEP_GAIN_9 | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_9 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.51.3 Fields

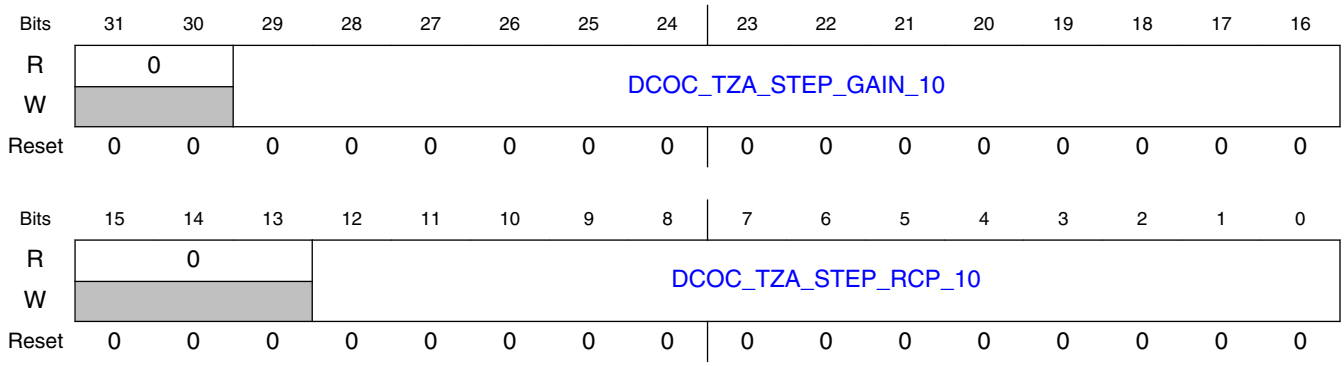
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-16<br>DCOC_TZA_STEP_GAIN_9 | DCOC_TZA_STEP_GAIN_9<br>DCOC TZA Step Size 9 with gain (format u11.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 9. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_9 | DCOC_TZA_STEP_RCP_9<br>DCOC TZA Reciprocal of Step Size 9, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_9. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.52 DCOC TZA DAC Step 10 (DCOC_TZA_STEP_10)

#### 64.3.3.3.1.52.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_10 | 138h |

#### 64.3.3.3.1.52.2 Diagram




#### 64.3.3.3.1.52.3 Fields

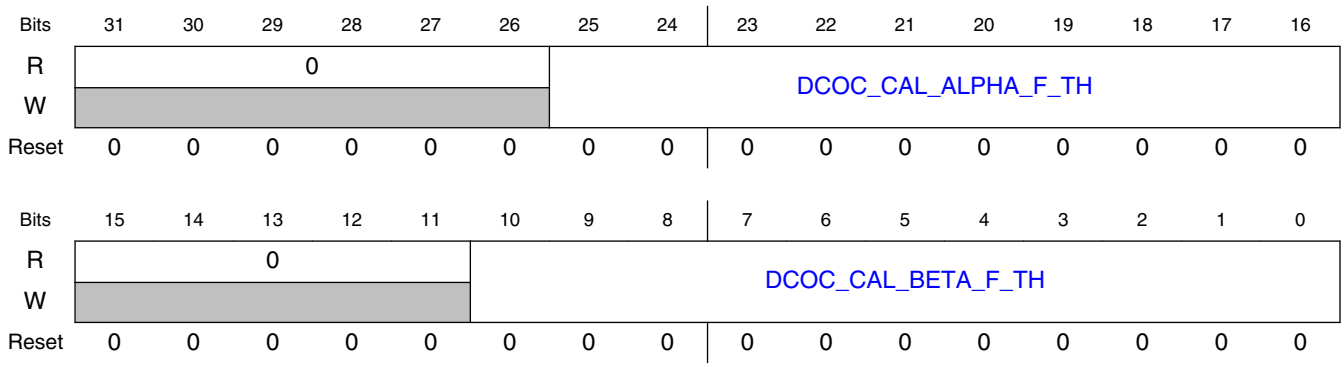
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-16<br>DCOC_TZA_STEP_GAIN_10 | DCOC_TZA_STEP_GAIN_10<br>DCOC TZA Step Size 10 with gain (format u11.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 10. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_10 | DCOC_TZA_STEP_RCP_10<br>DCOC TZA Reciprocal of Step Size 10, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_10. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## 64.3.3.3.1.53 DCOC Calibration Fail Thresholds (DCOC_CAL_FAIL_TH)

#### 64.3.3.3.1.53.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_FAIL_TH | 160h |

#### 64.3.3.3.1.53.2 Diagram




#### 64.3.3.3.1.53.3 Fields

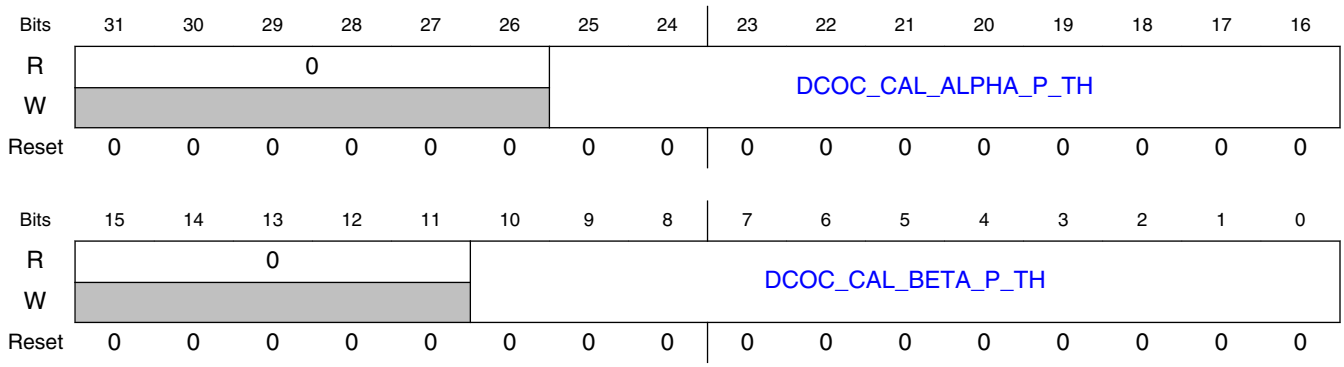
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>DCOC_CAL_ALPHA_F_TH | DCOC Calibration Alpha Fail Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the alpha-hat I and Q estimates are compared gainst this fail threshold. If any of the estimates are greater than the fail threshold, the calibration is considered a failure. Format is u4.6. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_BETA_F_TH | DCOC Calibration Beta Fail Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the beta-hat I and Q estimates are compared against this fail threshold. If any of the estimates are greater than the fail threshold, the calibration is considered a failure. Format is u11. |

## 64.3.3.3.1.54 DCOC Calibration Pass Thresholds (DCOC_CAL_PASS_TH)

#### 64.3.3.3.1.54.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_PASS_TH | 164h |

#### 64.3.3.3.1.54.2 Diagram




#### 64.3.3.3.1.54.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>DCOC_CAL_ALPHA_P_TH | DCOC Calibration Alpha Pass Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the alpha-hat I and Q estimates are compared against this pass threshold. If all of the estimates are less than the pass threshold, the calibration is considered as passing. Format is u4.6. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_BETA_P_TH | DCOC Calibration Beta Pass Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the beta-hat I and Q estimates are compared against this pass threshold. If all of the estimates are less than the pass threshold, the calibration is considered as passing. Format is u11. |

## 64.3.3.3.1.55 DCOC Calibration Alpha (DCOC_CAL_ALPHA)

#### 64.3.3.3.1.55.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_ALPHA | 168h |

#### 64.3.3.3.1.55.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_CAL_ALPHA_Q | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_CAL_ALPHA_I | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.55.3 Fields

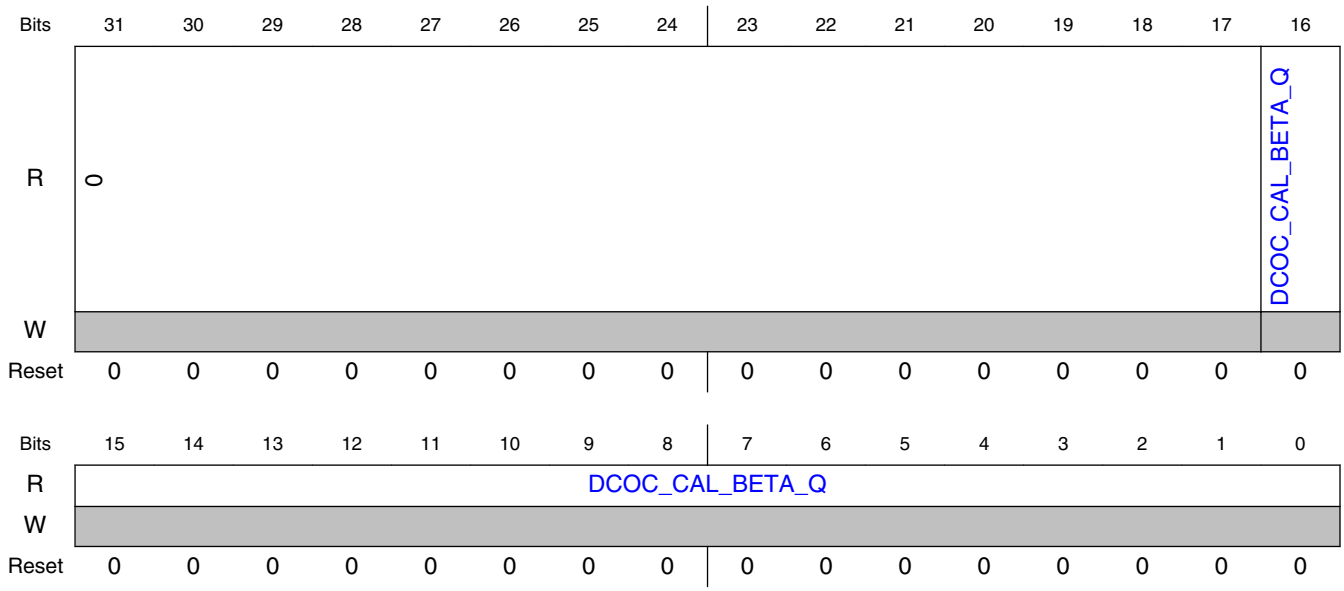
| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>DCOC_CAL_ALPHA_Q | DCOC_CAL_ALPHA_Q<br>DCOC Calibration Q-channel ALPHA. This read-only value represents the Q channel estimate of the ALPHA DC component calculated in DCOC calibration. Format s4.6. This is provided for debug/ characterization purposes. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_ALPHA_I | DCOC Calibration I-channel ALPHA constant<br>DCOC Calibration I-channel ALPHA. This read-only value represents the I channel estimate of the ALPHA DC component calculated in DCOC calibration. Format s4.6. This is provided for debug/ characterization purposes. |

### 64.3.3.3.1.56 DCOC Calibration Beta Q (DCOC_CAL_BETA_Q)

#### 64.3.3.3.1.56.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_BETA_Q | 16Ch |

#### 64.3.3.3.1.56.2 Diagram




#### 64.3.3.3.1.56.3 Fields

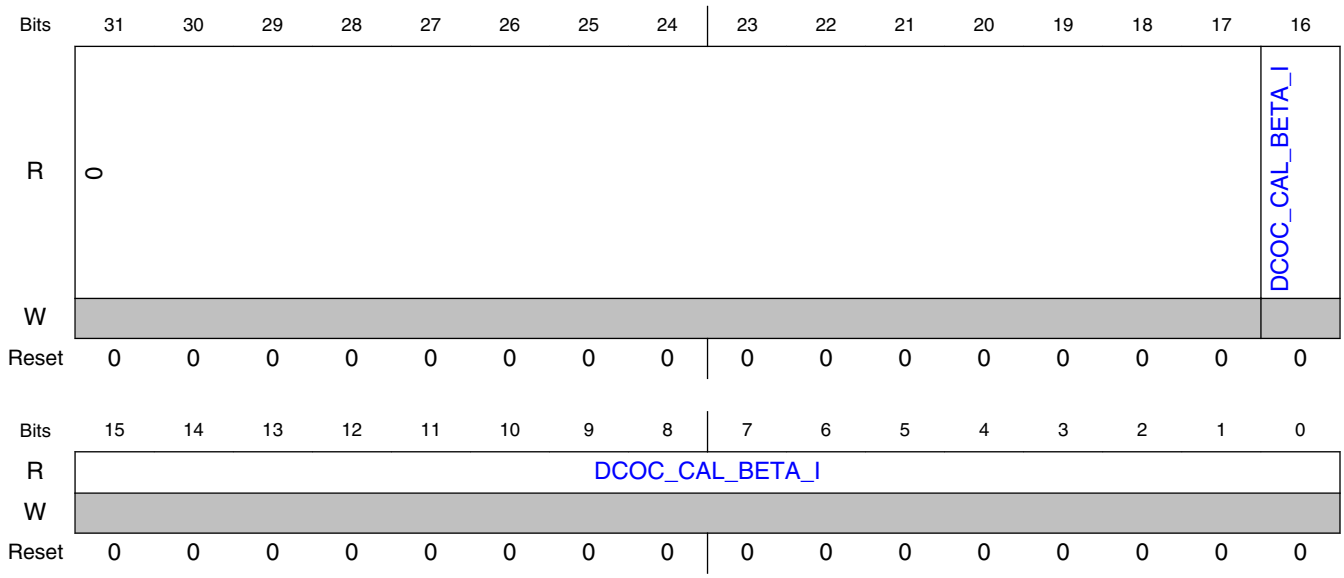
| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16-0<br>DCOC_CAL_BETA_Q | DCOC_CAL_BETA_Q<br>DCOC Calibration Q-channel BETA. This read-only value represents the Q channel estimate of the BETA DC component calculated in DCOC calibration. Format s11.5. This is provided for debug/characterization purposes. |

## 64.3.3.3.1.57 DCOC Calibration Beta I (DCOC_CAL_BETA_I)

#### 64.3.3.3.1.57.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_BETA_I | 170h |

#### 64.3.3.3.1.57.2 Diagram




#### 64.3.3.3.1.57.3 Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16-0<br>DCOC_CAL_BETA_I | DCOC_CAL_BETA_I<br>DCOC Calibration I-channel BETA. This read-only value represents the I channel estimate of the BETA DC component calculated in DCOC calibration. Format s11.5. This is provided for debug/characterization purposes. |

### 64.3.3.3.1.58 DCOC Calibration Gamma (DCOC_CAL_GAMMA)

#### 64.3.3.3.1.58.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_GAMMA | 174h |

#### 64.3.3.3.1.58.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCOC_CAL_GAMMA_Q | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCOC_CAL_GAMMA_I | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.58.3 Fields

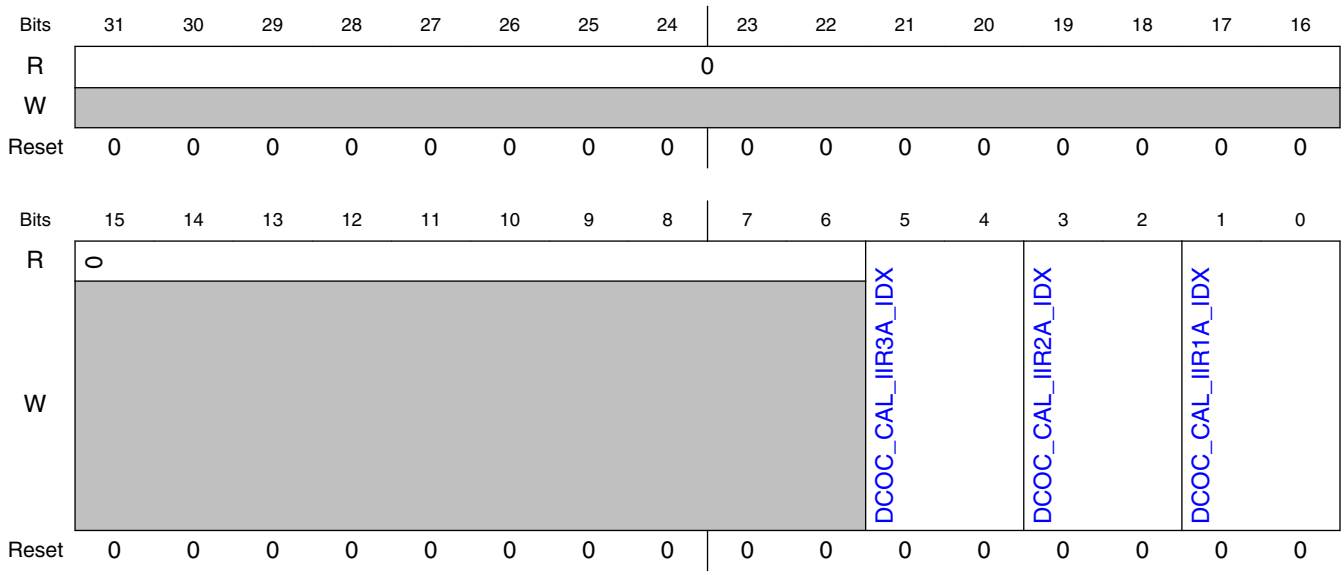
| Field | Function |
|---|---|
| 31-16<br>DCOC_CAL_GAMMA_Q | DCOC_CAL_GAMMA_Q<br>DCOC Calibration Q-channel GAMMA. This read-only value represents the Q channel estimate of the GAMMA DC component calculated in DCOC calibration. Format s11.4. This is provided for debug/ characterization purposes. |
| 15-0<br>DCOC_CAL_GAMMA_I | DCOC_CAL_GAMMA_I<br>DCOC Calibration I-channel GAMMA. This read-only value represents the I channel estimate of the GAMMA DC component calculated in DCOC calibration. Format s11.4. This is provided for debug/ characterization purposes. |

## 64.3.3.3.1.59 DCOC Calibration IIR (DCOC_CAL_IIR)

#### 64.3.3.3.1.59.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_IIR | 178h |

### 64.3.3.3.1.59.2 Diagram




### 64.3.3.3.1.59.3 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-4<br>DCOC_CAL_IIR 3A_IDX | DCOC Calibration IIR 3A Index<br>DCOC Calibration IIR 3A Index. Defines the filter coefficient use for the 3rd IIR filter in the DCOC calibration DC estimator.<br>00b - 1/4<br>01b - 1/8<br>10b - 1/16<br>11b - 1/32 |
| 3-2<br>DCOC_CAL_IIR 2A_IDX | DCOC Calibration IIR 2A Index<br>DCOC Calibration IIR 2A Index. Defines the filter coefficient use for the 2nd IIR filter in the DCOC calibration DC estimator.<br>00b - 1/1<br>01b - 1/4<br>10b - 1/8<br>11b - 1/16 |
| 1-0<br>DCOC_CAL_IIR 1A_IDX | DCOC Calibration IIR 1A Index<br>DCOC Calibration IIR 1A Index. Defines the filter coefficient use for the 1st IIR filter in the DCOC calibration DC estimator.<br>00b - 1/1<br>01b - 1/4<br>10b - 1/8<br>11b - 1/16 |

## 64.3.3.3.1.60 DCOC Calibration Result (DCOC_CAL1 - DCOC_CAL3)

### *64.3.3.3.1.60.1 Offset*

For a = 1 to 3:

| Register | Offset |
|---|---|
| DCOC_CALa | 17Ch + (a × 4h) |

### *64.3.3.3.1.60.2 Function*

Result of one of the calibration iterations.

### *64.3.3.3.1.60.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_CAL_RES_Q | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_CAL_RES_I | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.60.4 Fields*

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_CAL_RES_Q | DCOC Calibration Result - Q Channel<br>Q channel DCOC calibration result. This value represents the DCOC's Q channel DC estimate for the nth calibration gain setting. Format s11. This is provided for debug/chacterization purposes. |
| 15-12<br>— | Reserved. |
| 11-0<br>DCOC_CAL_RES_I | DCOC Calibration Result - I Channel<br>I channel DCOC calibration result. This value represents the DCOC's I channel DC estimate for the nth calibration gain setting. Format s11. This is provided for debug/chacterization purposes. |

## 64.3.3.3.1.61 RX_DIG CCA ED LQI Control Register 0 (CCA_ED_LQI_CTRL_0)

### *64.3.3.3.1.61.1 Offset*

| Register | Offset |
|---|---|
| CCA_ED_LQI_CTRL_0 | 190h |

### *64.3.3.3.1.61.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | SNR_ADJ | | | | | | LQI_CNTR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CORR_CNTR_THRESH | | | | | | | | LQI_CORR_THRESH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.61.3 Fields*

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-24<br>SNR_ADJ | SNR calculation adjustment<br>Signed value with quarter dB resolution. |
| 23-16<br>LQI_CNTR | LQI Counter<br>Duration of LQI in uS. |
| 15-8<br>CORR_CNTR_THRESH | Correlation Count Threshold<br>Threshold used to compare the counted correlation magnitudes exceeding LQI_CORR_THRESH. |
| 7-0<br>LQI_CORR_THRESH | LQI Correlation Threshold<br>Threshold used to compare correlation magnitude values from the PHY |

## 64.3.3.3.1.62 RX_DIG CCA ED LQI Control Register 1 (CCA_ED_LQI_CTRL_1)

### 64.3.3.3.1.62.1 Offset

| Register | Offset |
|---|---|
| CCA_ED_LQI_CTRL_1 | 194h |

### 64.3.3.3.1.62.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LQI_BIAS | | | | SNR_LQI_WEIGHT | | | | 0 | NB_WB_OVRD_EN | NB_WB_OVRD | MAN_MEAS_COMPLETE | CCA1_ED_EN_DIS | MEAS_TRANS_TO_IDLE | SEL_SNR_MODE | SNR_LQI_DIS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LQI_RSSI_SENS | | | | LQI_RSSI_WEIGHT | | | RSSI_NOISE_AVG_FACTOR | | | RSSI_NOISE_AVG_DELAY | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.3.3.1.62.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LQI_BIAS | LQI Bias.<br>Bias used for LQI calculation. Applied bias = -36 + 2*LQI_BIAS. |
| 27-24<br>SNR_LQI_WEIGHT | SNR LQI Weight<br>SNR weight used for LQI calculation<br>0000b - 0.0<br>0001b - 1.0<br>0010b - 1.125<br>0011b - 1.25<br>0100b - 1.375<br>0101b - 1.5 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0110b - 1.625<br>0111b - 1.75<br>1000b - 1.875<br>1001b - 2.0<br>1010b - 2.125<br>1011b - 2.25<br>1100b - 2.375<br>1101b - 2.5<br>1110b - 2.625<br>1111b - 2.75 |
| 23<br>—— | Reserved. |
| 22<br>NB_WB_OVRD_EN | Narrowband Wideband Override Enable<br>Override enable associated with NB_WB_OVRD. Intended to be used only for debug.<br>0b - Normal operation<br>1b - RSSI narrowband/wideband selection is via NB_WB_OVRD |
| 21<br>NB_WB_OVRD | Narrowband Wideband Override<br>Controls the RSSI's wideband/narrowband selection when NB_WB_OVRD_EN is set. Intended to be used only for debug.<br>0b - RSSI forced to be in Wideband mode if NB_WB_OVRD_EN is set<br>1b - RSSI forced to be in Narrowband mode if NB_WB_OVRD_EN is set |
| 20<br>MAN_MEAS_COMPLETE | Manual measurement complete<br>0b - Normal operation<br>1b - Manually asserts the measurement complete signal for the RX_DIG CCA/ED/LQI blocks. Intended to be used only for debug. |
| 19<br>CCA1_ED_EN_DIS | CCA1_ED_EN Disable<br>0b - Normal operation<br>1b - CCA1_ED_EN input is disabled |
| 18<br>MEAS_TRANS_TO_IDLE | Measurement Transition to IDLE<br>Establishes the state machine transition following an LQI or CCA1/ED measurement<br>0b - Module transitions to RSSI state<br>1b - Module transitions to IDLE state |
| 17<br>SEL_SNR_MODE | Select SNR Mode<br>Selects the SNR input used in the LQI calculator block.<br>0b - SNR estimate<br>1b - Mapped correlation magnitude |
| 16<br>SNR_LQI_DIS | SNR LQI Disable<br>0b - Normal operation.<br>1b - The RX_DIG CCA/ED/LQI block ignores the AA match input which starts an LQI measurement. |
| 15-12<br>LQI_RSSI_SENS | LQI RSSI Sensitivity<br>Unsigned integer used for calculation of LQI sensitivity. Sensitivity = -103 + LQI_RSSI_SENS. |
| 11-9<br>LQI_RSSI_WEIGHT | LQI RSSI Weight<br>RSSI weight used for LQI calculation<br>000b - 2.0<br>001b - 2.125<br>010b - 2.25<br>011b - 2.375 |

*Table continues on the next page...*

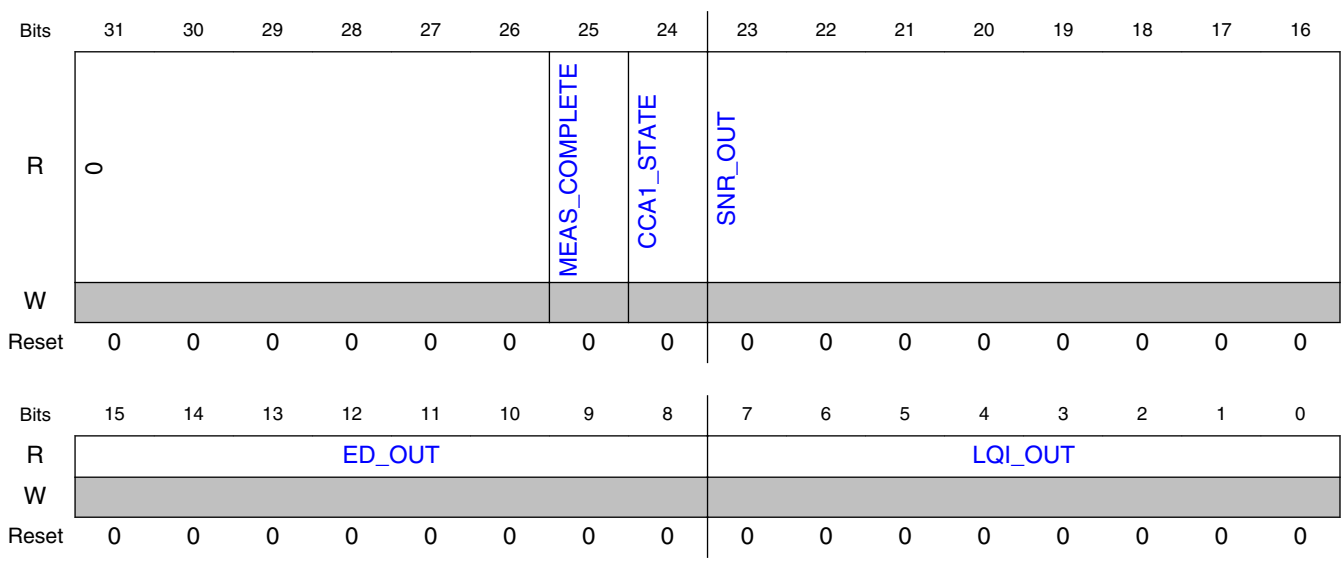
| Field | Function |
|---|---|
| | 100b - 2.5<br>101b - 2.625<br>110b - 2.75<br>111b - 2.875 |
| 8-6<br>RSSI_NOISE_AVG_FACTOR | RSSI Noise Averaging Factor<br>Factor used for RSSI and SNR averaging.<br>000b - 1<br>001b - 64<br>010b - 70<br>011b - 128<br>100b - 139<br>101b - 256<br>110b - 277<br>111b - 512 |
| 5-0<br>RSSI_NOISE_AVG_DELAY | RSSI Noise Averaging Delay<br>Programmable delay used to delay the enable of averagers "RSSI Averager" and "Noise Averager" after a cca1_ed_trig assertion (in wideband mode) or after a aa_sfd_matched assertion (in narrowband mode). The delay is expressed in ticks of frequenc Fs/N, where Fs is the ADC decimator output sampling frequency in wideband mode or the base-band sampling frequency in the narrowband mode. |

## 64.3.3.3.1.63 RX_DIG CCA ED LQI Status Register 0 (CCA_ED_LQI_STAT_0)

### *64.3.3.3.1.63.1 Offset*

| Register | Offset |
|---|---|
| CCA_ED_LQI_STAT_0 | 198h |

### *64.3.3.3.1.63.2 Diagram*



#### *64.3.3.3.1.63.3 Fields*

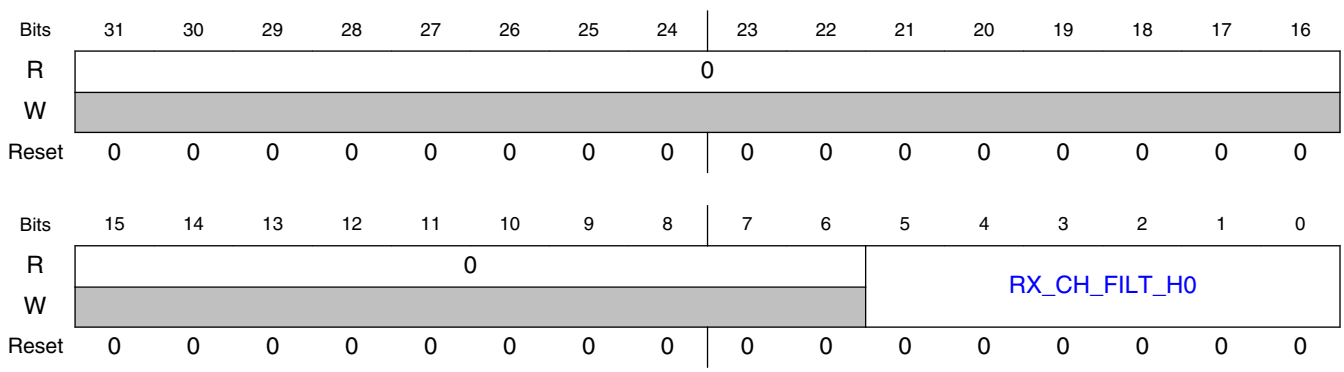
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>MEAS_COMPLETE | Measurement Complete<br>Set upon measurement complete |
| 24<br>CCA1_STATE | CCA1 State<br>Reflects the state of CCA1 channel state |
| 23-16<br>SNR_OUT | SNR output<br>Reflects the SNR measurement value. |
| 15-8<br>ED_OUT | ED output<br>Reflects the Energy Detect (DC) measurement value. |
| 7-0<br>LQI_OUT | LQI output<br>Reflects the LQI measurement value. |

### 64.3.3.3.1.64 Receive Channel Filter Coefficient 0 (RX_CHF_COEF_0)

#### *64.3.3.3.1.64.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_0 | 1A0h |

#### *64.3.3.3.1.64.2 Diagram*



#### 64.3.3.3.1.64.3 Fields

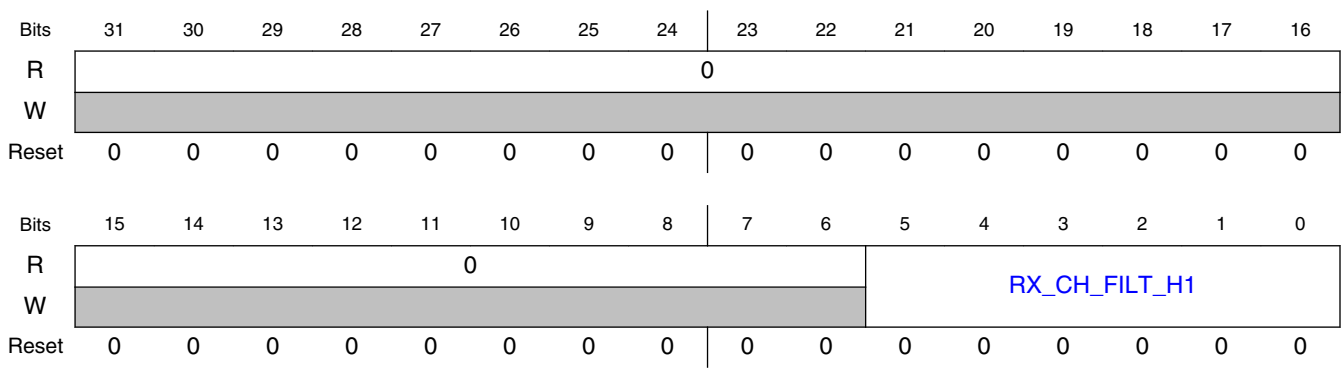
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>RX_CH_FILT_H0 | RX Channel Filter Coefficient 0<br>RX Channel Filter Coefficient 0, 6-bit signed fractional |

### 64.3.3.3.1.65 Receive Channel Filter Coefficient 1 (RX_CHF_COEF_1)

#### 64.3.3.3.1.65.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_1 | 1A4h |

#### 64.3.3.3.1.65.2 Diagram




#### 64.3.3.3.1.65.3 Fields

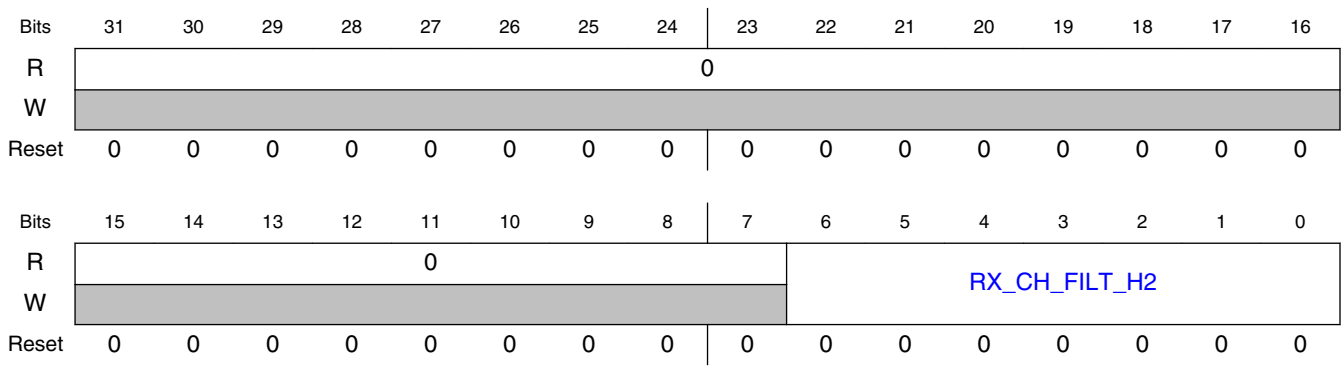
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>RX_CH_FILT_H1 | RX Channel Filter Coefficient 1<br>RX Channel Filter Coefficient 1, 6-bit signed fractional |

## 64.3.3.3.1.66 Receive Channel Filter Coefficient 2 (RX_CHF_COEF_2)

### *64.3.3.3.1.66.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_2 | 1A8h |

### *64.3.3.3.1.66.2 Diagram*




### *64.3.3.3.1.66.3 Fields*

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H2 | RX Channel Filter Coefficient 2<br>RX Channel Filter Coefficient 2, 7-bit signed fractional |

## 64.3.3.3.1.67 Receive Channel Filter Coefficient 3 (RX_CHF_COEF_3)

### *64.3.3.3.1.67.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_3 | 1ACh |

#### 64.3.3.3.1.67.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RX_CH_FILT_H3 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.67.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H3 | RX Channel Filter Coefficient 3<br>RX Channel Filter Coefficient 3, 7-bit signed fractional |

### 64.3.3.3.1.68 Receive Channel Filter Coefficient 4 (RX_CHF_COEF_4)

#### 64.3.3.3.1.68.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_4 | 1B0h |

#### 64.3.3.3.1.68.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RX_CH_FILT_H4 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.3.3.1.68.3 Fields*

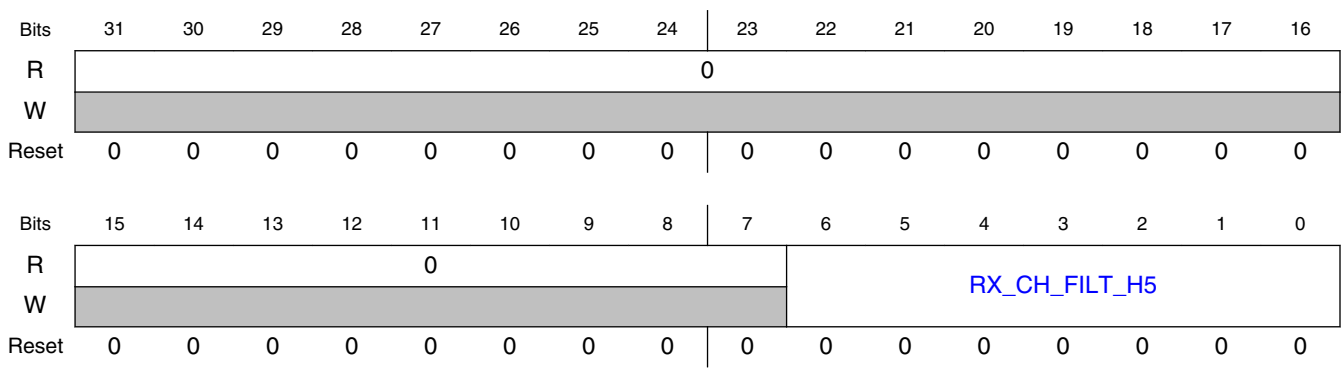
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H4 | RX Channel Filter Coefficient 4<br>RX Channel Filter Coefficient 4, 7-bit signed fractional |

### 64.3.3.3.1.69 Receive Channel Filter Coefficient 5 (RX_CHF_COEF_5)

#### *64.3.3.3.1.69.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_5 | 1B4h |

#### *64.3.3.3.1.69.2 Diagram*




#### *64.3.3.3.1.69.3 Fields*

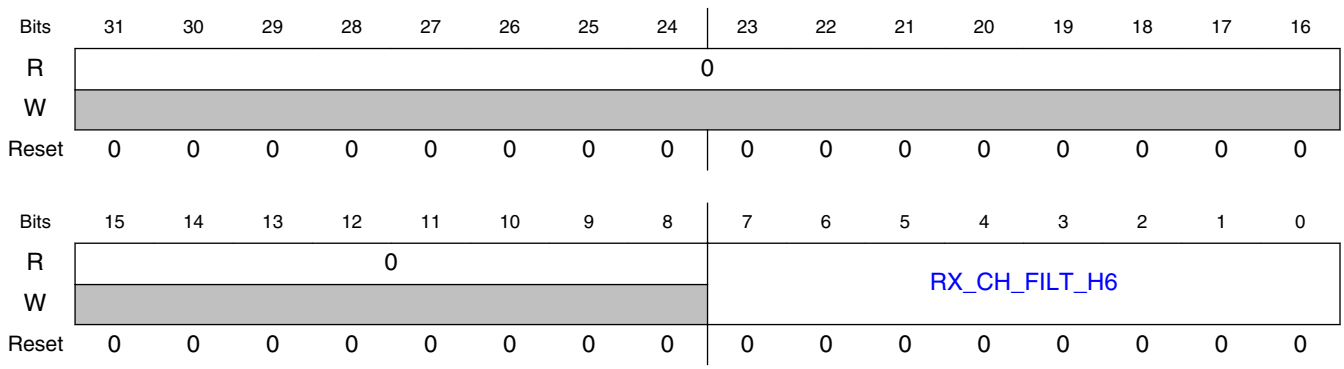
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H5 | RX Channel Filter Coefficient 5<br>RX Channel Filter Coefficient 5, 7-bit signed fractional |

## 64.3.3.3.1.70 Receive Channel Filter Coefficient 6 (RX_CHF_COEF_6)

### *64.3.3.3.1.70.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_6 | 1B8h |

### *64.3.3.3.1.70.2 Diagram*




### *64.3.3.3.1.70.3 Fields*

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>RX_CH_FILT_H6 | RX Channel Filter Coefficient 6<br>RX Channel Filter Coefficient 6, 8-bit signed fractional |

## 64.3.3.3.1.71 Receive Channel Filter Coefficient 7 (RX_CHF_COEF_7)

### *64.3.3.3.1.71.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_7 | 1BCh |

#### 64.3.3.3.1.71.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RX_CH_FILT_H7 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.71.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>RX_CH_FILT_H7 | RX Channel Filter Coefficient 7<br>RX Channel Filter Coefficient 7, 8-bit signed fractional |

### 64.3.3.3.1.72 Receive Channel Filter Coefficient 8 (RX_CHF_COEF_8)

#### 64.3.3.3.1.72.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_8 | 1C0h |

#### 64.3.3.3.1.72.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RX_CH_FILT_H8 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.3.3.1.72.3 Fields*

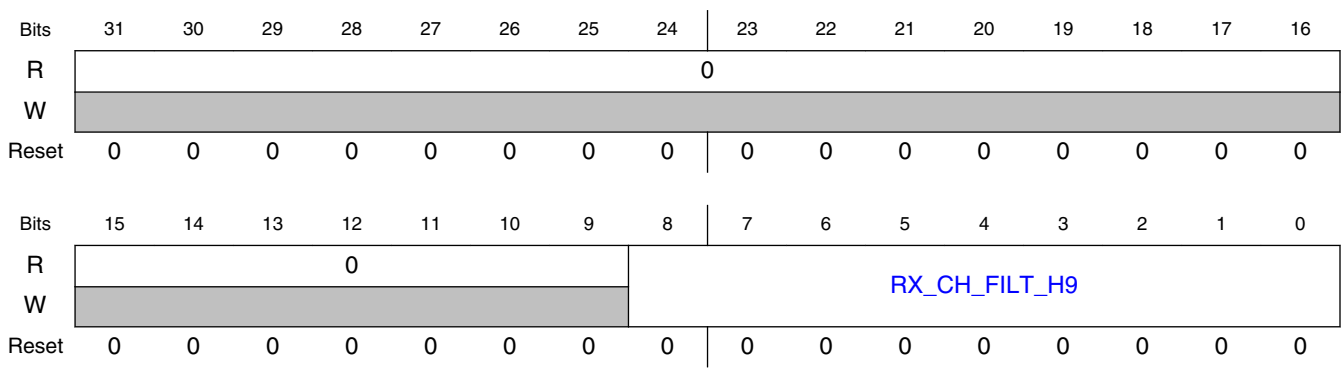
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>RX_CH_FILT_H8 | RX Channel Filter Coefficient 8<br>RX Channel Filter Coefficient 8, 9-bit signed fractional |

### 64.3.3.3.1.73 Receive Channel Filter Coefficient 9 (RX_CHF_COEF_9)

#### *64.3.3.3.1.73.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_9 | 1C4h |

#### *64.3.3.3.1.73.2 Diagram*




#### *64.3.3.3.1.73.3 Fields*

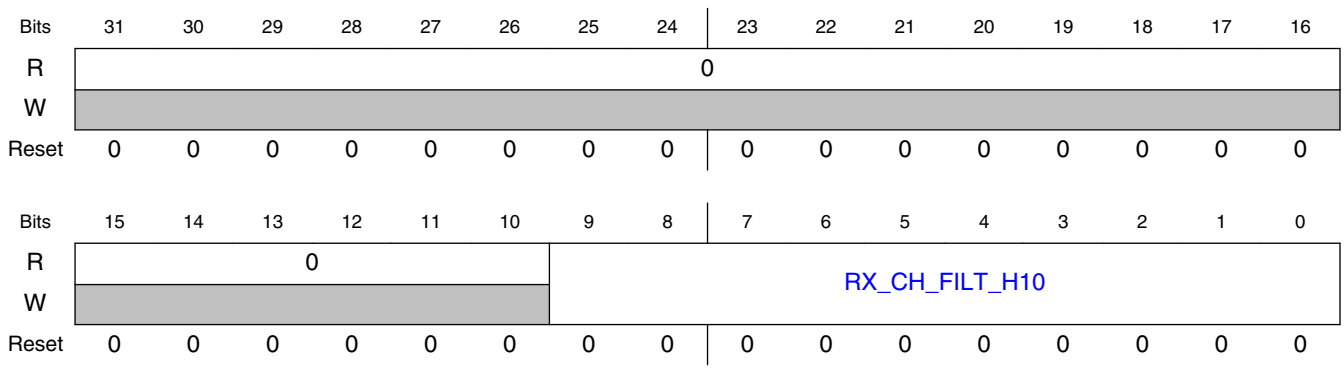
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>RX_CH_FILT_H9 | RX Channel Filter Coefficient 9<br>RX Channel Filter Coefficient 9, 9-bit signed fractional |

## 64.3.3.3.1.74 Receive Channel Filter Coefficient 10 (RX_CHF_COEF_10)

### *64.3.3.3.1.74.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_10 | 1C8h |

### *64.3.3.3.1.74.2 Diagram*




### *64.3.3.3.1.74.3 Fields*

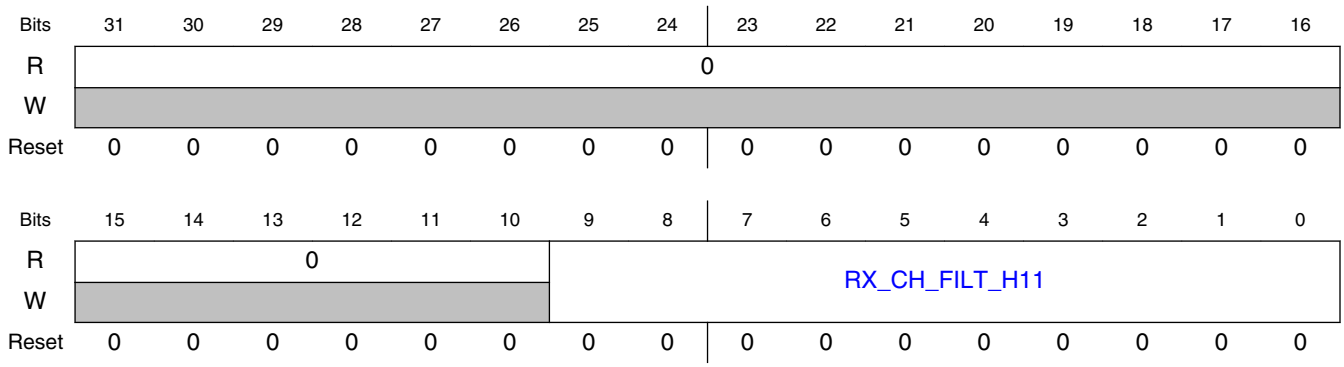
| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RX_CH_FILT_H10 | RX Channel Filter Coefficient 10<br>RX Channel Filter Coefficient 10, 10-bit signed fractional |

## 64.3.3.3.1.75 Receive Channel Filter Coefficient 11 (RX_CHF_COEF_11)

### *64.3.3.3.1.75.1 Offset*

| Register | Offset |
|---|---|
| RX_CHF_COEF_11 | 1CCh |

#### 64.3.3.3.1.75.2 Diagram




#### 64.3.3.3.1.75.3 Fields

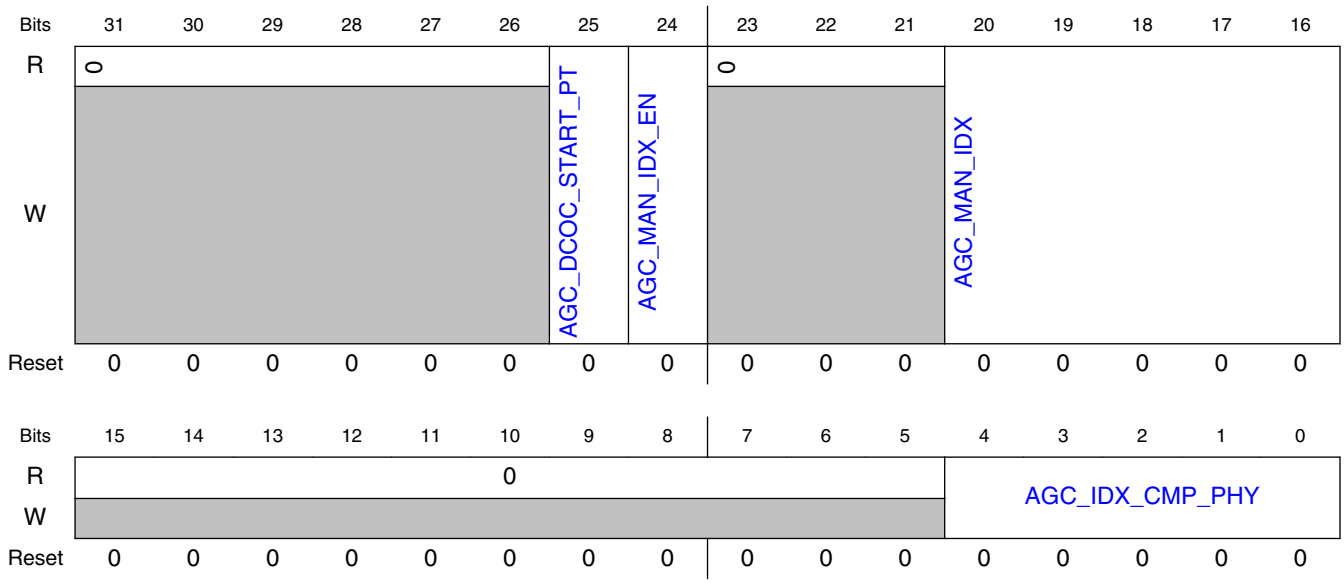
| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RX_CH_FILT_H11 | RX Channel Filter Coefficient 11<br>RX Channel Filter Coefficient 11, 10-bit signed fractional |

## 64.3.3.3.1.76 AGC Manual AGC Index (AGC_MAN_AGC_IDX)

#### 64.3.3.3.1.76.1 Offset

| Register | Offset |
|---|---|
| AGC_MAN_AGC_IDX | 1D0h |

#### 64.3.3.3.1.76.2 Diagram




#### 64.3.3.3.1.76.3 Fields

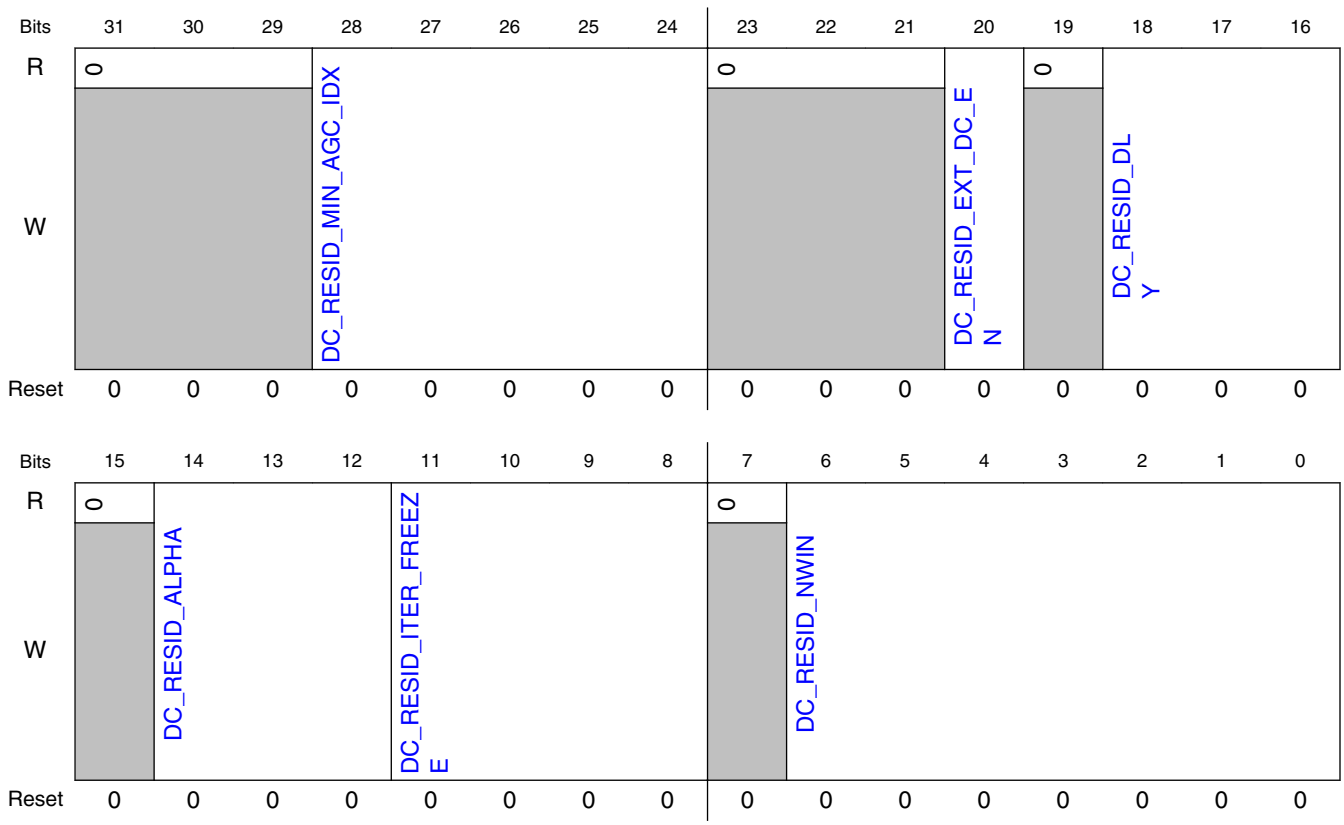
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>AGC_DCOC_START_PT | AGC DCOC Start Point<br>When set, the starting value for the AGC gain table index after DCOC calibration will be AGC_MAN_IDX instead of index 26. |
| 24<br>AGC_MAN_IDX_EN | AGC Manual Index Enable<br>When set, the AGC gain table index is overridden using AGC_MAN_IDX. |
| 23-21<br>— | Reserved. |
| 20-16<br>AGC_MAN_IDX | AGC Manual Index<br>AGC gain table index override value (when AGC_MAN_IDX_EN is set), or AGC gain table index starting point after DCOC calibration (when AGC_DCOC_START_PT is set). |
| 15-5<br>— | Reserved. |
| 4-0<br>AGC_IDX_CMP_PHY | AGC Index Compare threshold for PHY<br>The AGC's agc_thresh_below output (which is connected to the PHY's pd_mode_switch input) is asserted when the current AGC index is less than this threshold. |

## 64.3.3.3.1.77 DC Residual Control (DC_RESID_CTRL)

### 64.3.3.3.1.77.1 Offset

| Register | Offset |
|---|---|
| DC_RESID_CTRL | 1D4h |

### 64.3.3.3.1.77.2 Diagram




### 64.3.3.3.1.77.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-24<br>DC_RESID_MIN_AGC_IDX | DC Residual Minimum AGC Table Index<br>Specifies the minimum AGC table index value at which DC residual can be enabled. E.g., if this is 5'd0, then the DC residual is enabled for all AGC gain table indexes (assuming RX_DC_RESID_EN is set) if this is 5'd20, then tracking is enabled for AGC gain table indexes 20-26 (assuming RX_DC_RESID_EN is set). |
| 23-21<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 20<br>DC_RESID_EXT_DC_EN | DC Residual External DC Enable<br>0b - External DC disable. The DC Residual activates at a delay specified by DC_RESID_DLY after an AGC gain change pulse. The DC Residual is initialized with a DC offset of 0.<br>1b - External DC enable. The DC residual activates after the DCOC's tracking hold timer expires. The DC Residual is initialized with the DC estimate from the DCOC tracking estimator. |
| 19<br>— | Reserved. |
| 18-16<br>DC_RESID_DLY | DC Residual Delay<br>The delay from activation of the DC residual block to the time when it starts processing. For a 32MHz reference clock, the delay in microseconds matches the value programmed. For other clock frequencies, the delay is scaled based on the clock period. Supported values: 0-7. This delay is only used when EXT_DC_EN=0. |
| 15<br>— | Reserved. |
| 14-12<br>DC_RESID_ALPHA | DC Residual Alpha<br>The Alpha parameter controls the rate at which the DC estimate is updated. The update factor is 2^(-Alpha). |
| 11-8<br>DC_RESID_ITER_FREEZE | DC Residual Iteration Freeze<br>Number of windows of DC_RESID_NWIN samples before the DC is frozen. Supported values: 1-8 |
| 7<br>— | Reserved. |
| 6-0<br>DC_RESID_NWIN | DC Residual NWIN<br>Number of samples in a window. |

## 64.3.3.3.1.78 DC Residual Estimate (DC_RESID_EST)

### *64.3.3.3.1.78.1 Offset*

| Register | Offset |
|---|---|
| DC_RESID_EST | 1D8h |

#### 64.3.3.3.1.78.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DC_RESID_OFFSET_Q | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DC_RESID_OFFSET_I | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.78.3 Fields

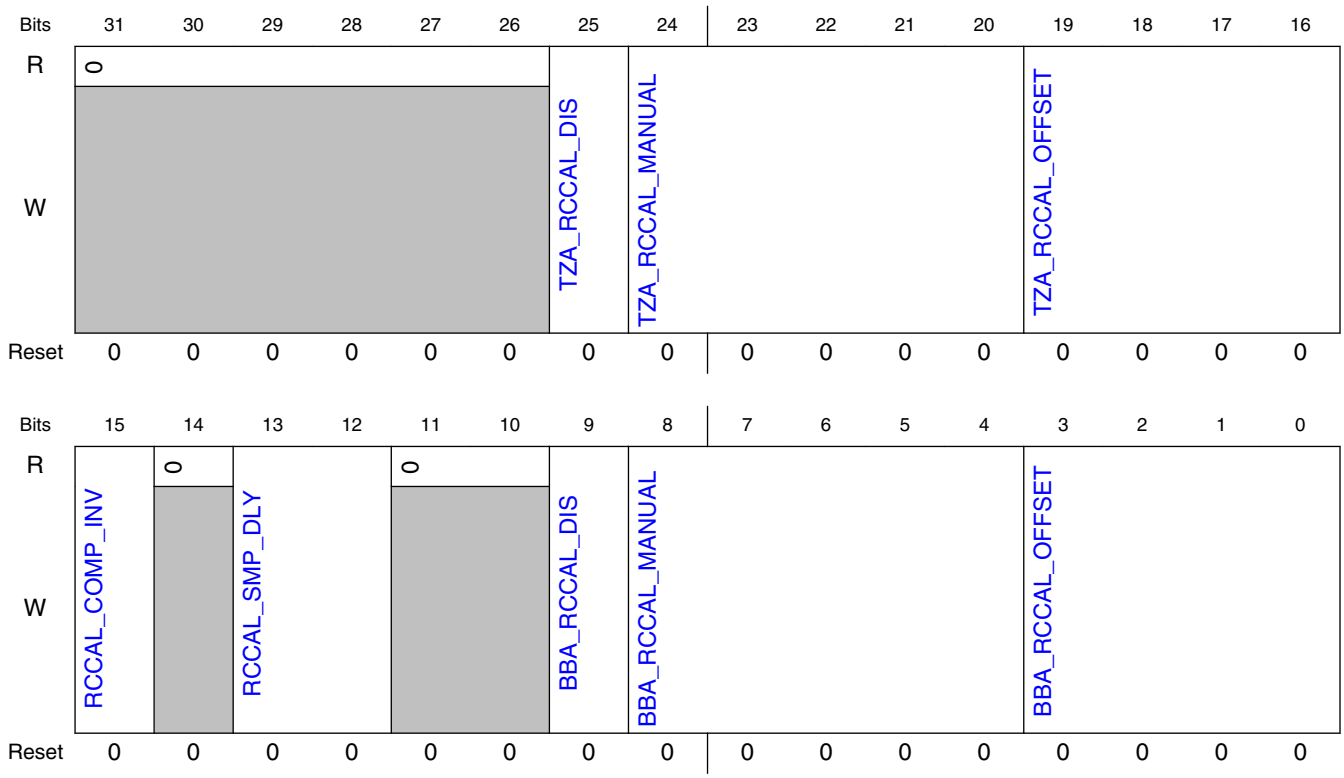
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DC_RESID_OFFSET_Q | DC Residual Offset Q<br>Reflects the current DC residual offset estimate for Q channel. Format is s11.1. This is provided for debug and characterization purposes only. |
| 15-13<br>— | Reserved. |
| 12-0<br>DC_RESID_OFFSET_I | DC Residual Offset I<br>Reflects the current DC residual offset estimate for I channel. Format is s11.1. This is provided for debug and characterization purposes only. |

## 64.3.3.3.1.79 RX RC Calibration Control0 (RX_RCCAL_CTRL0)

#### 64.3.3.3.1.79.1 Offset

| Register | Offset |
|---|---|
| RX_RCCAL_CTRL0 | 1DCh |

### 64.3.3.3.1.79.2 Diagram




### 64.3.3.3.1.79.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>TZA_RCCAL_DIS | TZA RC Calibration Disable<br>0b - TZA RC Calibration is enabled<br>1b - TZA RC Calibration is disabled |
| 24-20<br>TZA_RCCAL_MANUAL | TZA RC Calibration manual value<br>If TZA_RCCAL_DIS bit is set, this value is used for the TZA calibration. |
| 19-16<br>TZA_RCCAL_OFFSET | TZA RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the TZA. Format is s3. |
| 15<br>RCCAL_COMP_INV | RC Calibration comp_out Invert<br>0b - The comp_out signal polarity is NOT inverted<br>1b - The comp_out signal polarity is inverted |
| 14<br>— | Reserved. |
| 13-12 | RC Calibration Sample Delay |

*Table continues on the next page...*

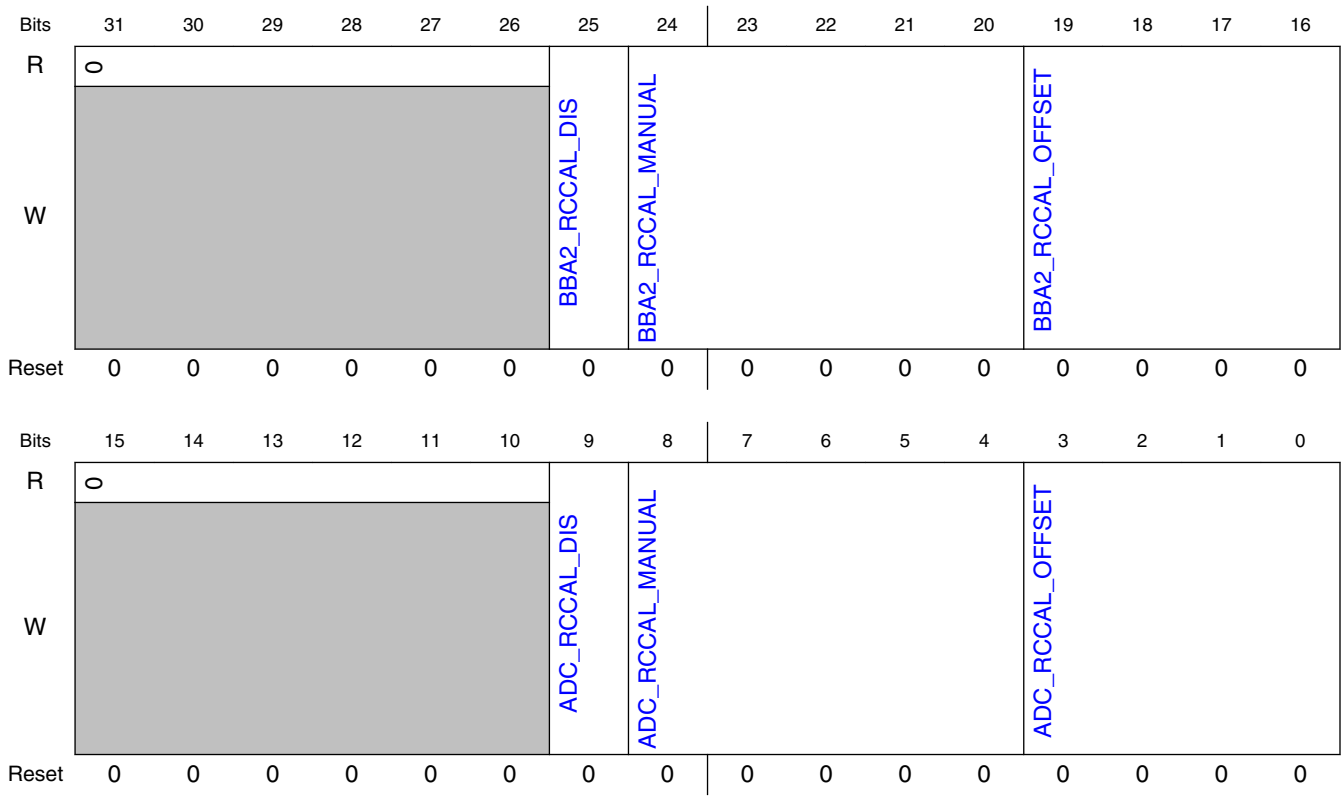
| Field | Function |
|---|---|
| RCCAL_SMP_DLY | 00b - The comp_out signal is sampled 0 clk cycle after sample signal is deasserted<br>01b - The comp_out signal is sampled 1 clk cycle after sample signal is deasserted<br>10b - The comp_out signal is sampled 2 clk cycle after sample signal is deasserted<br>11b - The comp_out signal is sampled 3 clk cycle after sample signal is deasserted |
| 11-10<br>— | Reserved. |
| 9<br>BBA_RCCAL_DIS | BBA RC Calibration Disable<br>0b - BBA RC Calibration is enabled<br>1b - BBA RC Calibration is disabled |
| 8-4<br>BBA_RCCAL_MANUAL | BBA RC Calibration manual value<br>If BBA_RCCAL_DIS bit is set, this value is used for the BBA calibration. |
| 3-0<br>BBA_RCCAL_OFFSET | BBA RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the BBA. Format is s3. |

## 64.3.3.3.1.80 RX RC Calibration Control1 (RX_RCCAL_CTRL1)

### *64.3.3.3.1.80.1 Offset*

| Register | Offset |
|---|---|
| RX_RCCAL_CTRL1 | 1E0h |

### 64.3.3.3.1.80.2 Diagram




### 64.3.3.3.1.80.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>BBA2_RCCAL_DIS | BBA2 RC Calibration Disable<br>0b - BBA2 RC Calibration is enabled<br>1b - BBA2 RC Calibration is disabled |
| 24-20<br>BBA2_RCCAL_MANUAL | BBA2 RC Calibration manual value<br>If BBA2_RCCAL_DIS bit is set, this value is used for the BBA2 calibration. |
| 19-16<br>BBA2_RCCAL_OFFSET | BBA2 RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the BBA2. Format is s3. |
| 15-10<br>— | Reserved. |
| 9<br>ADC_RCCAL_DIS | ADC RC Calibration Disable<br>0b - ADC RC Calibration is enabled<br>1b - ADC RC Calibration is disabled |
| 8-4 | ADC RC Calibration manual value |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ADC_RCCAL_MANUAL | If ADC_RCCAL_DIS bit is set, this value is used for the ADC calibration. |
| 3-0<br>ADC_RCCAL_OFFSET | ADC RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the ADC. Format is s3. |

## 64.3.3.3.1.81 RX RC Calibration Status (RX_RCCAL_STAT)

### *64.3.3.3.1.81.1 Offset*

| Register | Offset |
|---|---|
| RX_RCCAL_STAT | 1E4h |

### *64.3.3.3.1.81.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | TZA_RCCAL | | | | | BBA_RCCAL | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BBA2_RCCAL | | | | | ADC_RCCAL | | | | | RCCAL_CODE | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### *64.3.3.3.1.81.3 Fields*

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-21<br>TZA_RCCAL | TZA RC Calibration<br>The RC Calibration value used for TZA |
| 20-16<br>BBA_RCCAL | BBA RC Calibration<br>The RC Calibration value used for BBA |
| 15<br>— | Reserved. |
| 14-10 | BBA2 RC Calibration |

*Table continues on the next page...*

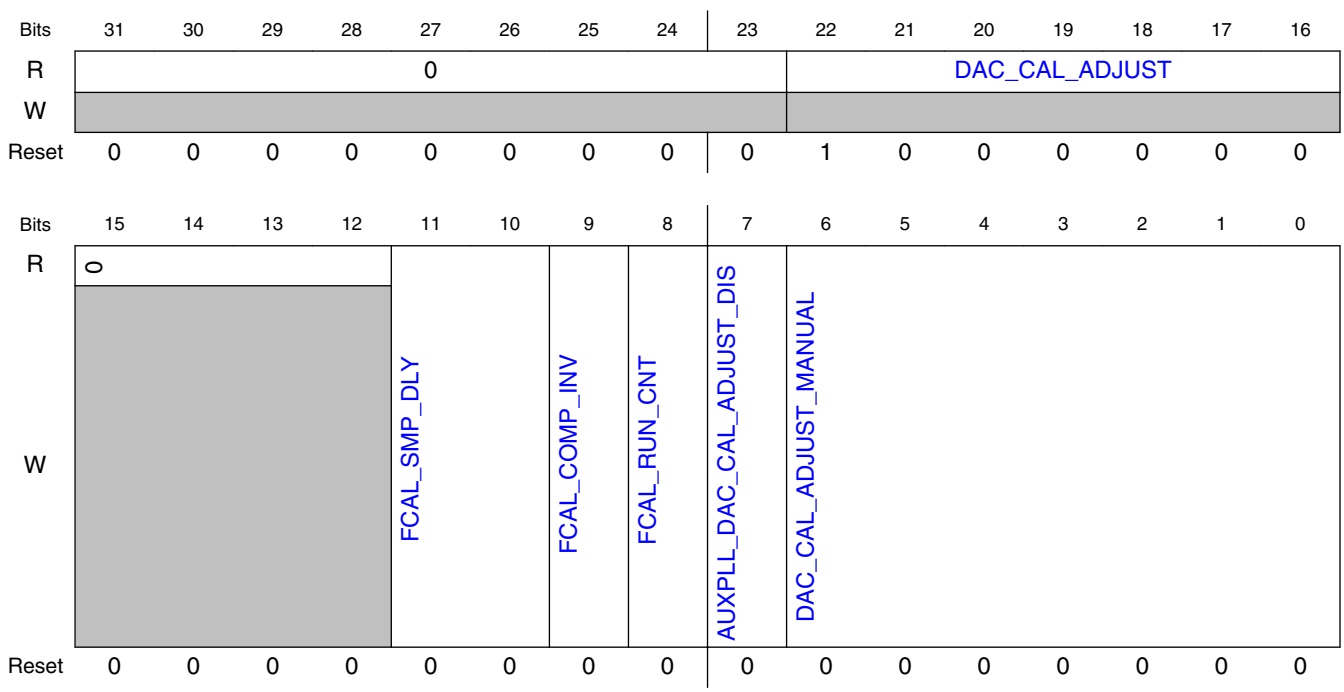
| Field | Function |
|---|---|
| BBA2_RCCAL | The RC Calibration value used for BBA2 |
| 9-5<br>ADC_RCCAL | ADC RC Calibration<br>The RC Calibration value used for ADC |
| 4-0<br>RCCAL_CODE | RC Calibration code<br>The RC Calibration code currently applied to the calibration circuit |

## 64.3.3.3.1.82 Aux PLL Frequency Calibration Control (AUXPLL_FCAL_CTRL)

### *64.3.3.3.1.82.1 Offset*

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CTRL | 1E8h |

### *64.3.3.3.1.82.2 Diagram*




### *64.3.3.3.1.82.3 Fields*

| Field | Function |
|---|---|
| 31-23 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 22-16<br>DAC_CAL_ADJUST | Aux PLL DAC Calibration Adjust value<br>The DAC calibration adjust value applied to the calibration circuit. |
| 15-12<br>— | Reserved. |
| 11-10<br>FCAL_SMP_DLY | Aux PLL Frequency Calibration Sample Delay<br>00b - The count signal is sampled 1 clk cycle after fcal_run signal is deasserted<br>01b - The count signal is sampled 2 clk cycle after fcal_run signal is deasserted<br>10b - The count signal is sampled 3 clk cycle after fcal_run signal is deasserted<br>11b - The count signal is sampled 4 clk cycle after fcal_run signal is deasserted |
| 9<br>FCAL_COMP_INV | Aux PLL Frequency Calibration Comparison Invert<br>0b - (Default) The comparison associated with the count is not inverted.<br>1b - The comparison associated with the count is inverted |
| 8<br>FCAL_RUN_CNT | Aux PLL Frequency Calibration Run Count<br>0b - Run count is 256 clock cycles<br>1b - Run count is 512 clock cycles |
| 7<br>AUXPLL_DAC_CAL_ADJUST_DIS | Aux PLL Frequency Calibration Disable<br>0b - Calibration is enabled<br>1b - Calibration is disabled |
| 6-0<br>DAC_CAL_ADJUST_MANUAL | Aux PLL Frequency DAC Calibration Adjust Manual value<br>If AUXPLL_DAC_CAL_ADJUST_DIS bit is set, this DAC calibration adjust value is applied to the calibration circuit. |

## 64.3.3.3.1.83 Aux PLL Frequency Calibration Count 6 (AUXPLL_FCAL_CNT6)

### *64.3.3.3.1.83.1 Offset*

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT6 | 1ECh |

#### 64.3.3.3.1.83.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_BESTDIFF | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_6 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.83.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_BESTDIFF | Aux PLL Frequency Calibration Best Difference<br>The smallest absolute difference between the count value output by the calibration circuit and the expected count value found during the calibration sequence. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_6 | Aux PLL Frequency Calibration Count 6<br>The count value output by the calibration circuit for calibration phase 6. |

### 64.3.3.3.1.84 Aux PLL Frequency Calibration Count 5 and 4 (AUXPLL_FCAL_CNT5_4)

#### 64.3.3.3.1.84.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT5_4 | 1F0h |

#### 64.3.3.3.1.84.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_5 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_4 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.84.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_5 | Aux PLL Frequency Calibration Count 5<br>The count value output by the calibration circuit for calibration phase 5. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_4 | Aux PLL Frequency Calibration Count 4<br>The count value output by the calibration circuit for calibration phase 4. |

### 64.3.3.3.1.85 Aux PLL Frequency Calibration Count 3 and 2 (AUXPLL_FCAL_CNT3_2)

#### 64.3.3.3.1.85.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT3_2 | 1F4h |

#### 64.3.3.3.1.85.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_3 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_2 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.85.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_3 | Aux PLL Frequency Calibration Count 3<br>The count value output by the calibration circuit for calibration phase 3. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_2 | Aux PLL Frequency Calibration Count 2<br>The count value output by the calibration circuit for calibration phase 2. |

### 64.3.3.3.1.86 Aux PLL Frequency Calibration Count 1 and 0 (AUXPLL_FCAL_CNT1_0)

#### 64.3.3.3.1.86.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT1_0 | 1F8h |

#### 64.3.3.3.1.86.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_1 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.3.3.1.86.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_1 | Frequency Calibration Count 1<br>The count value output by the calibration circuit for calibration phase 1. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_0 | Frequency Calibration Count 0<br>The count value output by the calibration circuit for calibration phase 0. |

## 64.3.4 Bit Streaming Mode

### 64.3.4.1 Introduction

The 2.4GHz Radio features a Bit Streaming Mode for BLE. When activated, this feature allows all received BLE packet data, to be serialized and shifted out to external hardware for further processing.

#### 64.3.4.1.1 Overview

Bit Streaming Mode (BSM) allows all received BLE packet data, to be serialized and shifted out to external hardware for further processing. A simple development system can be crafted to consume the BSM outputs and generate packet trace data for all BLE traffic

appearing on a network, within range of the receiving device. This enables network-level monitoring and debugging. BSM-for-BLE uses a simple, synchronous 3-wire interface, consisting of BSM_CLK, BSM_DATA, and BSM_FRAME outputs. Packet data is shifted out serially, at the BLE bit rate (1MHz). Signalling is provided on BSM_FRAME to indicate the start of reception. BSM_DATA and BSM_FRAME are synchronous to BSM_CLK. BSM_DATA and BSM_FRAME are valid at rising BSM_CLK, and are intended to be captured on that edge. A single control bit activates or deactivates BSM. Aside from controlling this bit, BSM requires no software support from the host processor. The SoC's BSM outputs are multiplexed with GPIO, so that the pins are available for general purpose use when BSM is disabled. BSM does not interfere with BLE packet processing in any way; it is merely a passive monitoring and debugging tool. BSM, when engaged, will not measurably increase current consumption, since the BSM module operates at a 1MHz clock rate. Only received packets are made available to BSM; packets transmitted by the device do not appear on the BSM interface.

This BSM implementation features some new automation not included in previous BSM versions. Access Address checking is now done in hardware, not software. No protocol engine (link layer) is engaged, so software communicates directly with the transceiver (PHY), reducing initialization overhead. And there are no "blind spots" due to link layer SCAN_INTERVAL timeouts. Packets can be received at any time under direct software control.

#### 64.3.4.1.2 Features

- Enables monitoring and debugging of all network traffic within range of the device
- Simple, synchronous 3-wire interface facilitates external hardware development
- Low power consumption
- Non-intrusive design
- Minimal software support required
- Hardware Access Address Correlation
- No Link Layer Overhead

### 64.3.4.1.3 Block Diagram

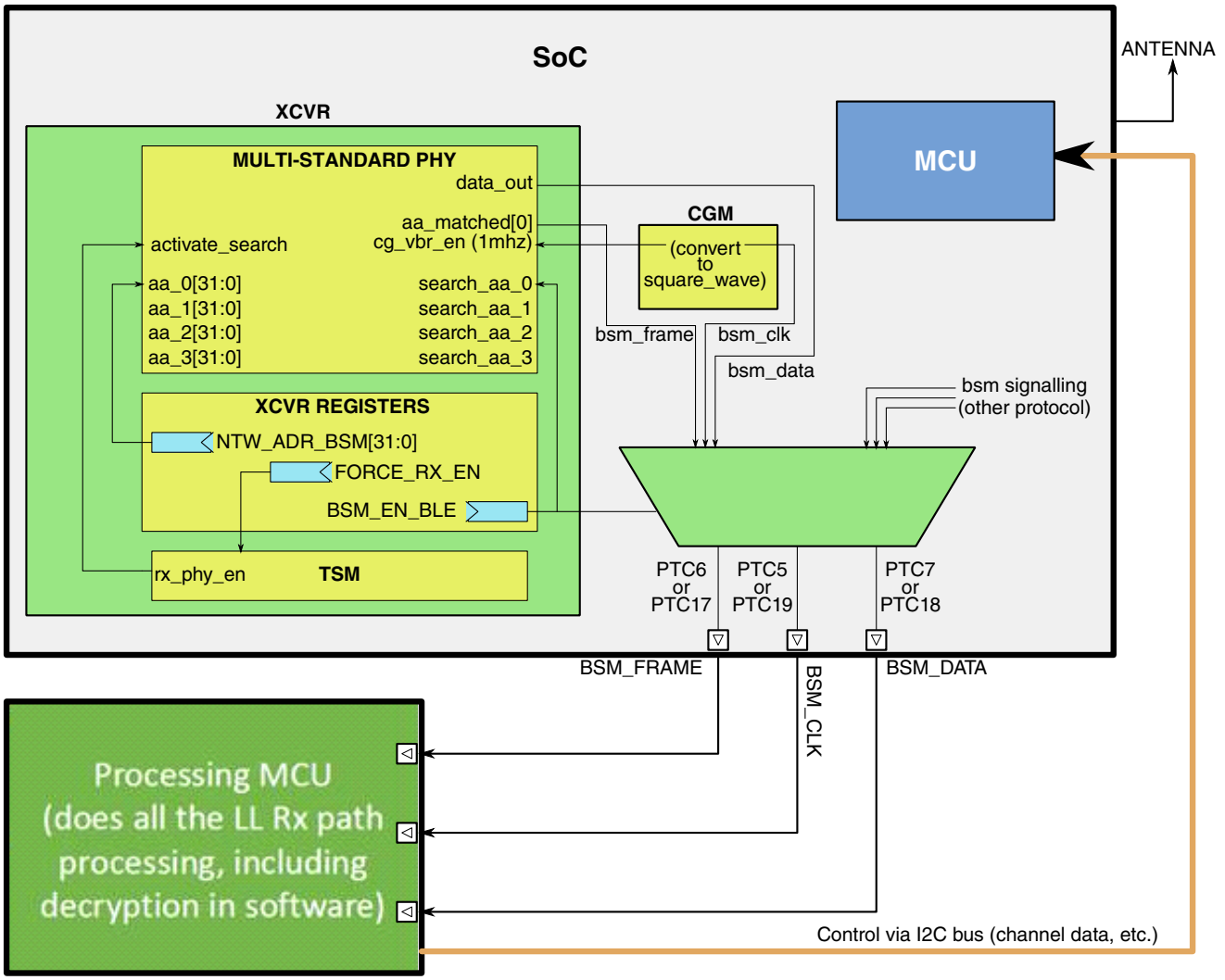



**Figure 64-58. BSM Block Diagram**

### 64.3.4.2 External Signal Descriptions

| Name | Direction | Size | Description |
|---|---|---|---|
| BSM_DATA | output | 1 | Serial BLE packet bit stream, LSB-first. Valid on rising edge of BSM_CLK |
| BSM_FRAME | output | 1 | Framing signal to indicate the start of reception. Active high |
| BSM_CLK | output | 1 | 1MHz bit-rate clock. BSM_DATA and BSM_FRAME are synchronized to BSM_CLK. |

| Name | Direction | Size | Descriptiom |
|---|---|---|---|
| | | | External device should capture BSM_FRAME and BSM_DATA on rising edge of BSM_CLK |

### 64.3.4.3 Memory Map and Register Definition

BSM-for-BLE is enabled by a single register bit. The bit is BSM_EN_BLE. This bit resides in the FSK_CFG register of XCVR address space. Several other programmable configuration parameters pertaining to BSM mode are listed in the table below.

| Register | Address | Field | Bit(s) | Desciption |
|---|---|---|---|---|
| FSK_CFG | XCVR_BASE + 0x408 | BSM_EN_BLE | [3] | 1: enable BSM (bit streaming mode) for BLE<br>0:disable BSM<br><br>When enabled, the 3 BSM outputs (BSM_DATA, BSM_FRAME, and BSM_BLK) appear on the BSM interface pins of the SoC. At the SoC level, these are alternate, muxed-GPIO pins, so the appropriate port programming is required. |
| NTW_ADR_BSM | XCVR_BASE + 0x410 | NTW_ADR_BSM[31:0] | [31:0] | PHY will search for this Access Address when BSM_EN_BLE=1 |
| FSK_CFG | XCVR_BASE + 0x408 | BLE_NTW_ADR_THR[2:0] | [22:20] | PHY will tolerate this many bit errors in its search for NTW_ADR_BSM |
| TSM_CTRL | XCVR_BASE + 0x2C0 | FORCE_RX_EN | [3] | Software will set FORCE_RX_EN=1 to launch an RX sequence in search of a packet with an Access Address equal to NTW_ADR_BSM. Software will clear FORCE_RX_EN=0 after the last bit of the packet has been received and processed by the application. Clearing this bit ends the RX sequence. |

### 64.3.4.4 Functional Description

The BSM implementation configures the device to enter receive mode and search the airwaves for BLE packets bearing an Access Address matching the one programmed by the application. In this mode, the application communicates directly with the Multi-standard PHY, and no link layers are engaged. Once configured and enabled for searching, BLE packets with an Access Address match will appear at the BSM pins of the device. An Access Address match will result in the device asserting BSM_FRAME=1. At this point, the device will begin shifting out the packet bits serially, LSB first, starting with the packet header (preamble and Access Address will *not* appear on the BSM pins). The BSM pins are a synchronous interface to an external MCU. BSM_FRAME and BSM_DATA bits change on the falling edge of BSM_CLK. The external device must capture BSM_FRAME and BSM_DATA on the rising edge. The application is responsible for all packet processing after the AA match (BSM_FRAME=1), including header parsing and CRC checking. The application terminates the receive operation once the last bit of the packet has been received and processed by the application. The number of bit errors tolerated by the PHY in its correlation of Access Address is programmable, with a default setting of 1 bit error tolerated. The BSM hardware on the device has no knowlege of Bluetooth Low Energy states, so it is up to the application to track BLE state (e.g., Advertising, Initiating), and configure the Access Address accordingly.

The following steps can be used to receive a BLE packet in BSM mode:

1. Configure the RX digital for BLE protocol
2. Configure the PHY for BLE protocol
3. Set FSK_CFG[BSM_EN_BLE]=1 to enable BSM feature
4. Program the desired Access Address into the NTW_ADR_BSM register
5. Program the desired maximum number of tolerated bit errors into FSK_CFG[BTLE_NTW_ADR_THR]
6. Set TSM_CTRL[FORCE_RX_EN]=1 to initiate the RX sequence
7. Monitor the BSM interface for BSM_FRAME=1 (Access Address match)
8. After BSM_FRAME=1, use each rising edge of BSM_CLK to capture the next bit of BSM_DATA
9. Shift in and parse the incoming packet on the fly, using the header to determine packet length
10. After the last bit of CRC, terminate the RX sequence by clearing TSM_CTRL[FORCE_RX_EN]=0.

The following diagram depicts the occurrence of a BLE packet which matches the programmed Access Address at the BSM interface:

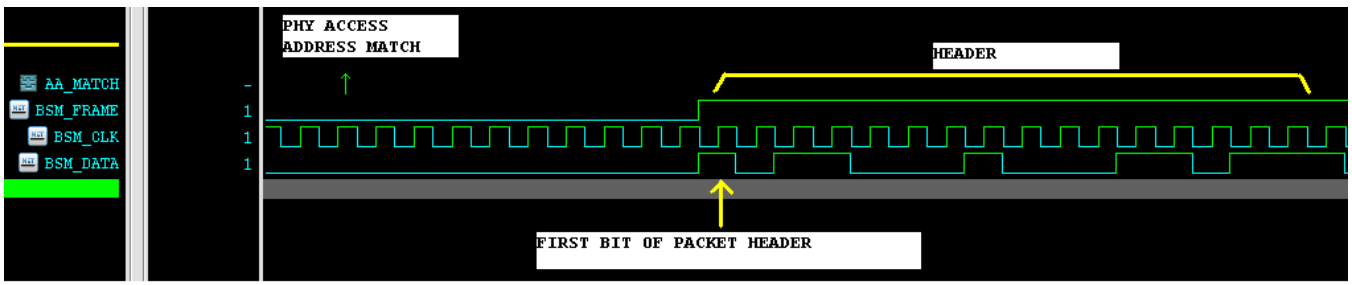



**Figure 64-59. BSM Timing Diagram**

The following sequence of events is depicted in the diagram

1. AA match occurs internally in the PHY
2. A delay of several microseconds after internal PHY AA match (transparent to application)
3. BSM_FRAME is asserted
4. At the next rising edge of BSM_CLK, first bit of packet header (LSB) is valid
5. Subsequent header bits are shifted out LSB-first on BSM_DATA, valid at each rising edge of BSM_CLK
6. Payload and CRC bits follow the header

The FSK_STAT register (XCVR_BASE + 0x414) contains several fields which may assist software monitoring the BSM interface:

| Field | Bit(s) | Desciption |
|---|---|---|
| PREAMBLE_FOUND | [0] | PHY has detected a BLE preamble. This should not be relied on by the application but it may be useful for debug. In the hardware, preamble detection is a prerequisite for Access Address correlation. |
| AA_MATCHED | [1] | PHY has detected an Access Address match. After a delay of several microseconds, BSM_FRAME will be asserted and packet data bit shifting will commence. |
| AA_MATCH[0] | [4] | Indicates that the Access Address match occurred on the PHY's Network Address 0 input. This will always be the case for BLE. |

Software could elect to poll the AA_MATCHED bit, to determine when to start monitoring the BSM_FRAME, since this bit will transition high several microseconds before BSM_FRAME=1. A procedure for polling AA_MATCHED as a prerequisite to the monitoring of the BSM interface pins, is as follows:

1. Ascertain the length of the RX Warmup by reading the third byte of the END_OF_SEQ register in XCVR space
2. Set variable **end_of_rx_wu** to this value: **end_of_seq** = (r32(END_OF_SEQ) & 0xFF0000) >> 16
3. Set TSM_CTRL[FORCE_RX_EN]=1 to initiate the RX sequence. (AA_MATCHED may still be asserted here from the previous packet reception)
4. Poll XCVR_STATUS register in XCVR space to determine when RX Warmup is complete. Set variable **tsm_count** to the LS byte of XCVR_STATUS
5. while (**tsm_count** < **end_of_rx_wu**) {**tsm_count** = (r32(XCVR_STATUS) & 0xFF);}
6. When the while() loop exits, the RX Warmup is complete. BSM_FRAME will assert soon
7. Begin monitoring the BSM pins for incoming BLE packet

## 64.3.5 Transceiver Sequence Manager

### 64.3.5.1 Introduction

The TSM is a fully-programmable, multi-protocol transceiver sequence manager, which supports Bluetooth Low Energy, 802.15.4, and other 2.4GHz transceivers.

#### 64.3.5.1.1 Overview

The Transceiver Sequence Manager controls the warmup and warmdown processes for all radio sequences. The TSM provides for a TX sequence, and an RX sequence. For both TX and RX, a warmup and a warmdown sequence is supported. The length of each sequence is programmable, from 1 – 254us. The resolution of the TSM is 1us. Controls for all analog and digital transceiver blocks are provided. The TSM has 67 outputs with which to control the warmup and warmdown processes. Each TSM output enables, or otherwise provides control for, a transceiver-related block. Each TSM output (or group of outputs) has 4 8-bit registers, with which to control the point at which the output asserts during the warmup (1 register for TX, 1 for RX), and the point at which it deasserts (1 register for TX, 1 for RX). Some TSM outputs, which are known in advance to require identical timing, are “ganged together”, to reduce area and required programming. For

DFT, and non-mission-mode validation, each TSM output can be put under direct software control, using register overrides. Any sequence can be optionally aborted by software, or by a PLL unlock condition. And, any sequence can be temporarily halted, by setting an optional, programmable breakpoint.

#### 64.3.5.1.2 Features

The TSM includes the following features:

- TX warmup and warmdown sequence and RX warmup and warmdown sequence sequence for every controllable output.
- Programmable sequence length from 1 to 254 microseconds for each sequence in 1 microsecond increments.
- 67 total outputs.
- Multi-protocol
- PA Target Power Selection
- PA Ramping
- Ability to place TSM outputs under software control for test and non-mission mode validation.
- Ability to abort sequences upon PLL unlock condition.
- Fast Warmup Capability for TX and RX
- LPPS Support
- Programmable breakpoint capability to halt any sequence temporarily.

### 64.3.5.1.3 Block diagram

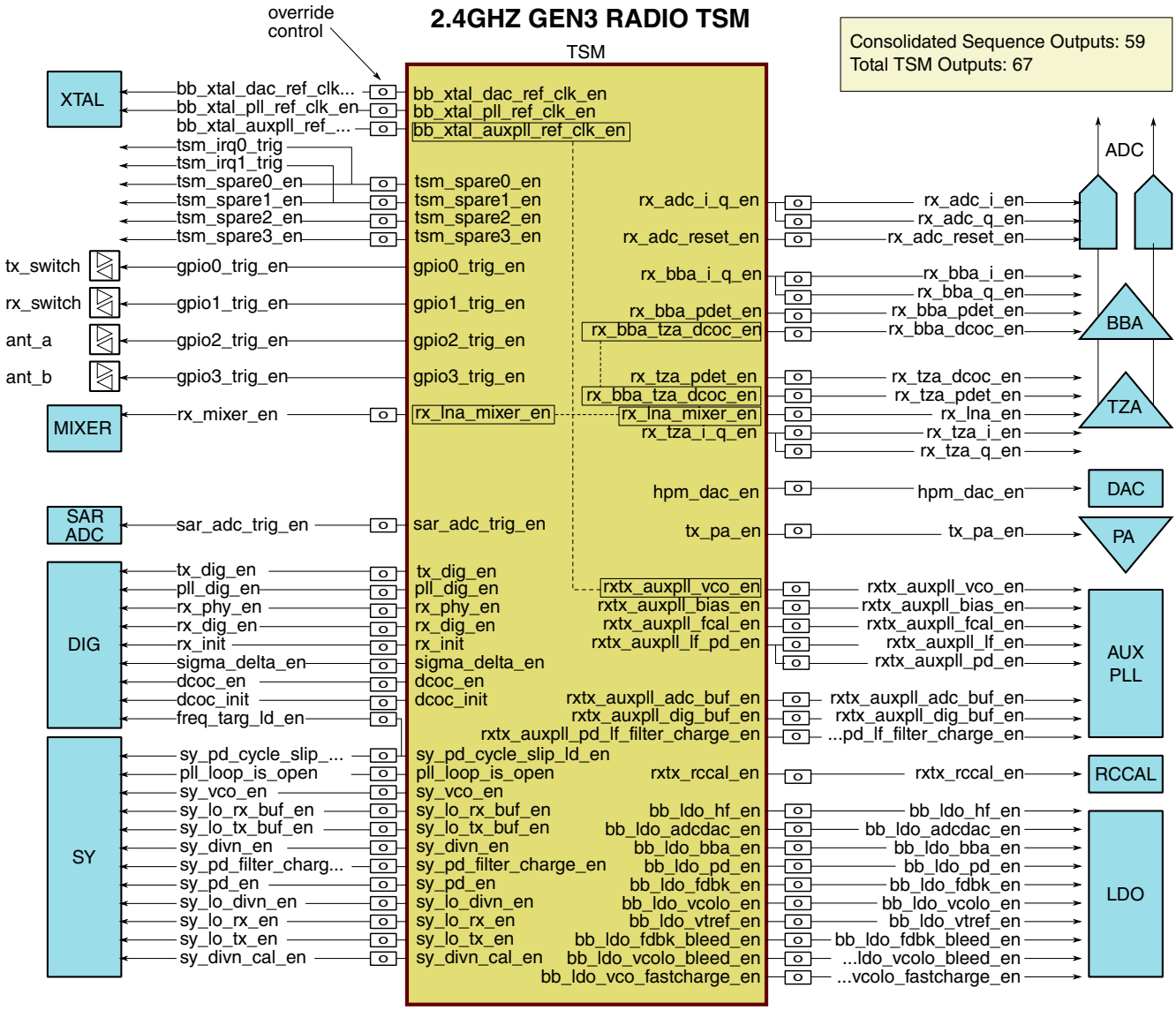



**Figure 64-60. Block diagram**

## 64.3.5.2 Functional description

### 64.3.5.2.1 Sequence Counter

The TSM supports 1 TX and 1 RX sequence. Each sequence consists of 3 phases:

1. WARMUP phase
2. ON phase
3. WARMDOWN phase

The central element of the TSM is an 8-bit counter. The counter is held at 0 during the idle state. From idle state, any sequence can be launched, by an initiating event. (See Section Sequence Initiation). At an initiating event, the TSM counter counts up from 0 to a programmed stop point during the WARMUP phase, determined by the END_OF_TX_WU or END_OF_RX_WU register, depending on whether the sequence is TX or RX. At the end-of-warmup “stop point”, the TSM counter holds its count, and the TSM sequence enters the ON phase. The TSM counter will remain in the ON phase, and hold its count, until the initiating event deasserts, or an abort occurs (See Section Sequence Termination). When either of these conditions occurs during the ON phase, the TSM will resume counting from the point it was holding during the ON phase, and the sequence will enter the WARMDOWN phase. The counter will continue counting until a programmed stop point is reached. This stop point is determined by the END_OF_TX_WD or END_OF_RX_WD register, depending on whether the sequence is TX or RX. Once this point is reached, the sequence returns to idle, and the TSM counter returns to 0.

The 4 8-bit registers which control the duration of the WARMUP phase (END_OF_TX_WU[7:0] and END_OF_RX_WU[7:0]), and the duration of the WARMDOWN phase (END_OF_TX_WD[7:0] and END_OF_RX_WD[7:0]), reside in the END_OF_SEQ register.

All TSM-controlled outputs are active-high. Any TSM-controlled output can be asserted once during a sequence (either TX or RX sequence), and then deasserted once. Or, the output can be held in a deasserted state for the duration of the sequence. The precise timing for the assertion and deassertion of each TSM-controlled output, for both TX and RX sequences, is determined by 4 8-bit registers assigned to that output (see Section TSM-Controlled Outputs). The TSM resolution is 1us. This is the update rate for the TSM counter, and sets the granularity with which warmup and warmdown processes can be controlled.

#### 64.3.5.2.2 TSM-Controlled Outputs

The TSM has 67 timing-controlled outputs, for enabling and control of the various transceiver blocks. The outputs are shown in the following diagram.

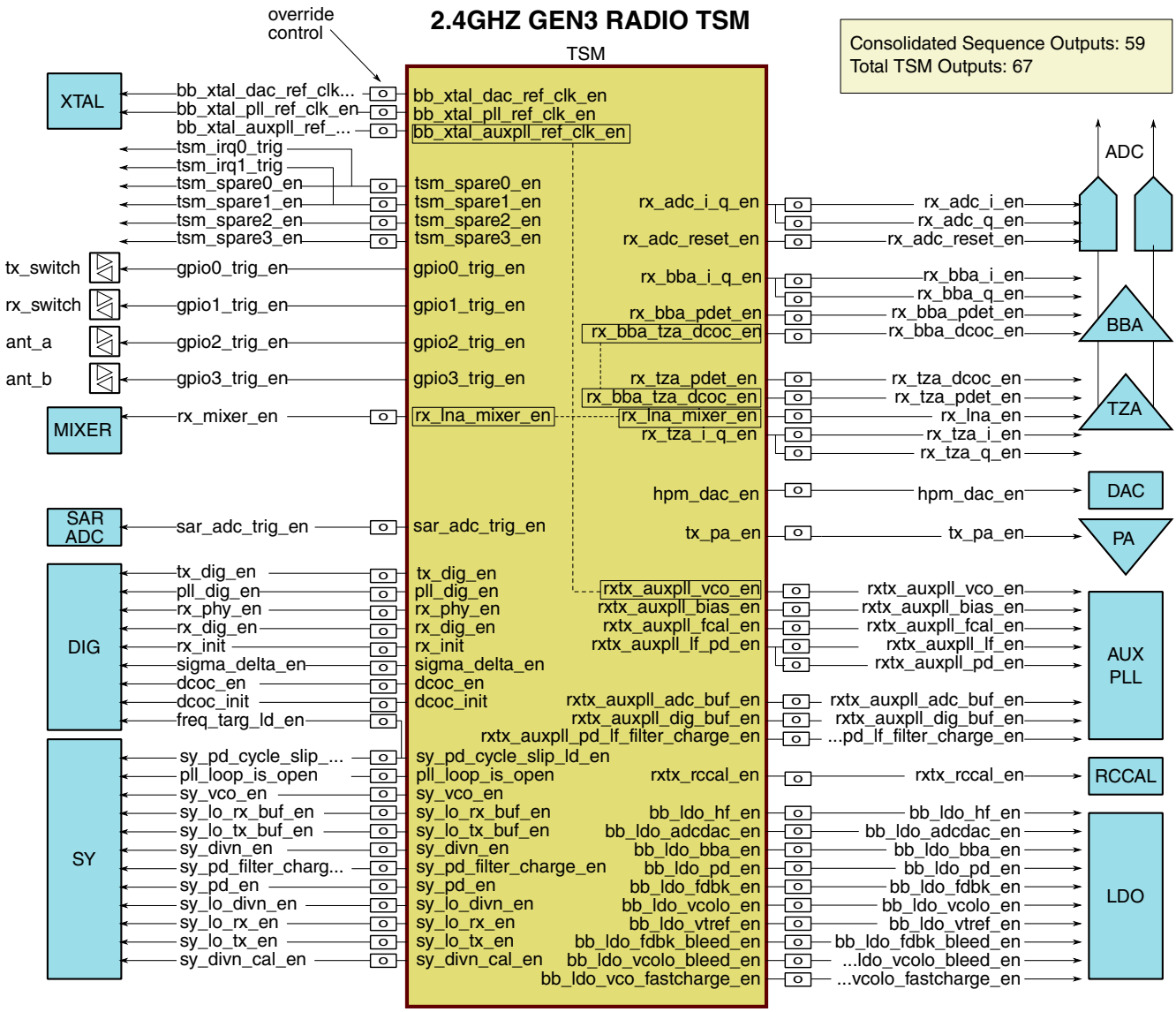



**Figure 64-61. 2.4GHZ RADIO TSM**

In some cases, some of the 67 outputs are grouped together, for outputs which need identical timing. For example, rx_bba_i_en and rx_bba_q_en. These groupings save area, because only 1 set of timing registers is needed to control timing for the entire group. Groupings also reduce the amount of required TSM programming. In all cases, even for grouped outputs, each output has an individual override control, to allow software to take command of any of the 67 TSM-controlled outputs at any time. As a consequence of the groupings, there are only 59 sets of timing registers, to control the 67 outputs. The table below, lists the 67 TSM-controlled outputs, and for the grouped outputs, indicates the signal which controls the timing for the entire group.

| Signal Index | 67 TSM-CONTROLLED OUTPUTS | SIGNAL CONTROLLING GROUP TIMING |
|---|---|---|
| 0 | bb_ldo_hf_en | bb_ldo_hf_en |
| 1 | bb_ldo_adcdac_en | bb_ldo_adcdac_en |
| 2 | bb_ldo_bba_en | bb_ldo_bba_en |
| 3 | bb_ldo_pd_en | bb_ldo_pd_en |
| 4 | bb_ldo_fdbk_en | bb_ldo_fdbk_en |
| 5 | bb_ldo_vcolo_en | bb_ldo_vcolo_en |
| 6 | bb_ldo_vtref_en | bb_ldo_vtref_en |
| 7 | bb_ldo_fdbk_bleed_en | bb_ldo_fdbk_bleed_en |
| 8 | bb_ldo_vcolo_bleed_en | bb_ldo_vcolo_bleed_en |
| 9 | bb_ldo_vcolo_fastcharge_en | bb_ldo_vcolo_fastcharge_en |
| 10 | bb_xtal_pll_ref_clk_en | bb_xtal_pll_ref_clk_en |
| 11 | bb_xtal_dac_ref_clk_en | bb_xtal_dac_ref_clk_en |
| 12 | bb_xtal_auxpll_ref_clk_en | rxtx_auxpll_vco_ref_clk_en |
| 13 | pll_loop_is_open | pll_loop_is_open |
| 14 | sy_pd_cycle_slip_ld_en | sy_pd_cycle_slip_ld_ft_en |
| 15 | sy_vco_en | sy_vco_en |
| 16 | sy_lo_rx_buf_en | sy_lo_rx_buf_en |
| 17 | sy_lo_tx_buf_en | sy_lo_tx_buf_en |
| 18 | sy_divn_en | sy_divn_en |
| 19 | sy_pd_filter_charge_en | sy_pd_filter_charge_en |
| 20 | sy_pd_en | sy_pd_en |
| 21 | sy_lo_divn_en | sy_lo_divn_en |
| 22 | sy_lo_rx_en | sy_lo_rx_en |
| 23 | sy_lo_tx_en | sy_lo_tx_en |
| 24 | sy_divn_cal_en | sy_divn_cal_en |
| 25 | rx_mixer_en | rx_lna_mixer_en |
| 26 | tx_pa_en | tx_pa_en |
| 27 | rx_adc_i_en | rx_adc_i_q_en |
| 28 | rx_adc_q_en | rx_adc_i_q_en |
| 29 | rx_adc_reset_en | rx_adc_reset_en |
| 30 | rx_bba_i_en | rx_bba_i_q_en |
| 31 | rx_bba_q_en | rx_bba_i_q_en |
| 32 | rx_bba_pdet_en | rx_bba_pdet_en |
| 33 | rx_bba_dcoc_en | rx_bba_tza_dcoc_en |
| 34 | rx_lna_en | rx_lna_mixer_en |
| 35 | rx_tza_i_en | rx_tza_i_q_en |
| 36 | rx_tza_q_en | rx_tza_i_q_en |
| 37 | rx_tza_pdet_en | rx_tza_pdet_en |
| 38 | rx_tza_dcoc_en | rx_bba_tza_dcoc_en |
| 39 | pll_dig_en | pll_dig_en |

*Table continues on the next page...*

| Signal Index | 67 TSM-CONTROLLED OUTPUTS | SIGNAL CONTROLLING GROUP TIMING |
|---|---|---|
| 40 | tx_dig_en | tx_dig_en |
| 41 | rx_dig_en | rx_dig_en |
| 42 | rx_init | rx_init |
| 43 | sigma_delta_en | sigma_delta_en |
| 44 | rx_phy_en | rx_phy_en |
| 45 | dcoc_en | dcoc_en |
| 46 | dcoc_init | dcoc_init |
| 47 | freq_targ_ld_en | sy_pd_cycle_slip_ld_ft_en |
| 48 | sar_adc_trig_en | sar_adc_trig_en |
| 49 | tsm_spare0_en | tsm_spare0_en |
| 50 | tsm_spare1_en | tsm_spare1_en |
| 51 | tsm_spare2_en | tsm_spare2_en |
| 52 | tsm_spare3_en | tsm_spare3_en |
| 53 | gpio0_trig_en | gpio0_trig_en |
| 54 | gpio1_trig_en | gpio1_trig_en |
| 55 | gpio2_trig_en | gpio2_trig_en |
| 56 | gpio3_trig_en | gpio3_trig_en |
| 57 | rxtx_auxpll_bias_en | rxtx_auxpll_bias_en |
| 58 | rxtx_auxpll_vco_en | rxtx_auxpll_vco_ref_clk_en |
| 59 | rxtx_auxpll_fcal_en | rxtx_auxpll_fcal_en |
| 60 | rxtx_auxpll_lf_en | rxtx_auxpll_lf_pd_en |
| 61 | rxtx_auxpll_pd_en | rxtx_auxpll_lf_pd_en |
| 62 | rxtx_auxpll_pd_lf_filter_charge_en | rxtx_auxpll_pd_lf_filter_charge_en |
| 63 | rxtx_auxpll_adc_buf_en | rxtx_auxpll_adc_buf_en |
| 64 | rxtx_auxpll_dig_buf_en | rxtx_auxpll_dig_buf_en |
| 65 | rxtx_rccal_en | rxtx_rccal_en |
| 66 | tx_hpm_dac_en | tx_hpm_dac_en |

For example, TSM-controlled ouputs rx_bba_i_en, and rx_bba_q_en, are both members of the same group. The timing registers to program the timing on these outputs (RX_BBA_I_Q_TX_HI, RX_BBA_I_Q_TX_LO, RX_BBA_I_Q_RX_HI, RX_BBA_I_Q_RX_LO), control the timing for both members of the group. Each member of the group, however, has in independent software override.

During the WARMUP and WARMDOWN phases of any sequence, any TSM-controlled output can be programmed to assert once and then deassert once, by programming the timing registers associated with the output. Each output has 4 8-bit timing registers associated with it. The name of the register matches the name of the output:

*OUTPUTNAME*_TX_HI[7:0]
*OUTPUTNAME*_TX_LO[7:0]

*OUTPUTNAME*_RX_HI[7:0]
*OUTPUTNAME*_RX_LO[7:0]

In the case of grouped outputs, the timing registers associated with the signal controlling the group, control timing for the entire group.

During the WARMUP phase of a TX sequence, as the 8-bit TSM counter increments, once the TSM counter equals or exceeds the programmed value of *OUTPUTNAME*_TX_HI[7:0], but is less than the programmed value of *OUTPUTNAME*_TX_LO[7:0], the corresponding TSM output will transition high. During the WARMUP or WARMDOWN phase of the TX sequence, once the TSM counter equals or exceeds the programmed value of *OUTPUTNAME*_TX_LO[7:0], the TSM output will transition low. If the programmed value of *OUTPUTNAME*_TX_HI[7:0], is greater than the length of the TX sequence (set by register END_OF_TX_WD[7:0]), then that signal will never transition high during the TX sequence. A convenient way of ensuring a signal does not assert during a TX sequence, is to set its *OUTPUTNAME*_TX_HI[7:0]=255.

During the WARMUP phase of a RX sequence, as the 8-bit TSM counter increments, once the TSM counter equals or exceeds the programmed value of *OUTPUTNAME*_RX_HI[7:0], but is less than the programmed value of *OUTPUTNAME*_RX_LO[7:0], the corresponding TSM output will transition high. During the WARMUP or WARMDOWN phase of the RX sequence, once the TSM counter equals or exceeds the programmed value of *OUTPUTNAME*_RX_LO[7:0], the TSM output will transition low. If the programmed value of *OUTPUTNAME*_RX_HI[7:0], is greater than the length of the RX sequence (set by register END_OF_RX_WD[7:0]), then that signal will never transition high during the RX sequence. A convenient way of assuring a signal does not assert during a RX sequence, is to set its *OUTPUTNAME*_RX_HI[7:0]=255.

The following diagram depicts the TSM counter, and the control logic used to assert and deassert the TSM-controlled outputs. For clarity, logic for only 2 of the 67 outputs is shown.

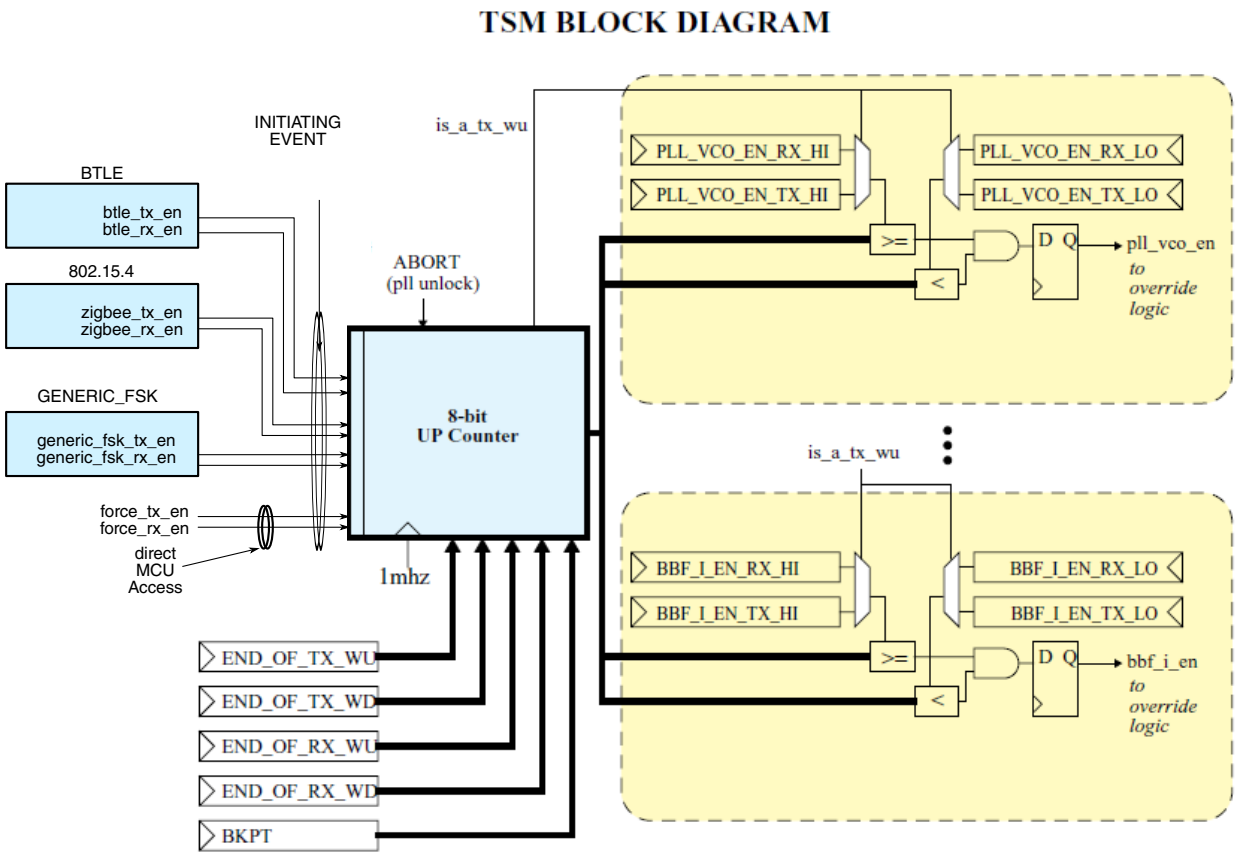



**Figure 64-62. 2.4GHZ RADIO TSM BLOCK DIAGRAM**

### 64.3.5.2.3 Timing Registers

There are 59 sets of timing registers for the TSM, one set for each TSM-controlled output (or, for the signal controlling the group for the “grouped” outputs). Each register set consists of 4 registers: one to control the assertion time of the output for TX sequences, one to control the assertion time for RX sequence, one to control the deassertion time for TX sequences, and one to control the deassertion time for RX sequences. For TSM-controlled outputs that are grouped, one set of timing registers controls the timing for all members of the group. For each of these 59 sets, the following table lists the register name (TSM_TIMING00 – TSM_TIMING58), and 4 timing register names associated with each.

| REGISTER NAME | BITS [31:24] | BITS [23:16] | BITS [15:8] | BITS [7:0] |
|---|---|---|---|---|
| TSM_TIMING00 | BB_LDO_HF_EN_RX_LO[7:0] | BB_LDO_HF_EN_RX_HI[7:0] | BB_LDO_HF_EN_TX_LO[7:0] | BB_LDO_HF_EN_TX_HI[7:0] |
| TSM_TIMING01 | BB_LDO_ADCDAC_EN_RX_LO[7:0] | BB_LDO_ADCDAC_EN_RX_HI[7:0] | BB_LDO_ADCDAC_EN_TX_LO[7:0] | BB_LDO_ADCDAC_EN_TX_HI[7:0] |

*Table continues on the next page...*

| REGISTER NAME | BITS [31:24] | BITS [23:16] | BITS [15:8] | BITS [7:0] |
|---|---|---|---|---|
| TSM_TIMING02 | BB_LDO_BBA_EN_RX_LO[7:0] | BB_LDO_BBA_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING03 | BB_LDO_PD_EN_RX_LO[7:0] | BB_LDO_PD_EN_RX_HI[7:0] | BB_LDO_PD_EN_TX_LO[7:0] | BB_LDO_PD_EN_TX_HI[7:0] |
| TSM_TIMING04 | BB_LDO_FDBK_EN_RX_LO[7:0] | BB_LDO_FDBK_EN_RX_HI[7:0] | BB_LDO_FDBK_EN_TX_LO[7:0] | BB_LDO_FDBK_EN_TX_HI[7:0] |
| TSM_TIMING05 | BB_LDO_VCOLO_EN_RX_LO[7:0] | BB_LDO_VCOLO_EN_RX_HI[7:0] | BB_LDO_VCOLO_EN_TX_LO[7:0] | BB_LDO_VCOLO_EN_TX_HI[7:0] |
| TSM_TIMING06 | BB_LDO_VTREF_EN_RX_LO[7:0] | BB_LDO_VTREF_EN_RX_HI[7:0] | BB_LDO_VTREF_EN_TX_LO[7:0] | BB_LDO_VTREF_EN_TX_HI[7:0] |
| TSM_TIMING07 | BB_LDO_FDBK_BLEED_EN_RX_LO[7:0] | BB_LDO_FDBK_BLEED_EN_RX_HI[7:0] | BB_LDO_FDBK_BLEED_EN_TX_LO[7:0] | BB_LDO_FDBK_BLEED_EN_TX_HI[7:0] |
| TSM_TIMING08 | BB_LDO_VCOLO_BLEED_EN_RX_LO[7:0] | BB_LDO_VCOLO_BLEED_EN_RX_HI[7:0] | BB_LDO_VCOLO_BLEED_EN_TX_LO[7:0] | BB_LDO_VCOLO_BLEED_EN_TX_HI[7:0] |
| TSM_TIMING09 | BB_LDO_VCOLO_FASTCHARGE_EN_RX_LO[7:0] | BB_LDO_VCOLO_FASTCHARGE_EN_RX_HI[7:0] | BB_LDO_VCOLO_FASTCHARGE_EN_TX_LO[7:0] | BB_LDO_VCOLO_FASTCHARGE_EN_TX_HI[7:0] |
| TSM_TIMING10 | BB_XTAL_PLL_REF_CLK_EN_RX_LO[7:0] | BB_XTAL_PLL_REF_CLK_EN_RX_HI[7:0] | BB_XTAL_PLL_REF_CLK_EN_TX_LO[7:0] | BB_XTAL_PLL_REF_CLK_EN_TX_HI[7:0] |
| TSM_TIMING11 | n/a | n/a | BB_XTAL_DAC_REF_CLK_EN_TX_LO[7:0] | BB_XTAL_DAC_REF_CLK_EN_TX_HI[7:0] |
| TSM_TIMING12 | RXTX_AUXPLL_VCO_REF_CLK_EN_RX_LO[7:0] | RXTX_AUXPLL_VCO_REF_CLK_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING13 | PLL_LOOP_IS_OPEN_RX_LO[7:0] | PLL_LOOP_IS_OPEN_RX_HI[7:0] | PLL_LOOP_IS_OPEN_TX_LO[7:0] | PLL_LOOP_IS_OPEN_TX_HI[7:0] |
| TSM_TIMING14 | SY_PD_CYCLE_SLIP_LD_FT_EN_RX_LO[7:0] | SY_PD_CYCLE_SLIP_LD_FT_EN_RX_HI[7:0] | SY_PD_CYCLE_SLIP_LD_FT_EN_TX_LO[7:0] | SY_PD_CYCLE_SLIP_LD_FT_EN_TX_HI[7:0] |
| TSM_TIMING15 | SY_VCO_EN_RX_LO[7:0] | SY_VCO_EN_RX_HI[7:0] | SY_VCO_EN_TX_LO[7:0] | SY_VCO_EN_TX_HI[7:0] |
| TSM_TIMING16 | SY_LO_RX_BUF_EN_RX_LO[7:0] | SY_LO_RX_BUF_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING17 | n/a | n/a | SY_LO_TX_BUF_EN_TX_LO[7:0] | SY_LO_TX_BUF_EN_TX_HI[7:0] |
| TSM_TIMING18 | SY_DIVN_EN_RX_LO[7:0] | SY_DIVN_EN_RX_HI[7:0] | SY_DIVN_EN_TX_LO[7:0] | SY_DIVN_EN_TX_HI[7:0] |
| TSM_TIMING19 | SY_PD_FILTER_CHARGE_EN_RX_LO[7:0] | SY_PD_FILTER_CHARGE_EN_RX_HI[7:0] | SY_PD_FILTER_CHARGE_EN_TX_LO[7:0] | SY_PD_FILTER_CHARGE_EN_TX_HI[7:0] |
| TSM_TIMING20 | SY_PD_EN_RX_LO[7:0] | SY_PD_EN_RX_HI[7:0] | SY_PD_EN_TX_LO[7:0] | SY_PD_EN_TX_HI[7:0] |
| TSM_TIMING21 | SY_LO_DIVN_EN_RX_LO[7:0] | SY_LO_DIVN_EN_RX_HI[7:0] | SY_LO_DIVN_EN_TX_LO[7:0] | SY_LO_DIVN_EN_TX_HI[7:0] |
| TSM_TIMING22 | SY_LO_RX_EN_RX_LO[7:0] | SY_LO_RX_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING23 | n/a | n/a | SY_LO_TX_EN_TX_LO[7:0] | SY_LO_TX_EN_TX_HI[7:0] |

*Table continues on the next page...*

| REGISTER NAME | BITS [31:24] | BITS [23:16] | BITS [15:8] | BITS [7:0] |
|---|---|---|---|---|
| TSM_TIMING24 | SY_DIVN_CAL_EN_RX_LO[7:0] | SY_DIVN_CAL_EN_RX_HI[7:0] | SY_DIVN_CAL_EN_TX_LO[7:0] | SY_DIVN_CAL_EN_TX_HI[7:0] |
| TSM_TIMING25 | RX_LNA_MIXER_EN_RX_LO[7:0] | RX_LNA_MIXER_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING26 | n/a | n/a | TX_PA_EN_TX_LO[7:0] | TX_PA_EN_TX_HI[7:0] |
| TSM_TIMING27 | RX_ADC_I_Q_EN_RX_LO[7:0] | RX_ADC_I_Q_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING28 | RX_ADC_RESET_EN_RX_LO[7:0] | RX_ADC_RESET_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING29 | RX_BBA_I_Q_EN_RX_LO[7:0] | RX_BBA_I_Q_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING30 | RX_BBA_PDET_EN_RX_LO[7:0] | RX_BBA_PDET_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING31 | RX_BBA_TZA_DCOC_EN_RX_LO[7:0] | RX_BBA_TZA_DCOC_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING32 | RX_TZA_I_Q_EN_RX_LO[7:0] | RX_TZA_I_Q_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING33 | RX_TZA_PDET_EN_RX_LO[7:0] | RX_TZA_PDET_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING34 | PLL_DIG_EN_RX_LO[7:0] | PLL_DIG_EN_RX_HI[7:0] | PLL_DIG_EN_TX_LO[7:0] | PLL_DIG_EN_TX_HI[7:0] |
| TSM_TIMING35 | n/a | n/a | TX_DIG_EN_TX_LO[7:0] | TX_DIG_EN_TX_HI[7:0] |
| TSM_TIMING36 | RX_DIG_EN_RX_LO[7:0] | RX_DIG_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING37 | RX_INIT_RX_LO[7:0] | RX_INIT_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING38 | SIGMA_DELTA_EN_RX_LO[7:0] | SIGMA_DELTA_EN_RX_HI[7:0] | SIGMA_DELTA_EN_TX_LO[7:0] | SIGMA_DELTA_EN_TX_HI[7:0] |
| TSM_TIMING39 | RX_PHY_EN_RX_LO[7:0] | RX_PHY_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING40 | DCOC_EN_RX_LO[7:0] | DCOC_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING41 | DCOC_INIT_RX_LO[7:0] | DCOC_INIT_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING42 | SAR_ADC_TRIG_EN_RX_LO[7:0] | SAR_ADC_TRIG_EN_RX_HI[7:0] | SAR_ADC_TRIG_EN_TX_LO[7:0] | SAR_ADC_TRIG_EN_TX_HI[7:0] |
| TSM_TIMING43 | TSM_SPARE0_EN_RX_LO[7:0] | TSM_SPARE0_EN_RX_HI[7:0] | TSM_SPARE0_EN_TX_LO[7:0] | TSM_SPARE0_EN_TX_HI[7:0] |
| TSM_TIMING44 | TSM_SPARE1_EN_RX_LO[7:0] | TSM_SPARE1_EN_RX_HI[7:0] | TSM_SPARE1_EN_TX_LO[7:0] | TSM_SPARE1_EN_TX_HI[7:0] |
| TSM_TIMING45 | TSM_SPARE2_EN_RX_LO[7:0] | TSM_SPARE2_EN_RX_HI[7:0] | TSM_SPARE2_EN_TX_LO[7:0] | TSM_SPARE2_EN_TX_HI[7:0] |
| TSM_TIMING46 | TSM_SPARE3_EN_RX_LO[7:0] | TSM_SPARE3_EN_RX_HI[7:0] | TSM_SPARE3_EN_TX_LO[7:0] | TSM_SPARE3_EN_TX_HI[7:0] |
| TSM_TIMING47 | GPIO0_TRIG_EN_RX_LO[7:0] | GPIO0_TRIG_EN_RX_HI[7:0] | GPIO0_TRIG_EN_TX_LO[7:0] | GPIO0_TRIG_EN_TX_HI[7:0] |

*Table continues on the next page...*

| REGISTER NAME | BITS [31:24] | BITS [23:16] | BITS [15:8] | BITS [7:0] |
|---|---|---|---|---|
| TSM_TIMING48 | GPIO1_TRIG_EN_RX_LO[7:0] | GPIO1_TRIG_EN_RX_HI[7:0] | GPIO1_TRIG_EN_TX_LO[7:0] | GPIO1_TRIG_EN_TX_HI[7:0] |
| TSM_TIMING49 | GPIO2_TRIG_EN_RX_LO[7:0] | GPIO2_TRIG_EN_RX_HI[7:0] | GPIO2_TRIG_EN_TX_LO[7:0] | GPIO2_TRIG_EN_TX_HI[7:0] |
| TSM_TIMING50 | GPIO3_TRIG_EN_RX_LO[7:0] | GPIO3_TRIG_EN_RX_HI[7:0] | GPIO3_TRIG_EN_TX_LO[7:0] | GPIO3_TRIG_EN_TX_HI[7:0] |
| TSM_TIMING51 | RXTX_AUXPLL_BIAS_EN_RX_LO[7:0] | RXTX_AUXPLL_BIAS_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING52 | RXTX_AUXPLL_FCAL_EN_RX_LO[7:0] | RXTX_AUXPLL_FCAL_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING53 | RXTX_AUXPLL_LF_PD_EN_RX_LO[7:0] | RXTX_AUXPLL_LF_PD_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING54 | RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_LO[7:0] | RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING55 | RXTX_AUXPLL_ADC_BUF_EN_RX_LO[7:0] | RXTX_AUXPLL_ADC_BUF_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING56 | RXTX_AUXPLL_DIG_BUF_EN_RX_LO[7:0] | RXTX_AUXPLL_DIG_BUF_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING57 | RXTX_RCCAL_EN_RX_LO[7:0] | RXTX_RCCAL_EN_RX_HI[7:0] | n/a | n/a |
| TSM_TIMING58 | n/a | n/a | TX_HPM_DAC_EN_TX_LO[7:0] | TX_HPM_DAC_EN_TX_HI[7:0] |

Some of the TSM outputs have relevance for TX sequences only. For those, timing registers are only provided for TX assertion and deassertion times. For such outputs, the RX timing registers are shown as “n/a” in the table above. The RX timings for such outputs are essentially hardwired to 255, meaning there will be no signal assertion during RX sequences.

Some of the TSM outputs have relevance for RX sequences only. For those, timing registers are only provided for RX assertion and deassertion times. For such outputs, the TX timing registers are shown as “n/a” in the table above. The TX timings for such outputs are essentially hardwired to 255, meaning there will be no signal assertion during TX sequences.

#### 64.3.5.2.4 Sequence Initiation

The SoC will include one or more protocol engines. Any of the protocol engines, can launch a TSM sequence. Each protocol engine will have an initiating signal for a TX sequence, and another for an RX sequence. In lieu of a protocol engine, the SoC host can launch a TSM sequence directly. The following table lists the initiating sources supported by the TSM, and the name of the sequence launching signal (TSM input):

| Initiating Source | Initiating Signal Names |
|---|---|
| BTLE | btle_tx_en, btle_rx_en |
| 802.15.4 | zigbee_tx_en, zigbee_rx_en |
| Generic FSK | generic_tx_en, generic_rx_en |
| MCU/Host | force_tx_en, force_rx_en |

For direct software control, the MCU/Host can initiate a TX sequence warmup directly, by setting the FORCE_TX_EN bit, and later initiate a TX sequence warmdown by clearing the bit. The MCU/Host can initiate a RX sequence warmup directly, by setting the FORCE_RX_EN bit, and later initiate a RX sequence warmdown by clearing the bit. These bits reside in the TSM_CTRL register.

From idle state, the asserting of any initiating event (*source*_tx_en or *source*_rx_en), will begin the TSM WARMUP phase and begin incrementing the TSM counter from 0. The initiating source needs to hold the initiating signal asserted (high) through the course of the WARMUP, and then for the duration of the desired ON phase. Deasserting the initiating signal transitions the TSM to the WARMDOWN phase (see Section Sequence Termination).

From idle state, a TX-initiating event (assertion of any TSM *source*_tx_en input) will launch a TX sequence. The TSM output **tx_mode** will go high to indicate a TX sequence is underway. Also, from idle state, an RX-initiating event (assertion of any TSM *source*_rx_en input) will launch an RX sequence. The TSM output **rx_mode** will go high to indicate a RX sequence is underway.

Only 1 initiating source can be allowed to assert at any given time. Under no circumstances should a TX and RX initiating source be asserted simultaneously.

Once a TSM sequence has been launched, the WARMUP phase is entered and TSM counter incrementing will commence. Up-counting will continue until one of the following conditions is met:

1. Deassertion of the initiating event. TSM will transition to WARMDOWN phase
2. Abort. TSM will transition to WARMDOWN phase
3. tsm_count=END_OF_*SEQ*_WU, where *SEQ*=TX or RX. TSM will transition to ON phase
4. Breakpoint. TSM stay in its current phase and hold its count

A transition from WARMUP to ON phase means TSM will hold its count at END_OF_*SEQ*_WU, where *SEQ*=TX or RX.

A transition from ON to WARMDOWN phase means TSM will cease its hold, and resume counting at END_OF_*SEQ*_WU+1, where *SEQ* = TX or RX. (See Section Sequence Termination).

A transition from WARMUP to WARMDOWN phase means the tsm_count will jump to END_OF_*SEQ*_WU+1 (where *SEQ* = TX or RX), and resume counting from there. (see Section Sequence Termination).

The 4 8-bit registers which control the duration of the WARMUP phase (END_OF_TX_WU[7:0] and END_OF_RX_WU[7:0]), and the duration of the WARMDOWN phase (END_OF_TX_WD[7:0] and END_OF_RX_WD[7:0]), reside in the END_OF_SEQ register.

#### 64.3.5.2.5 Sequence Termination

A sequence may be terminated during the WARMUP or ON phase. A sequence will be terminated when one of the following conditions is met:

1. Deassertion of the initiating event. TSM will transition to WARMDOWN phase
2. Abort. TSM will transition to WARMDOWN phase

Deassertion of the initiating event (*source*_tx_en or *source*_rx_en), will cause a transition to WARMDOWN phase, at which point the TSM counter will resume counting from END_OF_*SEQ*_WU+1 (where *SEQ*=TX or RX).

An abort is caused by a PLL unlock event. An unlock abort will cause a transition to WARMDOWN phase, at which point the TSM counter will resume counting from END_OF_*SEQ*_WU+1 (where *SEQ* = TX or RX). See Section TSM Aborting.

During the WARMDOWN phase, whether caused by a deassertion of the initiating event, or a PLL unlock, subsequent aborts (PLL unlock events) will not be recognized by the TSM.

#### 64.3.5.2.6 Overrides

Each of the 67 TSM-controlled outputs has independent override control, except for the 4 GPIO triggers, and the SAR_ADC_TRIG output. In addition, two special TSM outputs, tx_mode and rx_mode, also have their own override controls, for a total of 67 – 5 + 2 = 64 overrides.

Two register bits are assigned to each output, to implement the override function. The bits are named:

*OUTPUTNAME*_OVRD_EN
*OUTPUTNAME*_OVRD

Setting the *OUTPUTNAME*_OVRD_EN=1, allows the *OUTPUTNAME*_OVRD bit to directly control the state of the output. Clearing *OUTPUTNAME*_OVRD_EN=0 returns control of the output to the TSM.

The 64 override bit-pairs, reside in the TSM_OVRD0, TSM_OVRD1, TSM_OVRD2, and TSM_OVRD3 registers.

Overrides have no effect on TSM operation, TSM counting, sequence initiation, or sequence termination. Overrides may be engaged at any time, during any TSM phase, including during idle state.

#### 64.3.5.2.7 Breakpoint

Breakpoint can be used to temporarily suspend a TSM sequence. A breakpoint can be set during the WARMUP or WARMDOWN phase of any TSM sequence. An 8-bit BKPT[7:0] register field resides in the TSM_CTRL register. During a TSM sequence, when the TSM counter matches the BKPT register value, counting stops and the counter holds in its current state. Aborts (PLL unlocks) will be ignored while holding at a breakpoint. Deassertion of the initiating event will also be ignored during the breakpoint. The breakpoint can be lifted by modifying the BKPT[7:0] register. Once the breakpoint is lifted, the TSM proceeds in the phase it was in prior to the breakpoint match, and the TSM counter begins incrementing again from the point at which the breakpoint occurred. Aborts (PLL unlocks) and deassertion of the initiaing event, will then be recognized and handled as per the description in the “Sequence Termination” section. The default value for BKPT[7:0] is 255, which is greater than the length of any allowed sequence, so a breakpoint will not trigger during a sequence, unless a lower value is programmed.

#### 64.3.5.2.8 TSM Aborting

The TSM can be enabled to abort any sequence on a PLL unlock condition. There are 3 mechanisms available for detecting PLL unlock. These are:

1. Coarse Tune Unlock
2. Frequency Target Unlock
3. Cycle Slip Unlock

Any of these mechanisms, or any combination of them, can be enabled to cause a TSM abort. Each of these mechanisms has a “_FAIL_FLAG” flag in the PLL_LOCK_DETECT register. On an unlock detection by any of these mechanisms, the corresponding “_FAIL_FLAG” flag (sticky bit) will become set, regardless of whether the mechanism is enabled to abort the TSM sequence. The “_FAIL_FLAG” flags are type “write-1-to-clear”.

An overview of the 3 PLL unlock mechanism is shown below

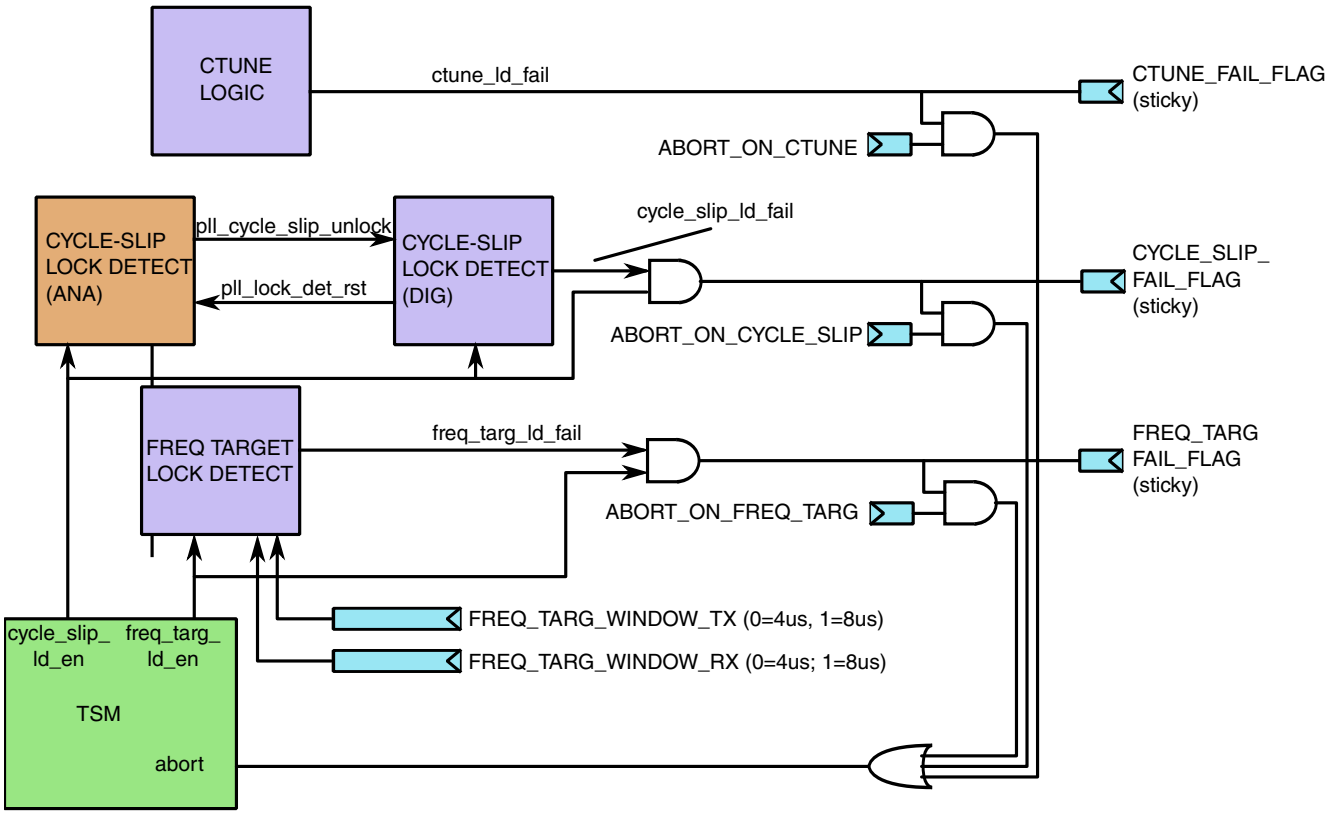



**Figure 64-63. PLL UNLOCK MONITORING AND ABORT LOGIC**

Each of the 3 PLL unlock mechanisms has an “ABORT_ON_” bit, to enable or disable that mechanism from aborting any TSM sequence. Any combination of “ABORT_ON_” bits can be configured.

In addition, the TSM directly controls the timing for 2 of the 3 unlock detect mechanisms: the Cycle Slip Lock Detect and the Frequency Target Lock Detect. (The timing for Coarse Tune Lock Detect mechanism is indirectly controlled by TSM via the pll_dig_en output; the PLL digital state machine is triggered by this signal and directly controls Coarse Tune). Enabling a lock detection mechanism via TSM programming allows an unlock condition to set the respective “_FAIL_FLAG” flag bit; To allow an unlock condition to cause an abort requires the respective “ABORT_ON_” bit to be set.

For each PLL unlock detect mechanism, the table below summarizes the TSM controlling signal for the mechanism, the timing register associated with that signal, and the default timing for the assertion of the controlling signal for both TX and RX sequences.

| Lock Detect Mechanism | TSM Controlling Signal | Associated TSM Timing Register | Controlling Signal Assertion Time(TX) | Controlling Signal Assertion Time(RX) |
|---|---|---|---|---|
| Coarse Tune | pll_dig_en (indirect) | TSM_TIMING34 | 7us | 7us |
| Freq. Target | freq_targ_ld_en | TSM_TIMING14 | 99us | 49us |
| Cycle Slip | pll_cycle_slip_ld_en | TSM_TIMING14 | 99us | 49us |

For each PLL unlock detect mechanism, the table below indicates the TSM controlling signal, the name of the "_FAIL_FLAG" flag bit, and the name of the "ABORT_ON_" enable bit to allow TSM aborting:

| Lock Detect Mechanism | TSM Controlling Signal | Lock Detect Fail Flag (PLL_LOCK_DETECT register) | Abort Enable Bit (TSM_CTRL register) |
|---|---|---|---|
| Coarse Tune | pll_dig_en (indirect) | CTUNE_FAIL_FLAG | ABORT_ON_CTUNE |
| Freq. Target | freq_targ_ld_en | FREQ_TARG_FAIL_FLAG | ABORT_ON_FREQ_TARG |
| Cycle Slip | pll_cycle_slip_ld_en | CYCLE_SLIP_FAIL_FLAG | ABORT_ON_CYCLE_SLIP |

All "_FAIL_FLAG" flag bits all reside in the PLL_LOCK_DETECT register; All "ABORT_ON_" bits reside in the TSM_CTRL register.

In addition to the controls listed above associated with each unlock source, there are 2 controls to allow TSM aborting to be disabled based on sequence type (TX or RX), instead of unlock source. When TX_ABORT_DIS is set to 1, TX sequences cannot be aborted, regardless of the state of the "ABORT_ON_" bits or any unlock condition. Likewise, when RX_ABORT_DIS is set to 1, RX sequences cannot be aborted, regardless of the state of the "ABORT_ON_" bits or any unlock condition. The TX_ABORT_DIS and RX_ABORT_DIS bits reside in the TSM_CTRL register. The TX_ABORT_DIS and RX_ABORT_DIS bits have no effect on the setting of the "_FAIL_FLAG" bits on any unlock condition; the TX_ABORT_DIS and RX_ABORT_DIS only affect aborting.

### 64.3.5.2.9 Special Handling for 802.15.4

Any of the 3 unlock detect mechanisms can be used to abort a 802.15.4 autosequence. However the method for aborting a 802.15.4 autosequence is different than a non-802.15.4 abort. For 802.15.4, the unlock condition does not abort the sequence directly. There is a higher-level 802.15.4 Sequence Manager (ZSM), which handles unlock aborts. For 802.15.4, the unlock condition is blocked from aborting the TSM as shown in the following diagram

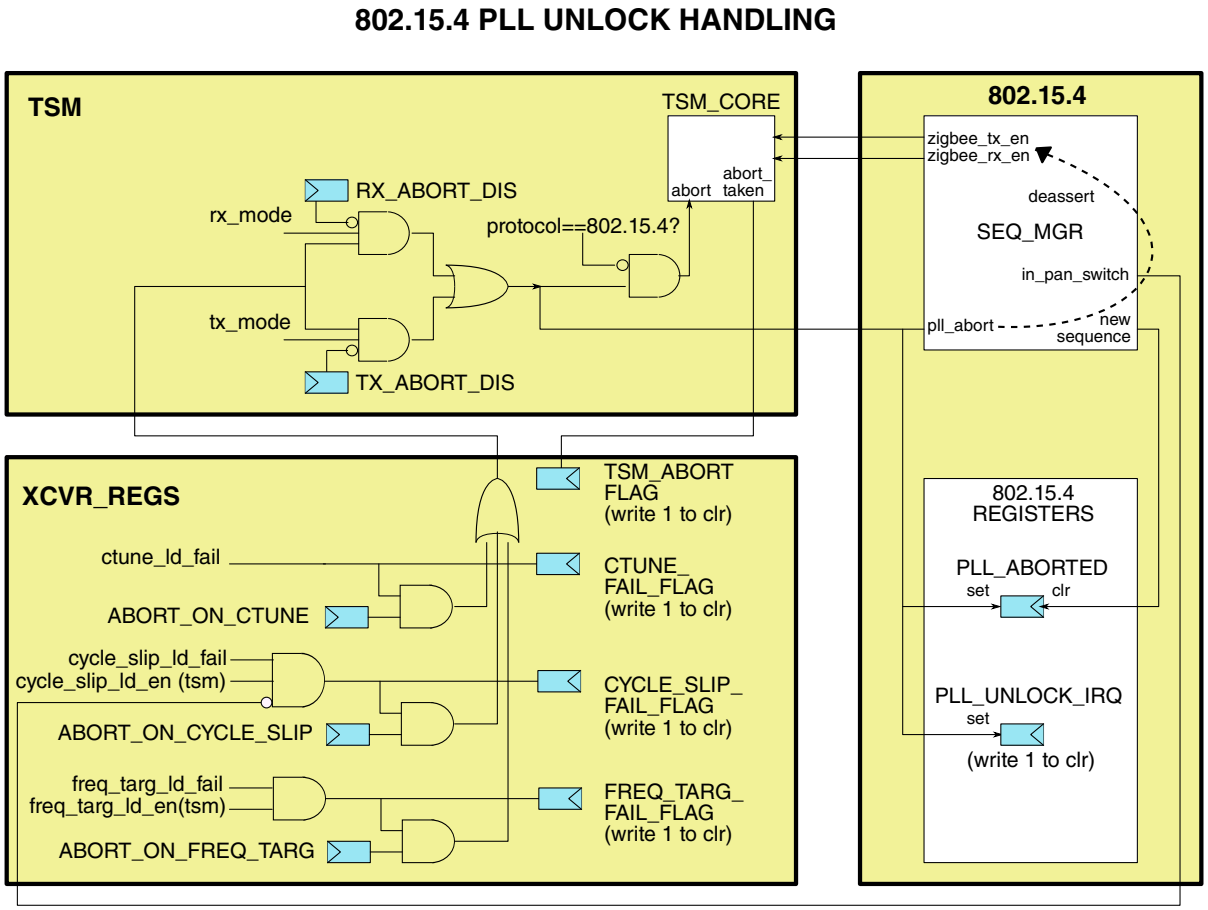



**Figure 64-64. 802.15.4 PLL UNLOCK HANDLING**

The combined-abort signal is blocked from reaching the TSM, and instead is routed to the ZSM. This is due to the fact that the ZSM already includes abort-handling logic, which is being re-used from previous NXP 802.15.4 transceivers. Keeping the unlock handling intact, facilitates the porting of software from one NXP 802.15.4 product to the next.

For 802.15.4, the logic which determines which ZSM states allow aborting, is designed into the ZSM state machine. Not all states require an abort-on-unlock. When an unlock event occurs during a ZSM state that requires an abort, the ZSM will deassert its initiating signal to TSM (zigbee_tx_en or zigbee_rx_en), which results in a de facto abort, similar to a TSM-generated abort. For a ZSM abort, the PLL_UNLOCK_IRQ bit will become set, as will the PLL_ABORTED bit. The PLL_UNLOCK_IRQ bit is of type write-1-to-clear, and resides in the IRQSTS1 register of 802.15.4 space. The PLL_ABORTED bit is a read-only bit, and resides in the ABORT_STS register of 802.15.4 space; it will self-clear at the start of the next autosequence.

For 802.15.4, the TX_ABORT_DIS and RX_ABORT_DIS control bits are still available, and allow/disallow aborting during the TX portions of any autosequence (TX_ABORT_DIS), or the RX portions of any autosequences (RX_ABORT_DIS). The

802.15.4 autosequences are concatenations of TX and RX operations, under the control of the ZSM. The ZSM engages the TSM when needed, to perform a TX or RX warmup or warmdown, by asserting or deasserting zigbee_tx_en or zigbee_rx_en.

See the 802.15.4 Chapter, Section Sequence Manager, sub-section ZSM/TSM Interaction.

As mentioned previously, for 802.15.4 the PLL_UNLOCK_IRQ will become set to indicate a PLL unlock event has caused an 802.15.4 autosequence to be aborted. When 802.15.4 is not in use, another bit is provided to indicate a non-802.15.4 unlock abort (a TSM abort) has occurred. This bit is TSM_ABORT_FLAG. This bit is of type write-1-to-clear, and resides in the PLL_LOCK_DETECT register of transceiver space. This bit will not be set for an 802.15.4 abort (use PLL_UNLOCK_FLAG or PLL_ABORTED instead).

#### 64.3.5.2.10 Special Handling for GENERIC_FSK

The GENERIC_FSK Link Layer Controller already has built-in handling for PLL unlock conditions, and an interrupt status bit to notify on such an event. The combined-abort signal is blocked from reaching the TSM, and instead is routed to the GENERIC_FSK Command Decoder. The Command Decoder state machine handles the unlock event by deasserting the TX or RX command signal to the TSM (e.g., generic_fsk_tx_en), causing a TSM warmdown to occur. The Generic controller will also set the PLL_UNLOCK_IRQ interrupt status bit in its address space. The diagram below shows the GENERIC_FSK Link Layer handling of the PLL unlock event.

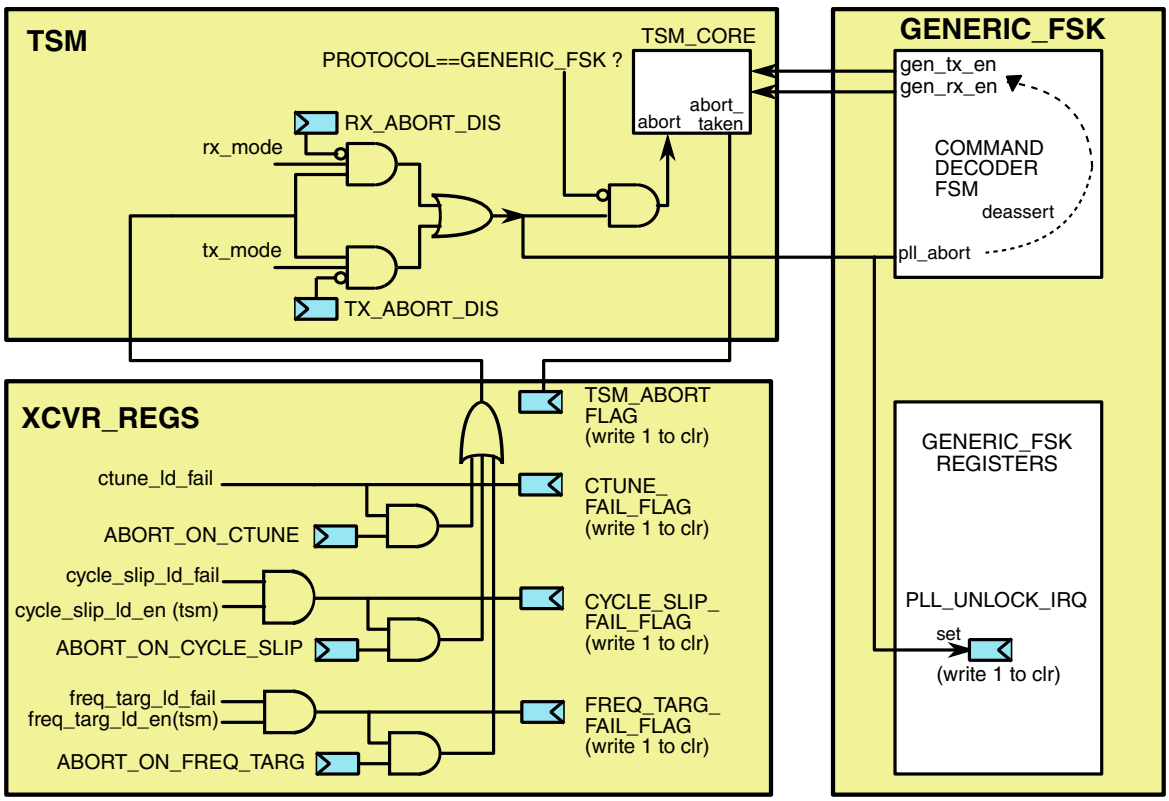



**Figure 64-65. GENERIC_FSK PLL UNLOCK HANDLING**

For GENERIC_FSK, the TX_ABORT_DIS and RX_ABORT_DIS control bits are still available, and selectively allow/disallow aborting during the TX sequences (TX_ABORT_DIS), or the RX sequences (RX_ABORT_DIS).

### 64.3.5.2.11 PA Target Power

The TSM controls the power level that is sent to the PA. Target power level is the power level that is used during packet transmission. Nominally, power level is constant over the course of the packet.

During a packet transmission sequence, prior to the start of preamble, while the PA is being turned on, power level must be ramped up gradually, to prevent unwanted spectral effects. After the packet is complete, and the PA is being turned off, power level must be ramped down slowly for the same reason.

The TSM controls power level to the PA during these 3 phases, as shown below:

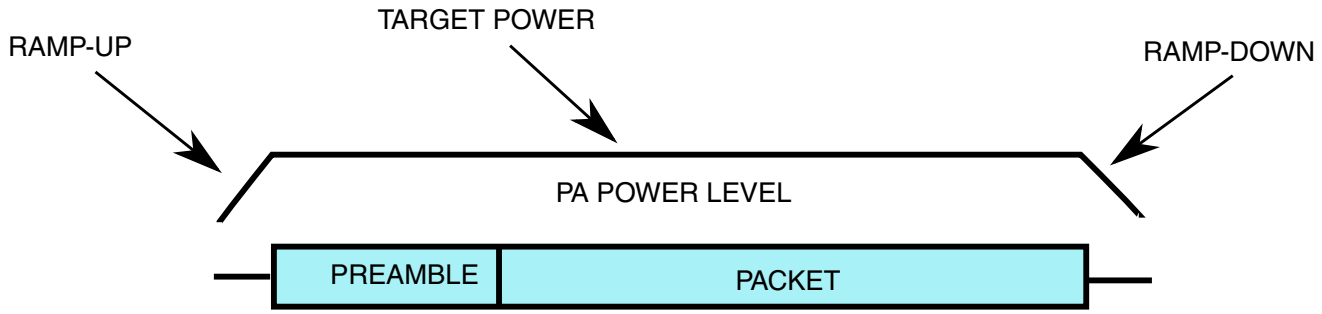



**Figure 64-66. PA Target Power**

Target power, the steady-state power level that is used to transmit the packet, can be selected from one of the following sources:

1. PA_POWER[5:0] register in XCVR address space
2. BTLE Link Layer (2 distinct power levels supported, for Advertising- and Data-channel Packets)
3. 802.15.4 Link Layer (PA_PWR[5:0] register in 802.15.4 address space)
4. GENERIC_FSK Link Layer (TX_POWER[5:0] register in GENERIC_FSK address space)

The following diagram depicts target power selection.

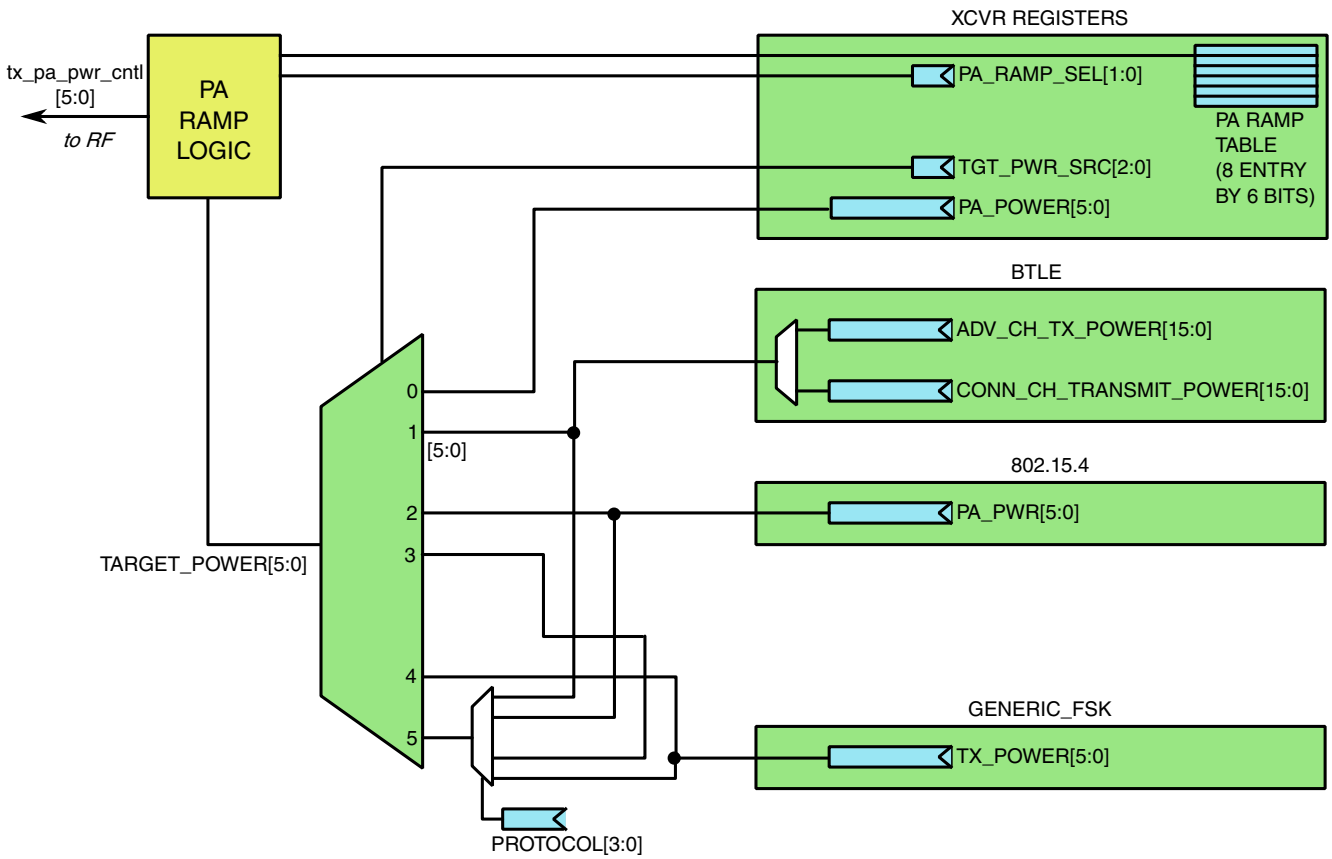



**Figure 64-67. PA Target Power Selection**

The XCVR register TGT_PWR_SRC[2:0], controls target power selection, according to the following table.

| TGT_PWR_SRC[2:0] | TARGET POWER SOURCE |
|---|---|
| 000 | PA_POWER[5:0] register (XCVR space) |
| 001 | BTLE Link Layer |
| 010 | 802.15.4 Link Layer (PA_PWR[5:0] register in 802.15.4 space) |
| 011 | Reserved |
| 100 | GENERIC_FSK Link Layer (TX_POWER[5:0] register in GENERIC space) |
| 101 | Reserved |
| 110 | Reserved |
| 111 | PROTOCOL[3:0] bits select target power source |

These bits reside in the XCVR_CTRL register in XCVR address space.

See Section “BTLE Radio Operation” for a description of how the BTLE Link Layer controls target power.

The TSM provides power control information to the PA as a 6-bit output: tx_pa_pwr_ctrl[5:0].

### 64.3.5.2.12 PA Ramping

PA ramping is included in the TSM, to prevent abrupt transitions in PA (power amplifier) power, that could cause unwanted spectral transients. During a TX sequence, PA ramping gradually ramps up PA power, between a programmable minimum, and the TX target power. The trajectory of the ramp is programmable, and ramping takes place over the course of 1, 2, or 4µs (programmable).

During a TX sequence, PA ramping, if enabled, is triggered by a low-to-high transition on the TSM output **tx_pa_en**, which is also the enable for the PA (tx buffer). At assertion of **tx_pa_en** during the TX sequence with ramping enabled, the TSM will enable the PA, with a minimum power level determined by the register PA_RAMP0[5:0]. After that, power level will increment according to a 16-deep pre-programmed register table (PA_RAMP Table). Incrementing will occur at a programmable rate. At each ramp step, PA power level will be taken from the next entry in the PA_RAMP Table, from 0-15. After the 16th and final ramp step, PA power level will be the selected target power. The target power will remain in effect for the duration of the packet transmission, i.e., the "ON" phase of the TSM seqeuence.

The ramp trajectory is determined by the contents of the PA_RAMP Table. The PA_RAMP table contains register fields PA_RAMP0[5:0] … PA_RAMP15[5:0]. These register fields can be found in the PA_RAMP_TBL0, PA_RAMP_TBL1, PA_RAMP_TBL2, and PA_RAMP_TBL3 registers in XCVR address space. The default contents of the PA_RAMP table are as shown:

| PA_RAMP Table Entry | POWER CONTROL VALUE (DEFAULT) |
|---|---|
| PA_RAMP0[5:0] | 0x01 |
| PA_RAMP1[5:0] | 0x02 |
| PA_RAMP2[5:0] | 0x04 |
| PA_RAMP3[5:0] | 0x06 |
| PA_RAMP4[5:0] | 0x08 |
| PA_RAMP5[5:0] | 0x0C |
| PA_RAMP6[5:0] | 0x10 |
| PA_RAMP7[5:0] | 0x14 |
| PA_RAMP8[5:0] | 0x18 |
| PA_RAMP9[5:0] | 0x1C |
| PA_RAMP10[5:0] | 0x22 |
| PA_RAMP11[5:0] | 0x28 |

*Table continues on the next page...*

| PA_RAMP Table Entry | POWER CONTROL VALUE (DEFAULT) |
|---|---|
| PA_RAMP12[5:0] | 0x2C |
| PA_RAMP13[5:0] | 0x30 |
| PA_RAMP14[5:0] | 0x36 |
| PA_RAMP15[5:0] | 0x3C |

At each step of the ramp-up, the selected entry of the PA_RAMP Table is compared against target_power. The lesser of the two becomes the output power for that ramp step. This allows the same PA_RAMP table to be used for various target_power levels, without re-programming the table.

The ramp rate is controlled by the PA_RAMP_SEL[1:0] field of the TSM_CTRL register in XCVR address space. For each setting, the table below describes the total ramp duration, and the duration of each ramp step.

| PA_RAMP_SEL[1:0] | TOTAL RAMP DURATION (32MHZ) | DURATION PER RAMP STEP (32MHZ) | TOTAL RAMP DURATION (26MHZ) | DURATION PER RAMP STEP (26MHZ) |
|---|---|---|---|---|
| 00 | No ramp | No ramp | No ramp | No ramp |
| 01 | 1.0µs | 2 Reference Clocks (0.0625µs) | 1.23µs | 2 Reference Clocks (0.077µs) |
| 10 | 2.0µs | 4 Reference Clocks (0.125µs) | 2.46µs | 4 Reference Clocks (0.154µs) |
| 11 | 4.0µs | 8 Reference Clocks (0.25µs) | 4.92µs | 8 Reference Clocks (0.308µs) |

When PA_RAMP_SEL[1:0] is set to 0, there is no ramping, and tx_pwr_ctrl[5:0] tracks the selected Target Power at all times (see Section PA Target Power).

When PA_RAMP_SEL[1:0] is set to 1, a 1µs ramp is generated, as shown:

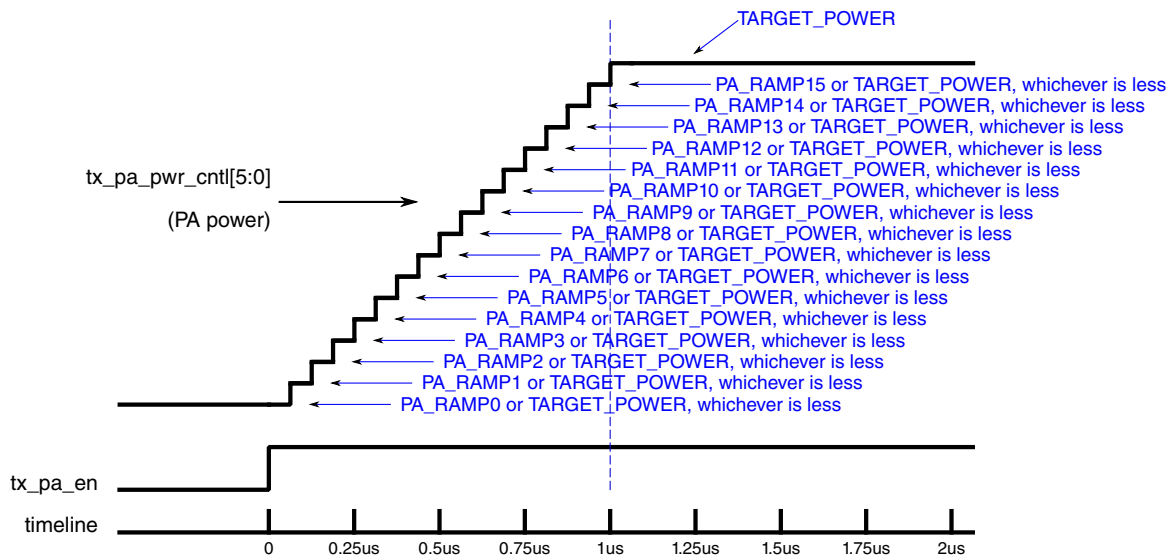



**Figure 64-68. 1µS PA RAMP UP (32MHZ TIMING)**

When PA_RAMP_SEL[1:0] is set to 2, a 2µs ramp is generated, as shown:

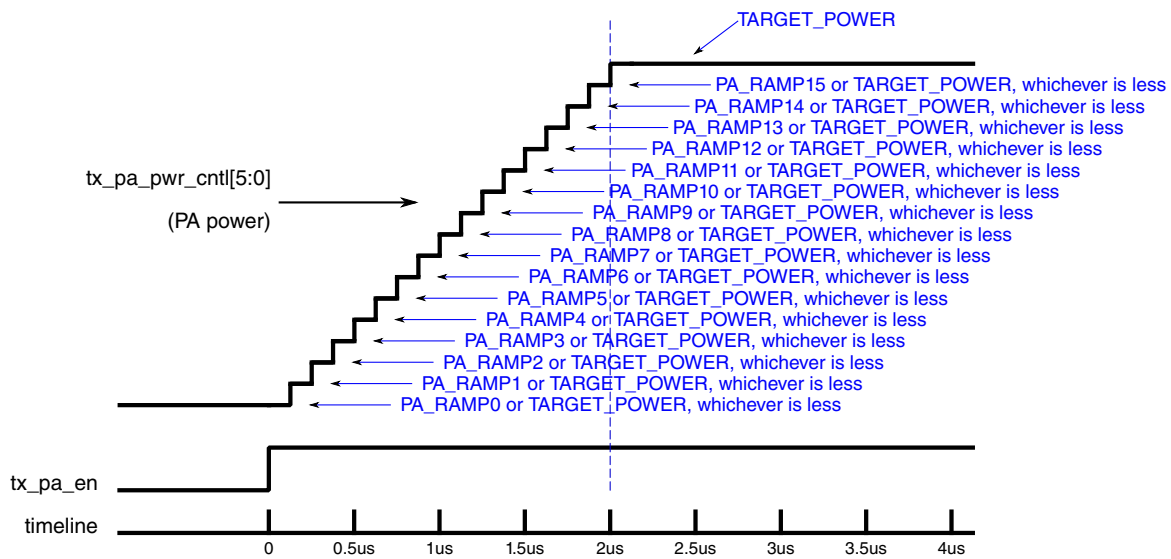



**Figure 64-69. 2µS PA RAMP UP (32MHZ TIMING)**

When PA_RAMP_SEL[1:0] is set to 3, a 4µs ramp is generated, as shown:

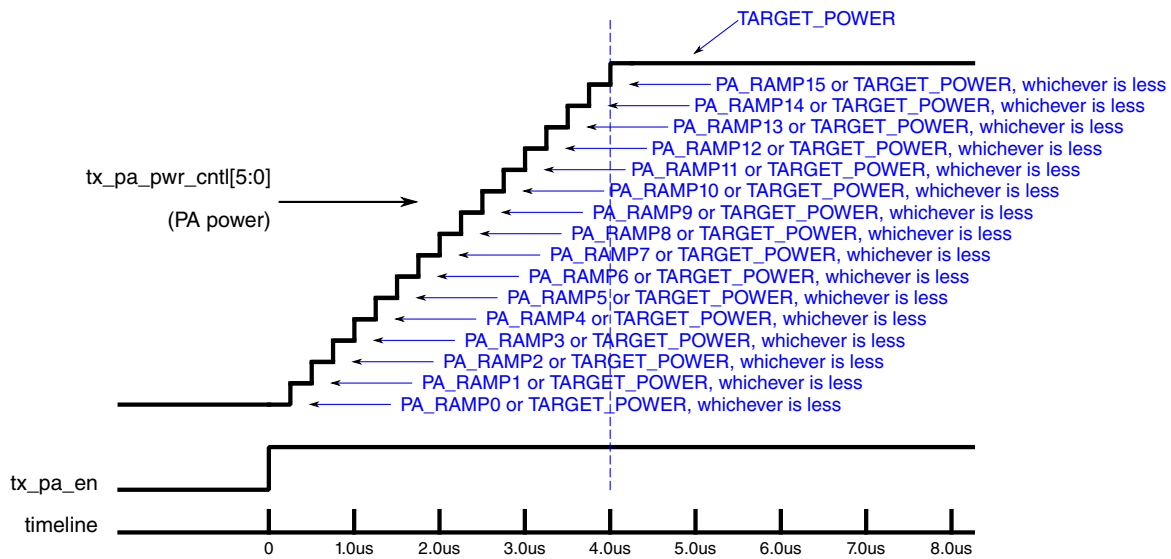



**Figure 64-70. 4µS PA RAMP UP (32MHZ TIMING)**

PA ramp-down is the mirror image of ramp-up. Ramp-down is triggered by the deassertion of the sequence-initiating event (e.g., btle_tx_en, zigbee_tx_en, or force_tx_en). At deassertion of the initiating event, output power will continue to hold at target_power for a programmable duration, controlled by the register TSM_CTRL[RAMP_DN_DLY]. This 4-bit register accommodates filter, and other, delay in the TX path before beginning the ramp down. RAMP_DN_DLY[3:0] is in units of µs. The default for RAMP_DN_DLY is 4. The effect of RAMP_DN_DLY, can be viewed as extending the "ON" phase of the TSM by a programmable number of microseconds. After RAMP_DN_DELAY expires, output power will step through the PA_RAMP Table in reverse order, starting at PA_RAMP15, and ending at PA_RAMP0. At each step of the ramp-down, the selected entry of the PA_RAMP Table is compared against target_power. The lesser of the two becomes the output power. The PA_RAMP_SEL[1:0] register controls the ramp-down rate in the same manner that it controls the ramp-up rate. After the last step of the ramp-down (PA_RAMP0 level), **tx_pa_en** deasserts to disable the PA.

The following diagram shows an example of PA ramp down. In this example, PA_RAMP_SEL=2 (2µs), and RAMP_DN_DLY=3 (3µs):

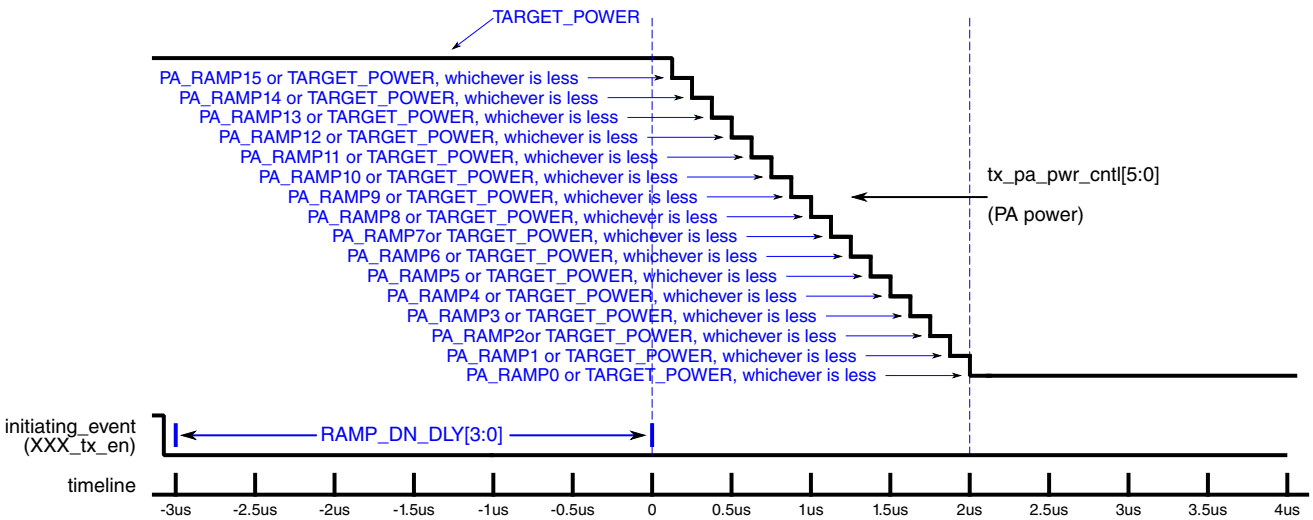



**Figure 64-71. PA RAMP DOWN (32MHZ TIMING)**

### 64.3.5.2.13 Fast Antenna Diversity (FAD)

#### FAD INTRODUCTION

The Fast Antenna Diversity (FAD) mode, when enabled, will allow the optimal choice between two antennae to be selected during the preamble search. The 2.4GHz Gen3 radio provides 2 alternatives for FAD:

1. FAD for 802.15.4
2. FAD for FSK

The 2 FAD options are described in the sections that follow.

#### 1. FAD for 802.15.4

See Chapter *Multi-standard PHY*, Section *FAD process in IEEE 802.15.4* for a full description of this FAD option.

The output of the 802.15.4 FAD algorithm, is a 1-bit signal, antx_out_zig, which indicates the selected antenna. This output is provided to the FAD Common Pin Interface logic.

#### 2. FAD for FSK

See Chapter *Multi-standard PHY*, Section *FAD process in FSK* for a full description of this FAD option.

The output of the FSK FAD algorithm, is a 1-bit signal, antx_out_fsk, which indicates the selected antenna. This output is provided to the FAD Common Pin Interface logic.

## FAD COMMON PIN INTERFACE

The FAD Common Pin Interface consists of 4 pins: ANT_A, ANT_B, TX_SWITCH, and RX_SWITCH. All pins are outputs from the device. The switch outputs convey timing information to the external module directly interfacing to the antennae, and they also can be used to control an external LNA or PA, via use of a Front End Module, or FEM. The FAD Common Pin Interface offers multiple configurations that allow for a wide range of FEM control schemes. Each of the 4 FAD-related outputs can be configured to be in FAD or TSM mode; in FAD mode, ANT_A and ANT_B are controlled by the selected FAD algorithm through signal antx_out_zig (802.15.4) or antx_out_fsk (FSK); in TSM mode, ANT_A and ANT_B are pure timing outputs with programmable timing configurable in the TSM. The 4-bit register FAD_CTRL[FAD_NOT_GPIO] determines whether each FAD-related output is in FAD or TSM mode:

**NOTE**

When FAD is enabled (FAD_EN=1), programming of the appropriate TSM_TIMING registers for TX_SWITCH and RX_SWITCH is *mandatory*, regardless of which FAD-pin Control Mode is selected, and even if the TX_SWITCH and RX_SWITCH pins are not used by the external module.

In FAD mode (FAD_NOT_GPIO[0]=1), ANT_A is controlled by the FAD algorithm, modified by the register settings {ANTX_CTRLMODE, ANTX_EN, and ANTX_POL}.

In FAD mode (FAD_NOT_GPIO[1]=1), ANT_B is controlled by the FAD algorithm, modified by the register settings {ANTX_CTRLMODE, ANTX_EN, and ANTX_POL}.

In FAD mode (FAD_NOT_GPIO[2]=1), TX_SWITCH output is derived from TSM output **gpio2_trig_en**, modified by the register settings {ANTX_CTRLMODE, ANTX_EN, and ANTX_POL}. Program TSM_TIMING49 register to obtain the desired timing on the TX_SWITCH pin.

In FAD mode (FAD_NOT_GPIO[3]=1), RX_SWITCH output is derived from TSM output **gpio3_trig_en**, modified by the register settings {ANTX_CTRLMODE, ANTX_EN, and ANTX_POL}. Program TSM_TIMING50 register to obtain the desired timing on the RX_SWITCH pin.

In TSM mode, ANT_A and ANT_B become pure TSM timing-controlled outputs, similar to TX_SWITCH and RX_SWITCH, and with independently-controlled timing and capable of providing signalling in TX mode, RX mode, or both; for pins in TSM mode, the ANTX_CTRLMODE, ANTX_EN, and ANTX_POL register settings have no effect.

In TSM mode (FAD_NOT_GPIO[0]=0), ANT_A output is a pure timing signal, derived from TSM output **gpio0_trig_en**. Thus, program TSM_TIMING47 register to obtain the desired timing on the ANT_A pin.

In TSM mode (FAD_NOT_GPIO[1]=0), ANT_B output is a pure timing signal, derived from TSM output **gpio1_trig_en**. Thus, program TSM_TIMING48 register to obtain the desired timing on the ANT_B pin.

In TSM mode (FAD_NOT_GPIO[2]=0), TX_SWITCH output is derived from TSM output **gpio2_trig_en**. Program TSM_TIMING49 register to obtain the desired timing on the TX_SWITCH pin.

In TSM mode (FAD_NOT_GPIO[3]=0), RX_SWITCH output is derived from TSM output **gpio3_trig_en**. Program TSM_TIMING50 register to obtain the desired timing on the RX_SWITCH pin.

The control register XCVR_CTRL[DEMOD_SEL] determines which FAD algorithm drives the FAD Common Pin Interface: 0b10 = 802.15.4 FAD, 0b01 = FSK FAD.

The state of the antenna selection can be ascertained at any time by reading the FAD_CTRL[ANTX] bit: 0 = ANT_A, 1 = ANT_B.

Antenna selection can be placed under direct software control: Setting the FAD_CTRL[ANTX_OVRD_EN]=1 allows FAD_CTRL[ANTX_OVRD] to substitute for the algorithm-driven antenna selection; in this case, ANTX_OVRD=0 asserts ANT_A, and ANTX_OVRD=1 asserts ANT_B. Note that the algorithm-driven state of the antenna is not lost while ANTX_OVRD_EN=1. The override is temporary, for as long as the override enable is set, and control over antenna state reverts to the selected FAD algorithm (antx_out_zig or antx_out_fsk) when ANTX_OVRD_EN is cleared to 0. The FAD_CTRL register resides in XCVR space.

The following diagram illustrates the FAD pinout, register controls, and FAD/TSM-mode multiplexing.

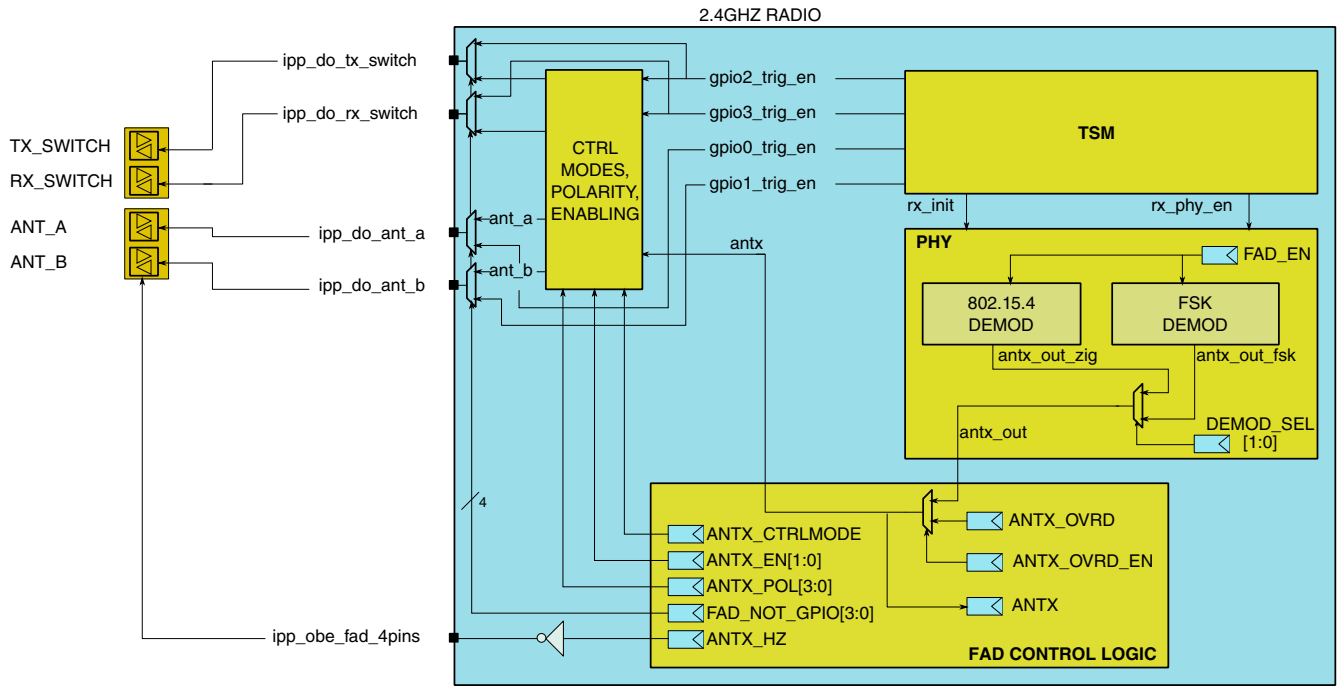



**Figure 64-72. FAD Common Pin Interface**

## FAD REGISTERS

The following table describes the programmable registers which control FAD:

**Table 64-37. FAD Registers**

| Field | R/W | Description |
|---|---|---|
| ANTX_CTRLMODE | rw | **FAD control mode-** selects the single or dual mode<br>**0 : single mode.**<br>ANT_A = antx_out<br>TXSWITCH = gpio2_trig_en<br>RXSWITCH= (gpio2_trig_en OR gpio3_trig_en)<br>**1:dual mode:**<br>ANTA = antx_out AND (gpio2_trig_en OR gpio3_trig_en)<br>ANTB = NOT(antx_out) AND (gpio2_trig_en OR gpio3_trig_en)<br>TX_SWITCH = gpio2_trig_en<br>RX_SWITCH = gpio3_trig_en |
| ANTX_EN[1:0] | rw | **FAD Pin Enable-**<br>00: all disabled<br>01:only RX_SWITCH/TX_SWITCH enabled |

*Table continues on the next page...*

## Table 64-37. FAD Registers (continued)

| Field | R/W | Description |
|---|---|---|
| | | 10:only ANT_A/ANT_B enabled<br><br>11: all enabled |
| ANTX_POL[3:0] | rw | **Antenna controls mode-** Control the polarity of the FAD pins:<br><br>ANTX_pol<0>=1 : invert the ANT_A output<br><br>ANTX_pol<1>=1 : invert the ANT_B output<br><br>ANTX_pol<2>=1 : invert the TX_SWITCH output<br><br>ANTX_pol<3>=1 : invert the RX_SWITCH output |
| ANTX_HZ | rw | **FAD controls high impedance**- Set ANTA,ANTB,RXSWITCH and TXSWITCH in high impedance |
| FAD_NOT_GPIO[3:0] | rw | **xxx1**: The ANT_A pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br><br>**xxx0**: The ANT_A pad is controlled directly by the TSM output gpio0_trig_en<br><br>**xx1x**: The ANT_B pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br><br>**xx0x**: The ANT_B pad is controlled directly by the TSM output gpio1_trig_en<br><br>**x1xx**: The TX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br><br>**x0xx**: The TX_SWITCH pad is controlled directly by the TSM output gpio2_trig_en<br><br>**1xxx**: The RX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br><br>**0xxx**: The RX_SWITCH pad is controlled directly by the TSM output gpio3_trig_en |
| ANTX | ro | **Antenna Selection State-**<br><br>This read-only register reflects the current state of the antenna selection: 0 = ANT_A, 1 = ANT_B. |
| DEMOD_SEL[1:0] | rw | **These bits select the demodulator used during reception. Also select which FAD engine is used to select the antenna-**<br><br>• 0b00: No demodulator selected<br>• 0b01: Use NXP Constant Envelope demodulator, and the FSK FAD engine<br>• 0b10: Use Legacy 802.15.4 demodulator, and the 802.15.4 FAD engine<br>• 0b11: Reserved |
| ANTX_OVRD_EN | rw | **FAD ANTX Override Enable-** |

*Table continues on the next page...*

**Table 64-37. FAD Registers (continued)**

| Field | R/W | Description |
|---|---|---|
| | | • 0: Selected FAD engine controls the antenna selection (mission mode)<br>• 1: ANTX_OVRD controls the antenna selection. 0 = ANT_A, 1 = ANT_B. |
| ANTX_OVRD | rw | **FAD ANTX Override -**<br>This bit is activated when ANTX_OVRD_EN=1, as per the following:<br>• 0: ANT_A is selected (software control)<br>• 1: ANT_B is selected (software control) |

**Note:** All register bits and fields listed in the table above, reside in the FAD_CTRL register, except FAD_THR, which resides in a register of the same name. Both FAD_CTRL and FAD_THR registers reside in XCVR address space.

## FRONT END MODULE

In 2.4GHz systems, the Radio Frequency (RF) Front-End Module is one of the most critical parts. Acting as an interface between the antenna and RF transceiver, the RF front-end module, or FEM, includes sensitive components such as an antenna Low Noise Amplifier (LNA) , power amplifiers (PA), antenna tuning switches, and a power management unit.

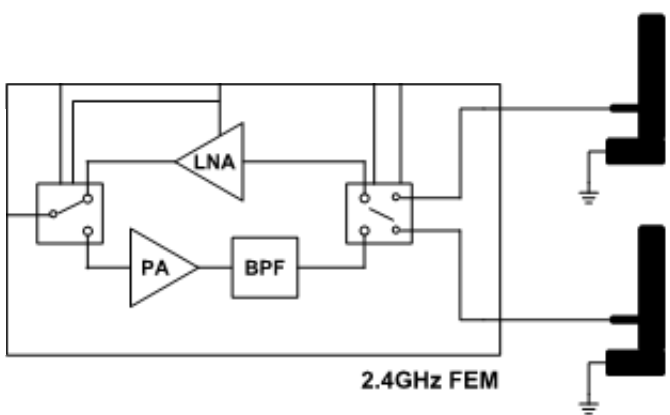



**Figure 64-73. FEM Block Diagram**

To accommodate multiple FEM interfaces and control schemes, Fast Antenna Diversity incorporates 2 Antenna Pin Control Modes, which are governed by the register control bit ANTX_CTRLMODE

Some FEM devices use a single input pin to switch the antenna selection. This is called single-mode operation. Operate Fast Antenna Diversity in this mode by programming ANTX_CTRLMODE=0. A timing diagram of single-mode operation is shown below (note: FAD output ANT_B is not used). In this example, one packet is received on Antenna A, followed by another packet received on Antenna B. Both packets require an auto-Acknowledge packet to be transmitted immmediately following reception.

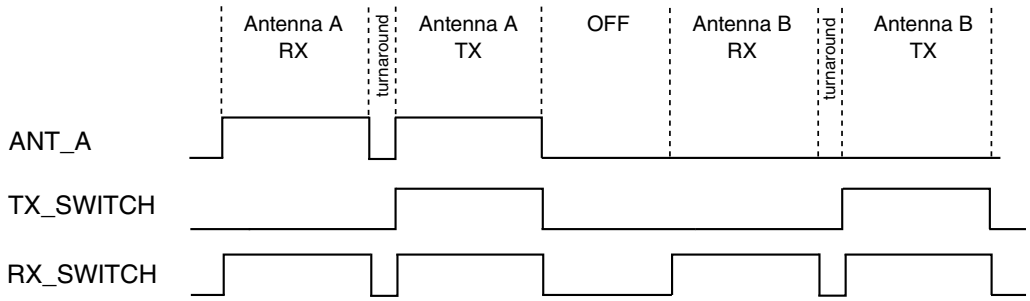



**Figure 64-74. FAD Single Mode Timing**

An example FEM Interface diagram, with FAD operated in single mode, is shown below.

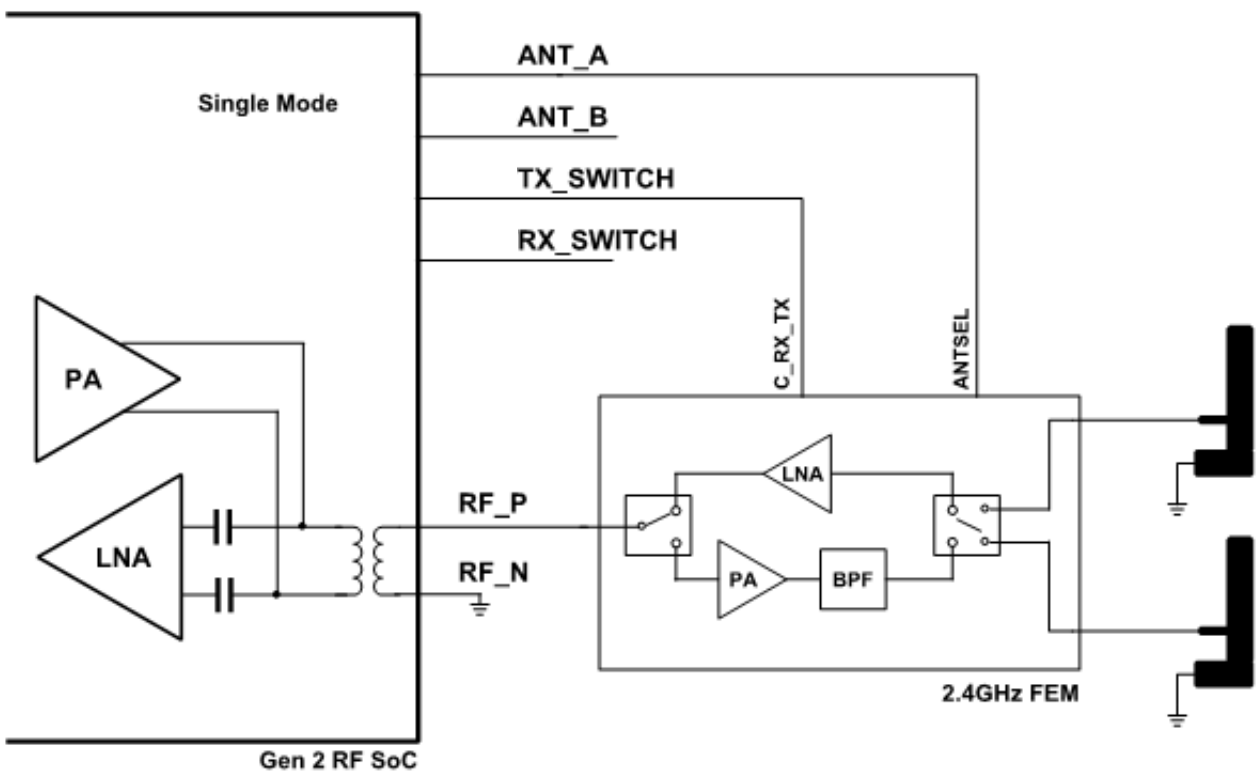



**Figure 64-75. FEM Interface Single Mode**

Other FEM devices use 2 input pins to switch the antenna selection. This is called dual-mode operation. Operate Fast Antenna Diversity in this mode by programming ANTX_CTRLMODE=1. A timing diagram of dual-mode operation is shown below. In this example, one packet is received on Antenna A, followed by another packet received on Antenna B. Both packets require an auto-Acknowledge packet to be transmitted immmediately following reception.

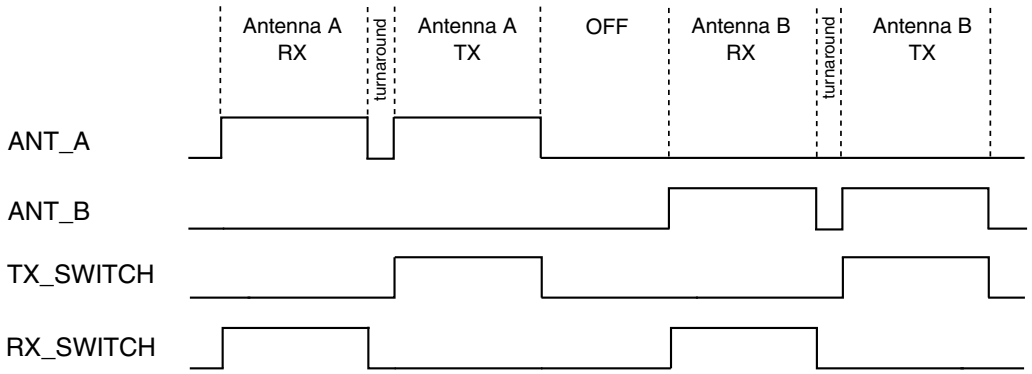



**Figure 64-76. FAD Dual Mode Timing**

An example FEM Interface diagram, with FAD operated in dual mode, is shown below.

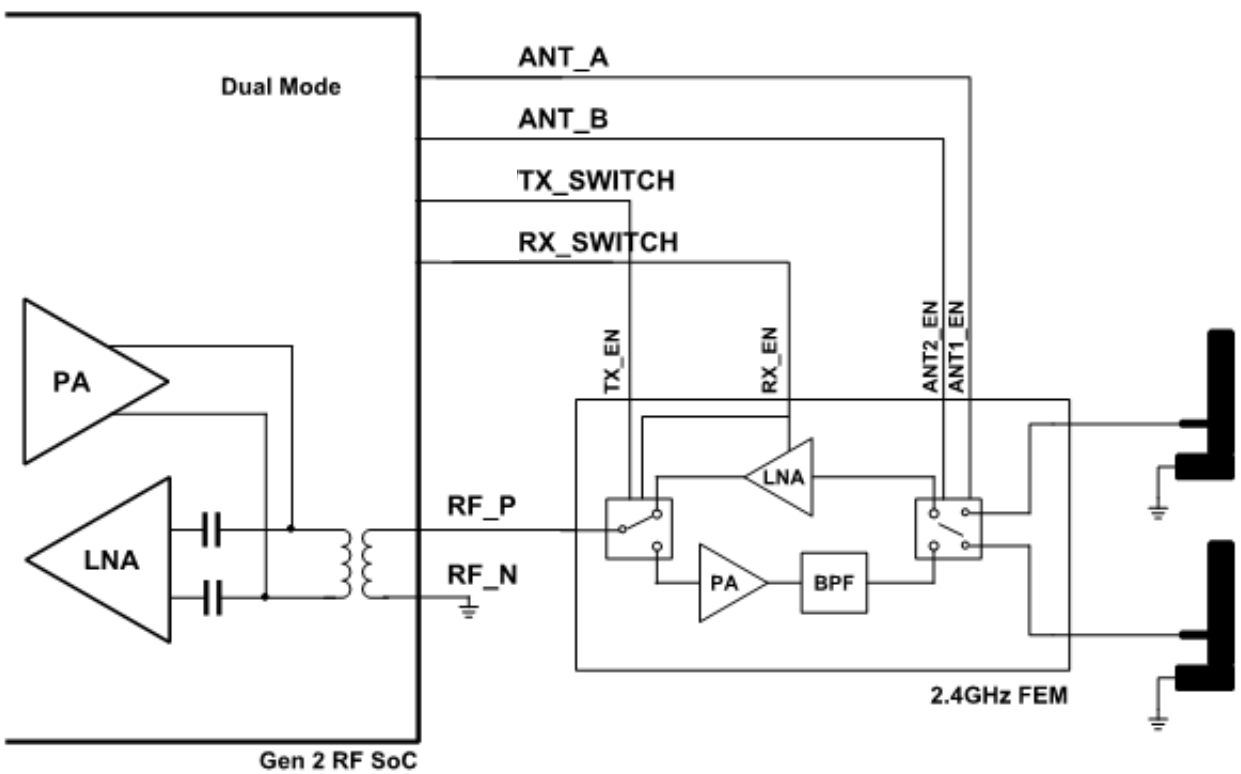



**Figure 64-77. FEM Interface Dual Mode**

Other FEM use cases, include external PA only, external LNA only, or external PA plus LNA. In such cases, ANT_A and ANT_B are not required, FAD can be disabled, and TX_SWITCH and RX_SWITCH pins can be operated in TSM mode (pure timing signals). This is called Single Antenna configuration. An example FEM interface diagram illustrating such a use case, is shown below.

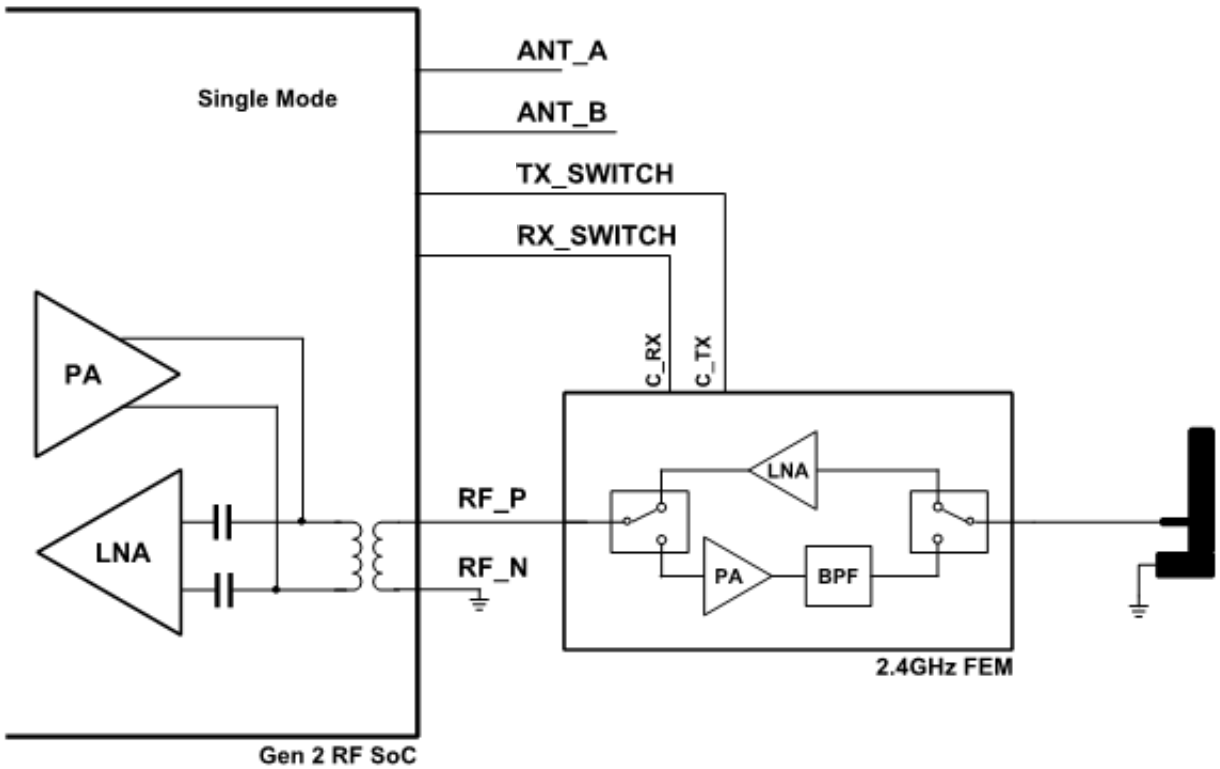



**Figure 64-78. FEM Interface Single Antenna Configuration**

### 64.3.5.2.14 Low Power Preamble Search (LPPS)

To reduce power consumption during 802.15.4 preamble-search receive state, a low-power-preamble-search (LPPS) mode can be optionally enabled. When engaged, the correlators are disabled approximately 50% of the time during preamble search. The same signal used to dynamically enable/disable the correlators, can be used to enable/disable selected analog and RF blocks in tandem, to further reduce power consumption. The LPPS_CTRL register in XCVR space, has one bit to master-enable LPPS mode (LPPS_ENABLE), and several other bits to allow selected analog and RF blocks to be "duty-cycled" in sync with the correlators. For more details on the LPPS algorithm, see Chapter Multi-standard PHY, Section *IEEE Low-power Preamble Search*.

By default, LPPS mode is not enabled.

During LPPS mode, correlators and RF blocks are duty-cycled with timing controlled by the LPPS signal **lpps_lp_en**, which is a single-wire output from the 802.15.4 demodulator. In general, eligible blocks will be powered off when **lpps_lp_en** is high, and perform normal operation when **lpps_lp_en** is low. Transitions on **lpps_lp_en** will be handled by the TSM in the following ways:

**lpps_lp_en 1 → 0:**

1. TSM outputs which are programmed to be eligible for LPPS duty-cycling (e.g, those associated with TZA, BBA, ADC) shall be reasserted at this point.
2. TSM shall recycle to a programmable point in the RX warmup. The TSM RECYCLE_COUNT2[7:0] register selects the TSM_COUNT to recycle to.
3. After jumping to the TSM_COUNT indicated in RECYCLE_COUNT2, TSM shall from this point forward execute a Fast RX Warmup (see Section FAST TSM Warmup and FAST RX2TX Transition later in this chapter). For the Fast RX Warmup, the registers LPPS_START_RX[7:0] and LPPS_DEST_RX[7:0] shall be used to program the TSM "jump" start and end points, as defined in the Fast RX Warmup section. The LPPS_START_RX and LPPS_DEST_RX registers reside in the LPPS_CTRL register. These registers are specific to LPPS fast warmups, and do not apply to non-LPPS fast warmups.

**lpps_lp_en 0 → 1:**

1. TSM outputs which are programmed to be eligible for LPPS duty-cycling (e.g, those associated with TZA, BBA, ADC) shall be deasserted at this point.
2. TSM outputs which are not eligible for LPPS duty-cycling shall remain asserted.

Note that several LPPS timing parameters are configurable (see Chapter Multi-standard PHY, Section *IEEE Low-power Preamble Search*), and LPPS_FILL_COUNT[6:0] is based on the LPPS fast warmup time. Software configuring LPPS parameters, such as LPPS_FILL_COUNT, must calculate the LPPS fast warmup time, in µs, based on the register programming described in the section above. Here is the equation:

**LPPS_FAST_WARMUP_DURATION = END_OF_RX_WU - RECYCLE_COUNT2 + 1 - (LPPS_DEST_RX - LPPS_START_RX - 1)**

*for example,*

**LPPS_FAST_WARMUP_DURATION = 104 - 28 + 1 - ( 102 - 40 - 1)**

**LPPS_FAST_WARMUP_DURATION = 16µs**

Note that the penalty for the reduced power consumption in LPPS mode during preamble search, is a slight degradation in receiver sensitivity. Note also that the TSM override bits, also override LPPS functionality. The override bits provide ultimate control over all TSM outputs.

Programming control for LPPS, includes a master-enable register bit for the LPPS state-machine, as well as individual control bits to enable selected analog and digital blocks to be turned on-and-off during LPPS operation. The following table lists the LPPS register bits, and the TSM outputs controlled by each.

**Table 64-38. LPPS Register Controls**

| LPPS Register Bit | Description |
|---|---|
| LPPS_ENABLE | Master enable for LPPS mode. Allows correlators to be duty-cycled during Preamble Search, and selected analog and digital blocks to be duty-cycled simultaneously. |
| LPPS_TZA_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_lna_en, rx_tza_i_en, rx_tza_q_en |
| LPPS_BBA_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_i_en, rx_bba_q_en |
| LPPS_ADC_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_adc_en, rx_adc_i_en, rx_adc_q_en |
| LPPS_DCOC_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_dcoc_en, rx_tza_dcoc_en |
| LPPS_PDET_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_pdet_en, rx_tza_pdet_en |
| LPPS_SY_LO_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_en |
| LPPS_SY_LO_BUF_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_buf_en |
| LPPS_RX_DIG_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_dig_en |
| LPPS_DCOC_DIG_ALLOW | When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: dcoc_en |

### 64.3.5.2.15 FAST TSM Warmup and FAST RX2TX Transition

The previous sections illustrate the detailed control and timing generated by the TSM, operating in nominal-timing mode. There are situations that allow for the possibility to streamline and shorten the TSM warmup sequences. These situations arise when the transceiver is operating on the same RF channel for many consecutive transmit and/or

receive operations. Multiple closely-spaced TX operations on the same channel allow the HPM calibration to be skipped, after such a calibration is performed once on a given RF channel; similarly, multiple closely spaced RX operations on the same channel allow DC offset calibration to be skipped, after such a calibration is performed once on a given RF channel.

Fast TSM warmups have been devised to accomplish this streamlining. There is one fast warmup defined for TX, and one for RX. Once engaged, during a fast warmup, a portion of the nominal, programmed warmup is bypassed, or skipped over, by instructing the TSM counter to jump, once it reaches a certain count, to a new count that is not just an increment-by-one. The starting point for the "jump" as well as the destination count, or end point for the "jump", are TSM programmable registers. There are one pair of start/ destination registers for TX, and another for RX, allowing one such jump per sequence. In addition, enables bits provide software with fine control over precisely when fast warmups are allowed, versus nominal warmups.

In addition, TSM sequences have in the past been discrete TX or RX operations, with a required software command to initiate the sequence, and a return-to-idle after the sequence is complete, before a new sequence-start command can be issued. For 802.15.4, it is highly desirable to be able to jump directly from the end of a CCA measurement (CCA = Clear Channel Assessment), which is essentially an RX operation, to a packet transmit operation, without returning the TSM to IDLE in between, if CCA deems the channel to be free. A new TSM transition called "Fast RX2TX" accomplishes this. The 802.15.4 Link Layer will signal to TSM when to perform such a transition, and TSM will respond by jumping from the RX "ON" phase, to a programmable point within the TX warmup. At the very least, this will allow the re-warmup of the required analog regulators to be skipped, since they can stay on during the transition. Also, since the PLL is staying on the same channel, there is no need to cycle the VCO off, and then on again, so the VCO-related TSM controls can be programmed to maintain their states during the transition. Enable bits provide software with fine control over precisely when a "Fast RX2TX" is allowed, versus a nominal RX-to-IDLE-to-TX ordering of events.

Under software management, Fast RX2TX can be combined with Fast TX Warmup to yield an ultrafast TX warmup sequence in this 802.15.4 scenario, if, as is likely, the RF channel has not changed since the previous TX warmup.

Interaction with other blocks is required to support TSM fast warmup capability. The PLL digital block is responsible for selecting the correct frequency for RF operations, and will keep track of whether consecutive warmup operations utilize the same channel or not; it will signal this information to the TSM. The PLL digital block will maintain such tracking separately for TX and RX warmups. The TSM will then signal back to the PLL digital block when a TX warmup is actually "fast" or not; the PLL digital block will use

this information to instruct its state machine to skip HPM calibration. For RX warmups, the TSM will signal to the RX digital block, when to skip DC calibration, and instead use stored values.

The new inter-block signaling to support fast TSM warmups is illustrated in the following diagram, followed by a table which describes the new signaling.

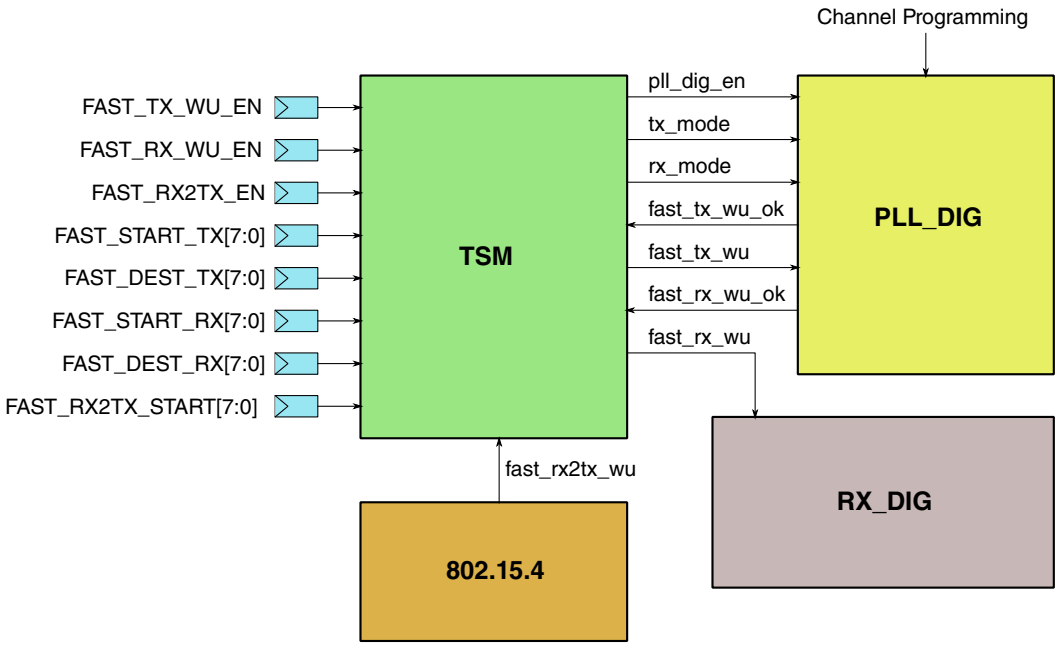



**Figure 64-79. XCVR SIGNALLING TO SUPPORT TSM FAST WARMUPS**

| "Fast Warmup" Signal | Source | Destination | Description |
|---|---|---|---|
| pll_dig_en | TSM | PLL Digital | Rising edge of this signal used by PLL_DIG to capture the Channel Programming for the current warmup |
| tx_mode | TSM | PLL Digital | Used in conjunction with pll_dig_en to capture the Channel Programming for TX sequences |
| rx_mode | TSM | PLL Digital | Used in conjunction with pll_dig_en to capture the Channel Programming for RX sequences |
| fast_tx_wu_ok | PLL Digital | TSM | Indicates that the Channel Programming on the *current* TX warmup has not changed since the *last* TX warmup |
| fast_tx_wu | TSM | PLL Digital | This is a fast TX warmup. PLL_DIG should skip HPM Cal. |

*Table continues on the next page...*

| "Fast Warmup" Signal | Source | Destination | Description |
|---|---|---|---|
| fast_rx_wu_ok | PLL Digital | TSM | Indicates that the Channel Programming on the *current* RX warmup has not changed since the *last* RX warmup |
| fast_rx_wu | TSM | RX Digital | This is a fast RX warmup. RX Dig should skip DCOC Cal. |
| fast_rx2tx_wu | ZSM (802.15.4) | TSM | ZSM has completed CCA and the RF channel is free to transmit. TSM should jump directly from RX "ON" phase to TX "WARMUP" phase, entering at the FAST_RX2TX_START point. |

TSM Registers to support Fast TSM Warmups and Fast RX2TX are described in the following table.

| Field | R/W | Description |
|---|---|---|
| FAST_TX_WU_EN | rw | 1: Enable Fast TX Warmup (if fast_tx_wu_ok asserted by PLL_DIG)<br>0: Disable Fast TX Warmups |
| FAST_RX_WU_EN | rw | 1: Enable Fast RX Warmup (if fast_rx_wu_ok asserted by PLL_DIG)<br>0: Disable Fast RX Warmups |
| FAST_RX2TX_EN | rw | 1: Enable Fast RX-to-TX transitions (if fast_rx2tx_wu is asserted by ZSM)<br>0: Disable Fast RX-to-TX transitions |
| FAST_START_TX[7:0] | rw | During a Fast TX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_TX[7:0], thereby executing the jump. |
| FAST_DEST_TX[7:0] | rw | During a Fast TX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, FAST_START_TX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>FAST_START_TX = 10 |

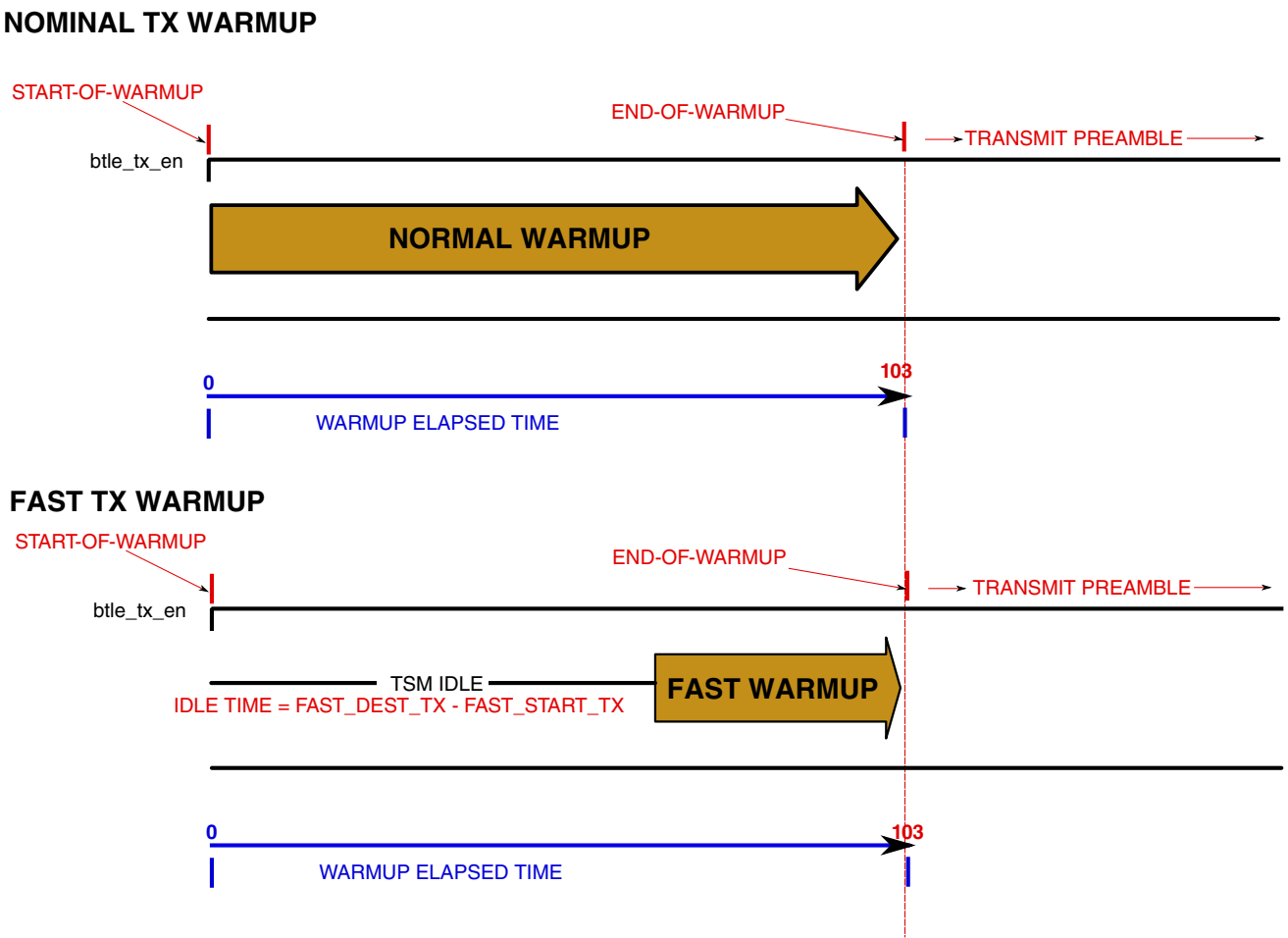
*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | FAST_DEST_TX = 15<br><br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| FAST_START_RX[7:0] | rw | During a Fast RX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_RX[7:0], thereby executing the jump. |
| FAST_DEST_RX[7:0] | rw | During a Fast RX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, FAST_START_RX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br><br>Example:<br><br>FAST_START_TX = 10<br>FAST_DEST_TX = 15<br><br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| FAST_RX2TX_START[7:0] | rw | During a Fast RX-to-TX transition (FAST_RX2TX_EN=1) the contents of this register represent the entry point (TSM count) in the TX warmup sequence to which the TSM jumps to when ZSM asserts fast_rx2tx_wu.<br><br>The TSM will jump from the RX “ON” phase to the TX “WARMUP” phase, starting at this count. |

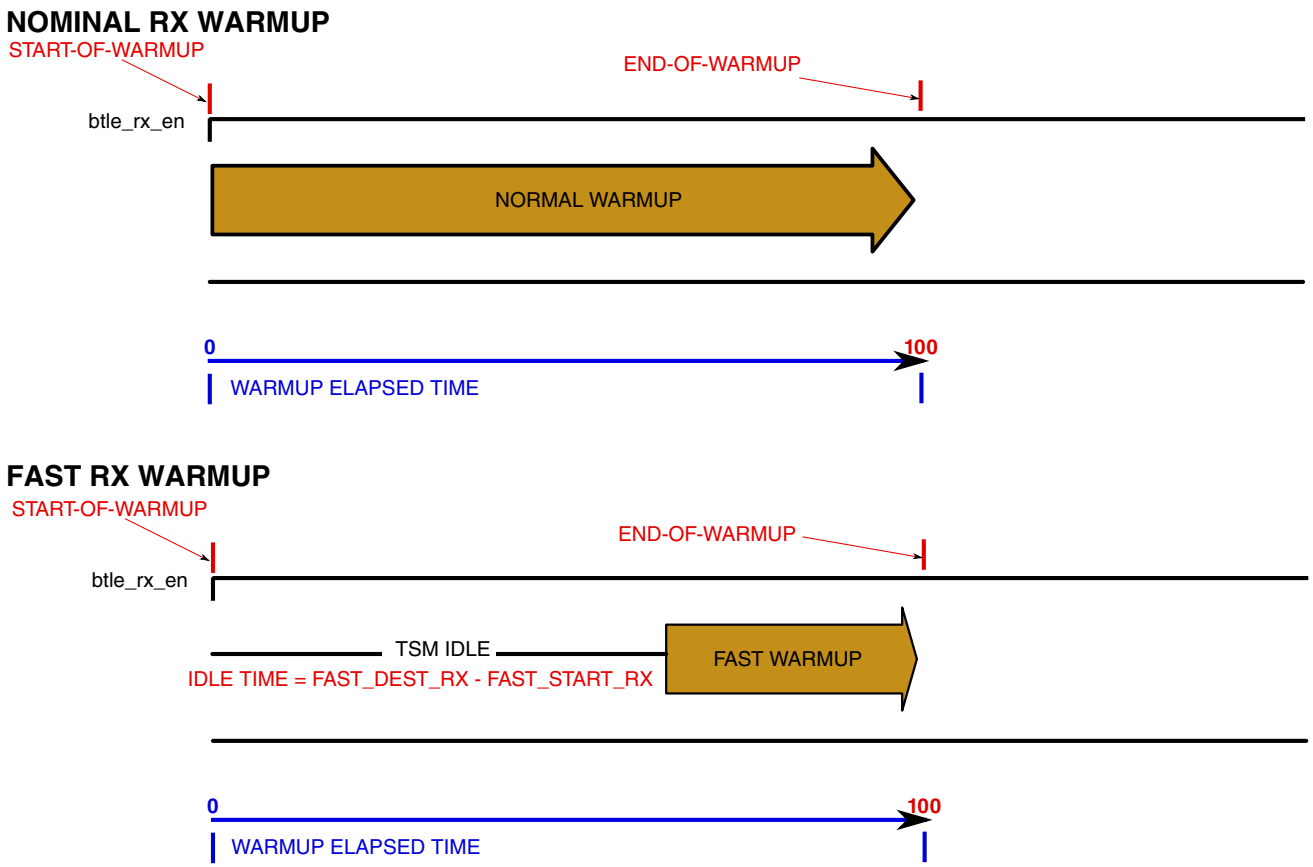
Software running on the various protocol engines which utilize the TSM to provide access to the channel, need not be aware that the total warmup time has been reduced, when the TSM executes a fast warmup, as opposed to a nominal warmup. Link Layer software expects TSM warmup times to be a fixed constant, actually 2 fixed constants, one for TX warmups and one for RX warmups. Software must account for these TX and RX warmup times when they are scheduling transceiver operations, so that the first symbol of preamble is transmitted at the correct instant for transmit operations, and that the receiver is ready to receive the first symbol of preamble at the correct instant for receive operations. Software views these warmup times as fixed lead times that must be accounted for in advance of a transceiver operation.

In order to make the fast warmup time reduction transparent to software, when a Fast TSM warmup is executed, the TSM will delay the start of the warmup, by the amount of time skipped by the jump. That keeps the total elapsed time, from the instant the software launches the TSM sequence, to the point at which the warmup completes, identical for fast and nominal warmups.

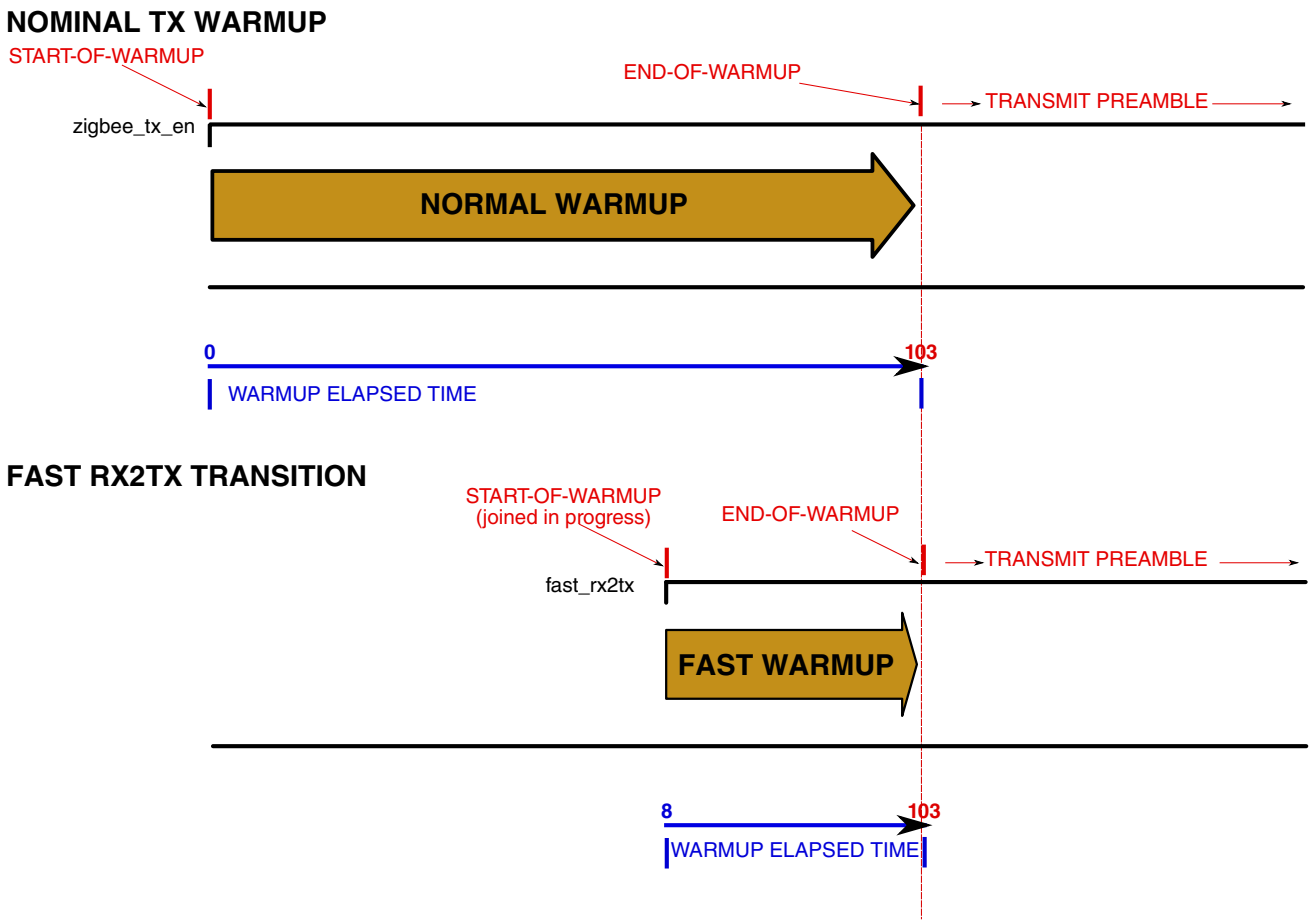
The following diagram illustrates how the TSM delays the start of the Fast TX sequence, relative to the nominal TX sequence, so that the delta time between btle_tx_en (TSM launch signal), and the transmission of the first bit of preamble, is the same in either case.




The following diagram illustrates how the TSM delays the start of the Fast RX sequence, relative to the nominal RX sequence, so that the delta time between btle_rx_en (TSM launch signal), and the ability to receive first bit of preamble, is the same in either case.




The FAST RX2TX transition is handled differently. The objective in such a transition is to get to the point of transmitting the first bit of preamble ASAP. There is no warmup time that software needs to account for in this case, since the RX-to-TX transition is handled completely by hardware. The following diagram compares a nominal TX warmup, with a foreshortened “Fast RX2TX” warmup.




## 64.3.5.2.16 Coexistence

The 2.4GHz radio includes mechanisms which facilitate packet traffic arbitration (PTA) between two collocated radios that will be sharing the same antenna for the 2.4GHz ISM band communications. For such mechanisms, IEEE 802.15.2 proposes MAC layer techniques to coordinate scheduling of packet transmission between the two wireless (WLAN and WPAN) networks by implementation of local controls. The mechanisms discussed in this document can be easily extended to work with the Bluetooth-LE radio, and in fact, all protocols currently supported by the 2.4GHz radio.

One such mechanism places the PTA arbiter on the WLAN device, and envisions a 3-wire interface to/from the WPAN device to the PTA. This protocol is supported by BT devices from some third-party vendors (Broadcom, CSR), and is shown in the image below ("BLUETOOTH" can be extrapolated to include any of the protocols supported by the 2.4GHz WPAN radio).

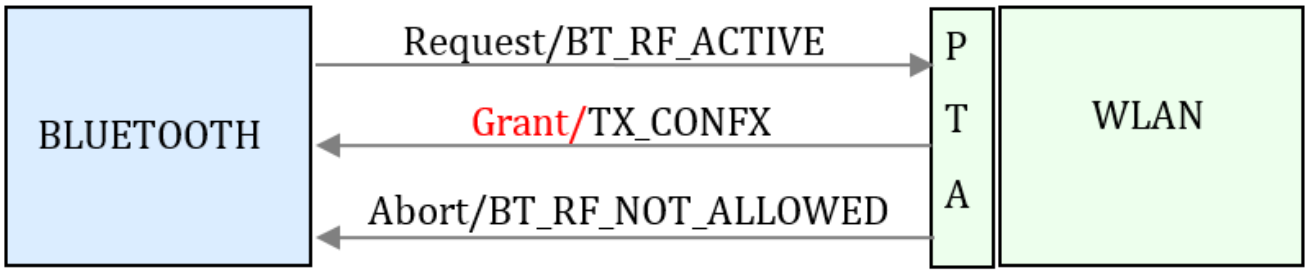



**Figure 64-80. 3-WIRE THIRD PARTY INTERFACE**

An alternative is the Nokia 4-wire interface, which is shown in the image below. (A 3-wire variant of the Nokia option exists, which merely omits the FREQ signal):

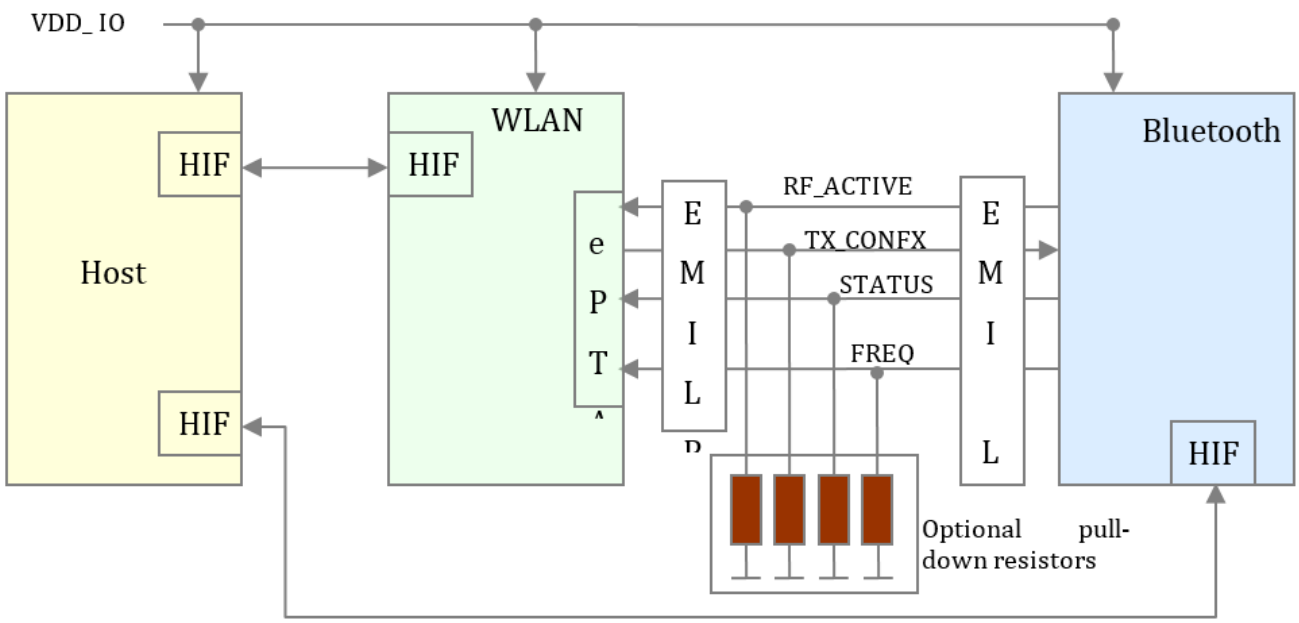



**Figure 64-81. 4-WIRE THIRD PARTY INTERFACE**

In either case, the signal RF_ACTIVE is asserted prior to any 2.4GHz WPAN radio transaction and it remains active for the duration of the transaction. Transaction is defined as a polling slot(s) and an immediately following response slot(s). Retransmissions are treated as separate transactions. The RF_ACTIVE pin at the SoC level can be driven by either an RSIM_RF_ACTIVE or a TSM_RF_ACTIVE source. TSM_RF_ACTIVE, since it is controlled by TSM, can be asserted at most ~200us before a radio event. TSM output TSM_SPARE1_EN is to be used as the TSM source for this purpose.

RF_ACTIVE needs to be asserted high a programmable amount of time before a radio event takes place and remains high until either the radio event has been completed or aborted. There should be no glitches/unwanted transitions between successive radio events (RX->TX or TX->TX or vice versa). This can be a problem for protocols, such as

BLE or 802.15.4, which require the transmission of an acknowledge packet shortly after reception. These protocols also require reception of an acknowledge packet shortly after transmission. BLE Advertising and Scan states offer other scenarios where closely-spaced (but not adjoining) TX and RX operations are required. The inter-frame spacing between the successive RX and TX operations, is long enough such that TSM_SPARE1_EN will deassert and reassert between operations, creating the unwanted transitions which must be avoided.

The following timing diagram shows the desired (glitch-free) assertion of RF_ACTIVE, encompassing a BLE RX operation followed by a TX operation.

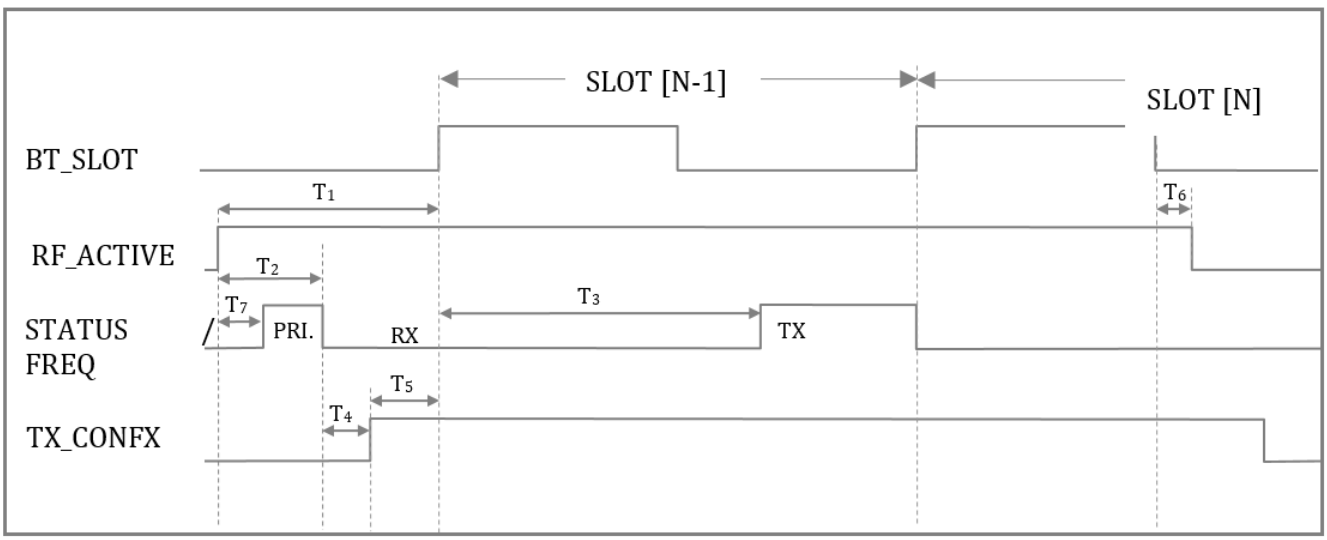



**Figure 64-82. IDEAL RF_ACTIVE TIMING (NO GLITCH)**

The following shows the actual signal timing which would occur on RF_ACTIVE (TSM_SPARE1_EN), in a typical BLE Advertising scenario, if no action were to be taken to address the unwanted, inter-frame transitions:

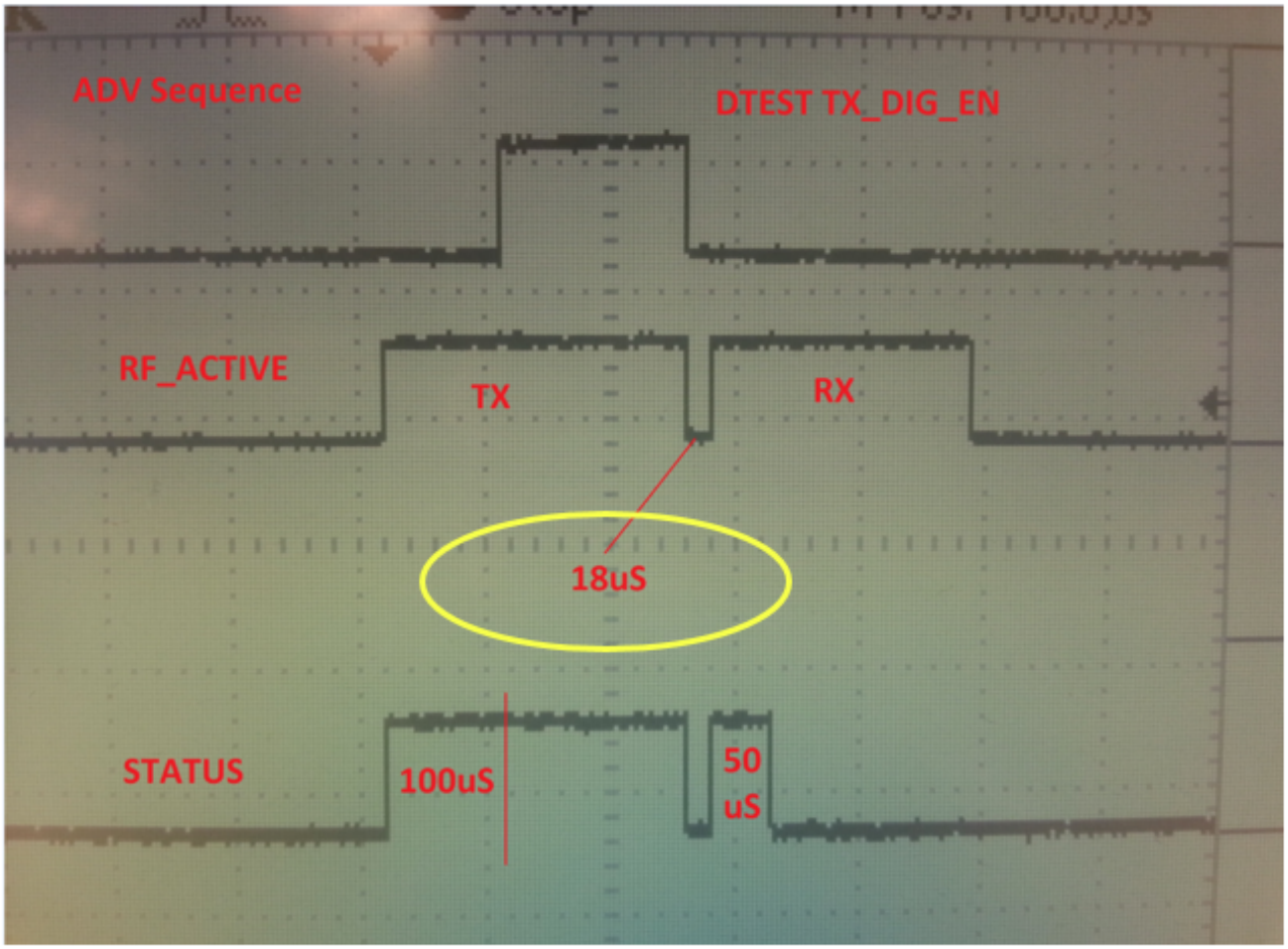



**Figure 64-83. ACTUAL RF_ACTIVE TIMING (UNCOMPENSATED)**

To remedy this, a feature has been added to TSM, to allow the assertion of TSM_SPARE1_EN output to be optionally extended by a programmable amount, after the nominal TSM sequence completes. The programmable limit is long enough to accommodate the RF_ACTIVE glitch which would otherwise occur using the 802.15.4 192µs turnaround time. The TSM_SPARE1_EN extension will occur for all sequences (TX and RX) for which TSM_SPARE1_EN is programmed to assert, whenever register TSM_SPARE1_EXTEND>0. The hardware mechanism for extending TSM_SPARE1_EN is illustrated in the following diagram:

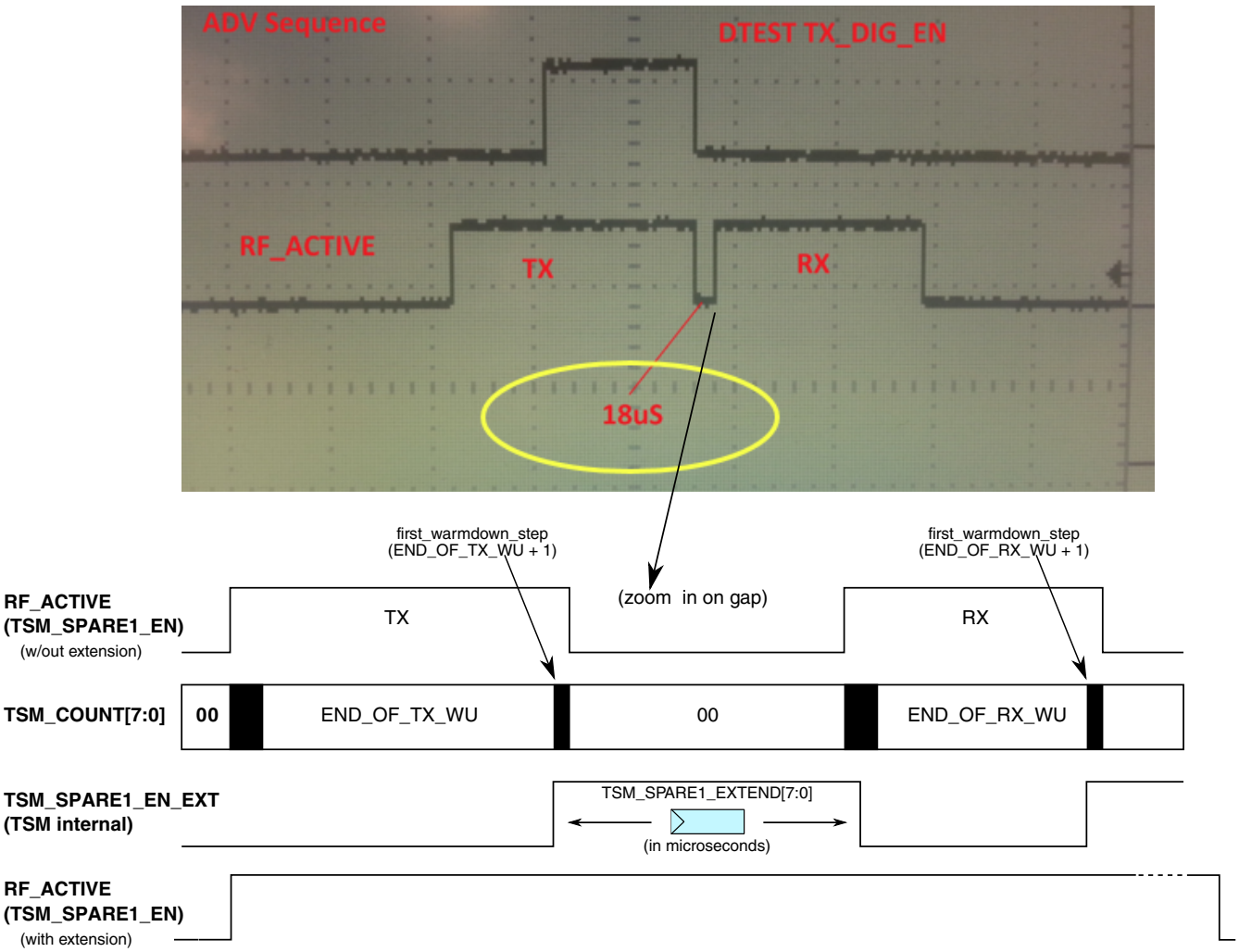



**Figure 64-84. TSM_SPARE1_EN EXTENSION**

The trigger for the extension is the TSM's first warmdown step (END_OF_xx_WU + 1), which will be reached whenever the Link Layer deasserts its TX/RX command, or on any abort. If enabled, the extension signal will assert (high) at this point, and will remain high for *TSM_SPARE1_EXTEND* microseconds. The extension signal (TSM_SPARE1_EN_EXT) is logically "OR-ed" with the existing TSM-controlled output TSM_SPARE1_EN, to generate the composite, extended RF_ACTIVE signal. Software can cut short the extension at any time by writing TSM_SPARE1_EXTEND=0. Setting this register to 0 also disables the extension feature from activating on any new TSM sequence. The register TSM_SPARE1_EXTEND[7:0] resides in the COEX_CTRL register of XCVR address space, and yields an extension range from 0 to 255µs.

### 64.3.5.2.17 Programming Restrictions

The following restrictions are imposed on TSM programming. Violating the restrictions may result in unpredictable behavior.

1. The maximum value for END_OF_TX_WU[7:0] is 253
2. The maximum value for END_OF_RX_WU[7:0] is 253
3. The maximum value for END_OF_TX_WD[7:0] is 254
4. The maximum value for END_OF_RX_WD[7:0] is 254
5. END_OF_TX_WD[7:0] – END_OF_TX_WU[7:0] >= 1
6. END_OF_RX_WD[7:0] – END_OF_RX_WU[7:0] >= 1
7. BKPT[7:0] != END_OF_TX_WD[7:0]
8. BKPT[7:0] != END_OF_RX_WD[7:0]
9. For any TSM output: OUTPUTNAME_TX_HI[7:0] <= OUTPUTNAME_TX_LO[7:0]
10. For any TSM output: OUTPUTNAME_RX_HI[7:0] <= OUTPUTNAME_RX_LO[7:0]

Restrictions 9 and 10 above, if violated, will result in the TSM-controlled output held in a deasserted state throughout the sequence.

To guarantee non-assertion of a particular TSM-controlled output throughout the course of a sequence, simply program its OUTPUTNAME_SEQ_HI = OUTPUTNAME_SEQ_LO = 255 (where SEQ = TX or RX).

#### 64.3.5.2.18 Clocks

The TSM uses 4 clocks:

1. ipg_clk
2. tsm_clk
3. tsm_dly_clk
4. tsm_4m_clk

The "**ipg_clk",** is a gated 32mhz (or 26mhz) clock from the reference oscillator. This clock only used to re-clock the tx_pwr_cntl[5:0] output, to ensure no glitching or skew between the bits, on this power control bus to the PA.

The main clock is the 1MHz **tsm_clk**. This clock establishes the TSM timebase (1us), and runs the TSM counter, as well as the control and abort-handling logic.

Another 1MHz clock is **tsm_dly_clk**. This clock is used to re-clock the 67 TSM-controlled outputs. This clock is time-shifted (delayed) by 1 ipg_clk cycle relative to tsm_clk.

The 4Mhz clock **tsm_4m_clk** runs the PA ramping logic.

All 4 TSM clocks derive from the same 32MHz (or 26mhz) source, the reference oscillator. They are divided down from this reference frequency in the CGM (Clocks Generator Module). Since all 4 are in the same clock domain, they will be balanced (skew-controlled) during clock-tree synthesis. There there are no clock-domain-crossings or asynchronous interfaces in the TSM.

#### 64.3.5.2.19 Reset

The TSM has a single, active-low, asynchronous reset input: **ipg_hard_async_reset_b** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

#### 64.3.5.2.20 Interrupts

For debug purposes, TSM can generate up to 2 interrupts to the host processor. Like all other TSM-controlled outputs, the point during the TX or RX warmup at which the interrupt is triggered, is fully programmable. Since there is no defined mission-mode use for TSM interrupts, no new dedicated timing registers have been provisioned. Instead, the 2 TSM interrupts share timing registers with TSM-controlled outputs **tsm_spare0_en** and **tsm_spare1_en**. The following diagram depicts the reuse scheme.

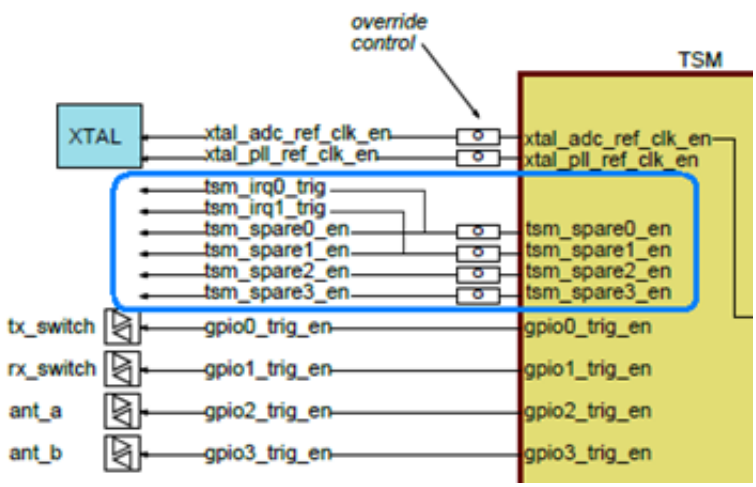



**Figure 64-85. TSM Interrupt Re-use Scheme**

Two new control bits have been added to TSM_CTRL to enable each TSM interrupt. The following table lists the TSM Interrupt, the TSM enable bit for the interrupt, and the TSM timing register which controls the timing of the interrupt.

**Table 64-39. TSM Interrupt Control**

| TSM Interrupt | Interrupt Enable Bit (TSM_CTRL) | Timing Control Register |
|---|---|---|
| TSM_IRQ0 | TSM_IRQ0_EN | TSM_TIMING43 |
| TSM_IRQ1 | TSM_IRQ1_EN | TSM_TIMING44 |

Because the TSM is part of XCVR space, and XCVR space does not have its own interrupt vector, the TSM interrupt is "assigned" to the Link Layer currently in possession of the 2.4GHz radio. Possession is determined by the state of the XCVR_CTRL[PROTOCOL] bits. Each of the Link Layer controllers (BLE, GENERIC_FSK, 802.15.4), has in its address space, a TSM_IRQ status bit, as well as a enable bit to allow/disallow the TSM interrupt to assert the Link Layer's interrupt line to the MCU. The bit positioning of the TSM_IRQ varies slightly amongst the various Link Layers, to optimally group the TSM_IRQ with existing, protocol-specific interrupt sources, and to minimize the software impact to existing service routines, by requiring as few additional register reads as possible to service the Link Layer's interrupts with TSM_IRQ enabled. The TSM_IRQ status bit, as it appears in each Link Layer's address space, is a logical "OR" of the TSM's TSM0_IRQ and TSM1_IRQ. The Link Layer TSM_IRQ is read-only. It can be cleared by clearing both TSM0_IRQ and TSM1_IRQ in the XCVR's XCVR_STATUS register. In the XCVR_STATUS register, TSM0_IRQ and TSM1_IRQ are both write-1-to-clear bits. See the Register Description for each individual Link Layer, for a description of the TSM_IRQ read-only interrupt status bit, and its bit positioning within the Link Layer's address space.

### 64.3.5.3 Memory Map and register definition

There are 59 sets of timing registers for the TSM, one set for each TSM-controlled output (or, for the signal controlling the group for the “grouped” outputs). Each register set consists of 4 registers: one to control the assertion time of the output for TX sequences, one to control the assertion time for RX sequence, one to control the deassertion time for TX sequences, and one to control the deassertion time for RX sequences. For TSM-controlled outputs that are grouped, one set of timing registers controls the timing for all members of the group. For each of these 59 sets, the following table lists the SoC register name (TSM_TIMING00 – TSM_TIMING58), and 4 timing register names associated with each. There are various other registers to control the overall TX and RX sequence lengths, configure for Fast Warmups, etc.

See below for the TSM register locations and detailed definitions.

## 64.3.5.3.1 XCVR_TSM register descriptions

### 64.3.5.3.1.1 XCVR_TSM_ADDR Memory map

XCVR_TSM base address: 4103_02C0h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TSM CONTROL (CTRL) | 32 | RW | FF00_4000h |
| 4h | TSM END OF SEQUENCE (END_OF_SEQ) | 32 | RW | 6766_6A63h |
| 8h | PA POWER (PA_POWER) | 32 | RW | 0000_0000h |
| Ch | PA RAMP TABLE 0 (PA_RAMP_TBL0) | 32 | RW | 0604_0201h |
| 10h | PA RAMP TABLE 1 (PA_RAMP_TBL1) | 32 | RW | 1410_0C08h |
| 14h | PA RAMP TABLE 2 (PA_RAMP_TBL2) | 32 | RW | 2822_1C18h |
| 18h | PA RAMP TABLE 3 (PA_RAMP_TBL3) | 32 | RW | 3C36_302Ch |
| 24h | TSM RECYCLE COUNT (RECYCLE_COUNT) | 32 | RW | 001A_0464h |
| 28h | TSM FAST WARMUP CONTROL 1 (FAST_CTRL1) | 32 | RW | 0000_FF00h |
| 2Ch | TSM FAST WARMUP CONTROL 2 (FAST_CTRL2) | 32 | RW | FFFF_FFFFh |
| 30h | TSM_TIMING00 (TIMING00) | 32 | RW | 6700_6A00h |
| 34h | TSM_TIMING01 (TIMING01) | 32 | RW | 6700_6A00h |
| 38h | TSM_TIMING02 (TIMING02) | 32 | RW | 6700_FFFFh |
| 3Ch | TSM_TIMING03 (TIMING03) | 32 | RW | 6700_6A00h |
| 40h | TSM_TIMING04 (TIMING04) | 32 | RW | 6700_6A00h |
| 44h | TSM_TIMING05 (TIMING05) | 32 | RW | 6700_6A00h |
| 48h | TSM_TIMING06 (TIMING06) | 32 | RW | 6700_6A00h |
| 4Ch | TSM_TIMING07 (TIMING07) | 32 | RW | 0500_0500h |
| 50h | TSM_TIMING08 (TIMING08) | 32 | RW | 0300_0300h |
| 54h | TSM_TIMING09 (TIMING09) | 32 | RW | 0300_0300h |
| 58h | TSM_TIMING10 (TIMING10) | 32 | RW | 6703_6A03h |
| 5Ch | TSM_TIMING11 (TIMING11) | 32 | RW | FFFF_6A03h |
| 60h | TSM_TIMING12 (TIMING12) | 32 | RW | 6703_FFFFh |
| 64h | TSM_TIMING13 (TIMING13) | 32 | RW | 1600_4A00h |
| 68h | TSM_TIMING14 (TIMING14) | 32 | RW | 672F_645Fh |
| 6Ch | TSM_TIMING15 (TIMING15) | 32 | RW | 6703_6A03h |
| 70h | TSM_TIMING16 (TIMING16) | 32 | RW | 671A_FFFFh |
| 74h | TSM_TIMING17 (TIMING17) | 32 | RW | FFFF_6A5Ah |
| 78h | TSM_TIMING18 (TIMING18) | 32 | RW | 6705_6A05h |
| 7Ch | TSM_TIMING19 (TIMING19) | 32 | RW | 1605_4A05h |
| 80h | TSM_TIMING20 (TIMING20) | 32 | RW | 6705_6A05h |
| 84h | TSM_TIMING21 (TIMING21) | 32 | RW | 6704_6A04h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 88h | TSM_TIMING22 (TIMING22) | 32 | RW | 6704_FFFFh |
| 8Ch | TSM_TIMING23 (TIMING23) | 32 | RW | FFFF_6A04h |
| 90h | TSM_TIMING24 (TIMING24) | 32 | RW | 1600_4A00h |
| 94h | TSM_TIMING25 (TIMING25) | 32 | RW | 671B_FFFFh |
| 98h | TSM_TIMING26 (TIMING26) | 32 | RW | FFFF_6A5Ah |
| 9Ch | TSM_TIMING27 (TIMING27) | 32 | RW | 671E_FFFFh |
| A0h | TSM_TIMING28 (TIMING28) | 32 | RW | 1F1E_FFFFh |
| A4h | TSM_TIMING29 (TIMING29) | 32 | RW | 671C_FFFFh |
| A8h | TSM_TIMING30 (TIMING30) | 32 | RW | 671E_FFFFh |
| ACh | TSM_TIMING31 (TIMING31) | 32 | RW | 671D_FFFFh |
| B0h | TSM_TIMING32 (TIMING32) | 32 | RW | 671B_FFFFh |
| B4h | TSM_TIMING33 (TIMING33) | 32 | RW | 671E_FFFFh |
| B8h | TSM_TIMING34 (TIMING34) | 32 | RW | 6705_6A05h |
| BCh | TSM_TIMING35 (TIMING35) | 32 | RW | FFFF_6A5Dh |
| C0h | TSM_TIMING36 (TIMING36) | 32 | RW | 6764_FFFFh |
| C4h | TSM_TIMING37 (TIMING37) | 32 | RW | 6564_FFFFh |
| C8h | TSM_TIMING38 (TIMING38) | 32 | RW | 670C_6A40h |
| CCh | TSM_TIMING39 (TIMING39) | 32 | RW | 6764_FFFFh |
| D0h | TSM_TIMING40 (TIMING40) | 32 | RW | 6724_FFFFh |
| D4h | TSM_TIMING41 (TIMING41) | 32 | RW | 2524_FFFFh |
| D8h | TSM_TIMING42 (TIMING42) | 32 | RW | FFFF_FFFFh |
| DCh | TSM_TIMING43 (TIMING43) | 32 | RW | FFFF_FFFFh |
| E0h | TSM_TIMING44 (TIMING44) | 32 | RW | FFFF_FFFFh |
| E4h | TSM_TIMING45 (TIMING45) | 32 | RW | FFFF_FFFFh |
| E8h | TSM_TIMING46 (TIMING46) | 32 | RW | FFFF_FFFFh |
| ECh | TSM_TIMING47 (TIMING47) | 32 | RW | FFFF_FFFFh |
| F0h | TSM_TIMING48 (TIMING48) | 32 | RW | FFFF_FFFFh |
| F4h | TSM_TIMING49 (TIMING49) | 32 | RW | FFFF_FFFFh |
| F8h | TSM_TIMING50 (TIMING50) | 32 | RW | FFFF_FFFFh |
| FCh | TSM_TIMING51 (TIMING51) | 32 | RW | 6703_FFFFh |
| 100h | TSM_TIMING52 (TIMING52) | 32 | RW | 1504_FFFFh |
| 104h | TSM_TIMING53 (TIMING53) | 32 | RW | 6704_FFFFh |
| 108h | TSM_TIMING54 (TIMING54) | 32 | RW | 1504_FFFFh |
| 10Ch | TSM_TIMING55 (TIMING55) | 32 | RW | 671E_FFFFh |
| 110h | TSM_TIMING56 (TIMING56) | 32 | RW | 671E_FFFFh |
| 114h | TSM_TIMING57 (TIMING57) | 32 | RW | 1A03_FFFFh |
| 118h | TSM_TIMING58 (TIMING58) | 32 | RW | FFFF_6A03h |
| 11Ch | TSM OVERRIDE REGISTER 0 (OVRD0) | 32 | RW | 0000_0000h |
| 120h | TSM OVERRIDE REGISTER 1 (OVRD1) | 32 | RW | 0000_0000h |
| 124h | TSM OVERRIDE REGISTER 2 (OVRD2) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

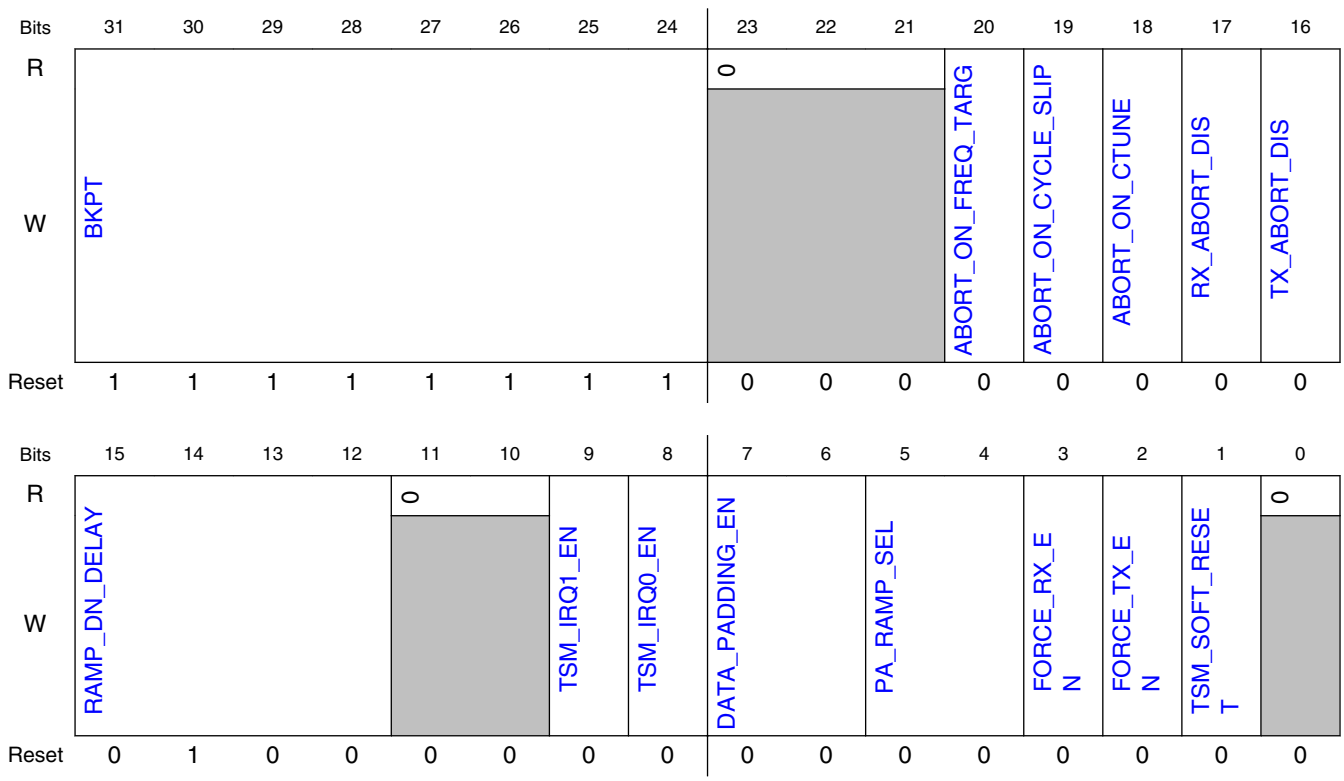
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 128h | TSM OVERRIDE REGISTER 3 (OVRD3) | 32 | RW | 0000_0000h |

## 64.3.5.3.1.2 TSM CONTROL (CTRL)

### *64.3.5.3.1.2.1 Offset*

| Register | Offset |
|---|---|
| CTRL | 0h |

### *64.3.5.3.1.2.2 Diagram*




### *64.3.5.3.1.2.3 Fields*

| Field | Function |
|---|---|
| 31-24 | TSM Breakpoint |
| BKPT | Temporarily halt a TSM sequence during the warmup or warmdown phase. When the TSM counter matches the value of BKPT[7:0], breakpoint will take effect and the TSM counter will stop and hold its |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | count. Breakpoint will remain in effect as long as BKPT[7:0] matches the TSM counter value. The TSM Breakpoint can be lifted by modifying the contents of this register. The default value of this register, 0xFF, is greater than the length of the longest possible sequence, so a breakpoint will never be triggered unless BKPT[7:0] is programmed to a value less than the length of sequence. |
| 23-21<br>— | Reserved |
| 20<br>ABORT_ON_FREQ_TARG | Abort On Frequency Target Lock Detect Failure<br>0b - don't allow TSM abort on Frequency Target Unlock Detect<br>1b - allow TSM abort on Frequency Target Unlock Detect |
| 19<br>ABORT_ON_CYCLE_SLIP | Abort On Cycle Slip Lock Detect Failure<br>0b - don't allow TSM abort on Cycle Slip Unlock Detect<br>1b - allow TSM abort on Cycle Slip Unlock Detect |
| 18<br>ABORT_ON_CTUNE | Abort On Coarse Tune Lock Detect Failure<br>0b - don't allow TSM abort on Coarse Tune Unlock Detect<br>1b - allow TSM abort on Coarse Tune Unlock Detect |
| 17<br>RX_ABORT_DIS | Receive Abort Disable<br>RX Abort disable. When set, prevents PLL unlock events during RX sequences from aborting the sequence. |
| 16<br>TX_ABORT_DIS | Transmit Abort Disable<br>TX Abort disable. When set, prevents PLL unlock events during TX sequences from aborting the sequence. |
| 15-12<br>RAMP_DN_DELAY | PA Ramp Down Delay<br>Delays the start of the PA Ramp Down, relative to the start of the TSM warmdown, by *N* microseconds, where *N*=RAMP_DN_DELAY. Range is 0 to 15us. |
| 11-10<br>— | Reserved |
| 9<br>TSM_IRQ1_EN | TSM_IRQ1 Enable/Disable bit<br>TSM_IRQ1 Enable/Disable bit. Intended for debug purposes only.<br>0b - TSM_IRQ1 is disabled<br>1b - TSM_IRQ1 is enabled |
| 8<br>TSM_IRQ0_EN | TSM_IRQ0 Enable/Disable bit<br>TSM_IRQ0 Enable/Disable bit. Intended for debug purposes only.<br>0b - TSM_IRQ0 is disabled<br>1b - TSM_IRQ0 is enabled |
| 7-6<br>DATA_PADDING_EN | Data Padding Enable<br>Enables TX Data Padding. Data padding works in conjunction with PA ramping to minimize spectral transients during PA turn-on and turn-off. The nature of the data padding depends on the setting of XCVR_CTRL[PROTOCOL].<br>00b - Disable TX Data Padding<br>01b - Enable TX Data Padding |
| 5-4<br>PA_RAMP_SEL | PA Ramp Selection<br>Selects the ramp-rate, and thus the duration, for PA ramping. Ramp-rate is the rate at which the PA ramping logic steps through the PA Ramp Table. There are always 16 ramp steps, if ramping is enabled.<br>The following table lists the total ramp duration for each PA_RAMP_SEL setting: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0: No ramp<br>1: 1.0µs<br>2: 2.0µs<br>3: 4.0µs<br><br>The following table lists the duration of each ramp step for each PA_RAMP_SEL setting:<br><br>0: No ramp<br>1: 2 Reference Clocks (0.0625µs)<br>2: 4 Reference Clocks (0.125µs)<br>3: 8 Reference Clocks (0.25µs) |
| 3<br><br>FORCE_RX_EN | Force Receive Enable<br><br>Direct software control to launch a RX TSM sequence. Initiates RX Warmup on a 0 to 1 transition and RX Warmdown on a 1 to 0 transition.<br>0b - TSM Idle<br>1b - TSM executes a RX sequence |
| 2<br><br>FORCE_TX_EN | Force Transmit Enable<br><br>Direct software control to launch a TX TSM sequence. Initiates a TX Warmup sequence on a 0 to 1 transition and a TX Warmdown sequence on a 1 to 0 transition.<br>0b - TSM Idle<br>1b - TSM executes a TX sequence |
| 1<br><br>TSM_SOFT_RESET | TSM Soft Reset<br><br>Writing 1 to this bit returns TSM to its IDLE state, with TSM_COUNT=0. Writing 0 to this bit removes the forced-reset condition and resumes normal operation. The bit is not self-clearing. For contingency purposes only<br>0b - TSM Soft Reset removed. Normal operation.<br>1b - TSM Soft Reset engaged. TSM forced to IDLE, and holds there until the bit is cleared. |
| 0<br><br>— | Reserved |

## 64.3.5.3.1.3 TSM END OF SEQUENCE (END_OF_SEQ)

### *64.3.5.3.1.3.1 Offset*

| Register | Offset |
|---|---|
| END_OF_SEQ | 4h |

#### 64.3.5.3.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | END_OF_RX_WD | | | | | | | | END_OF_RX_WU | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | END_OF_TX_WD | | | | | | | | END_OF_TX_WU | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

#### 64.3.5.3.1.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>END_OF_RX_WD | End of RX Warmdown<br>This register defines the point at which the TSM RX sequence warmdown completes, and the TSM returns to idle. The duration of the TSM warmdown phase is determined by: END_OF_RX_WD – END_OF_RX_WU. |
| 23-16<br>END_OF_RX_WU | End of RX Warmup<br>This register defines the length of the TSM RX warmup sequence. After the assertion of a RX sequence-initiating event, when the TSM counter reaches the count matching this register, it will stop and hold its count, and the TSM will transition from the WARMUP to the ON phase. |
| 15-8<br>END_OF_TX_WD | End of TX Warmdown<br>This register defines the point at which the TSM TX sequence warmdown completes, and the TSM returns to idle. The duration of the TSM warmdown phase is determined by: END_OF_TX_WD – END_OF_TX_WU. |
| 7-0<br>END_OF_TX_WU | End of TX Warmup<br>This register defines the length of the TSM TX warmup sequence. After the assertion of a TX sequence-initiating event, when the TSM counter reaches the count matching this register, it will stop and hold its count, and the TSM will transition from the WARMUP to the ON phase. |

## 64.3.5.3.1.4 PA POWER (PA_POWER)

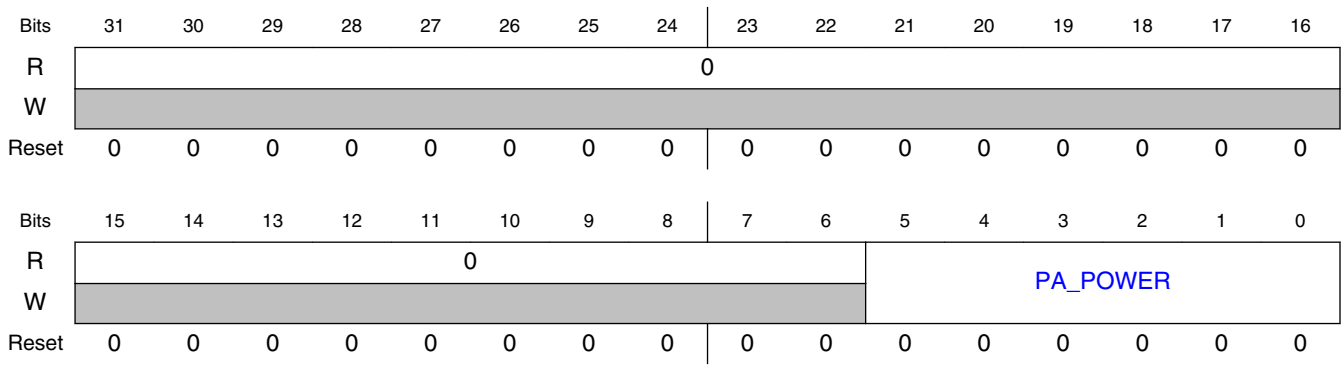
#### 64.3.5.3.1.4.1 Offset

| Register | Offset |
|---|---|
| PA_POWER | 8h |

#### 64.3.5.3.1.4.2 Function

This contents of this register are used as PA target power when XCVR_CTRL[TGT_PWR_SRC] = 0

#### 64.3.5.3.1.4.3 Diagram




#### 64.3.5.3.1.4.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0<br>PA_POWER | PA POWER<br>This contents of this register are used as PA target power when XCVR_CTRL[TGT_PWR_SRC] = 0.The valid values for this field are 0,1,2,4,6,8,…,62 (in increasing order of PA power). That is, above 1, only even numbers are permitted in the PA_POWER bitfield. Odd values of 3..63 are not permitted and will result in unexpected behavior. |

### 64.3.5.3.1.5 PA RAMP TABLE 0 (PA_RAMP_TBL0)

#### 64.3.5.3.1.5.1 Offset

| Register | Offset |
|---|---|
| PA_RAMP_TBL0 | Ch |

#### 64.3.5.3.1.5.2 Function

PA Ramp Table 0

### 64.3.5.3.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP3 | | | | | | 0 | | PA_RAMP2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP1 | | | | | | 0 | | PA_RAMP0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 64.3.5.3.1.5.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP3 | PA_RAMP3<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fourth ramp step. During PA ramp down, the contents of this register are the PA power value during the fourth-to-last ramp step. In both cases, PA_RAMP3 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP2 | PA_RAMP2<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the third ramp step. During PA ramp down, the contents of this register are the PA power value during the third-to-last ramp step. In both cases, PA_RAMP2 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP1 | PA_RAMP1<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the second ramp step. During PA ramp down, the contents of this register are the PA power value during the second-to-last ramp step. In both cases, PA_RAMP1 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP0 | PA_RAMP0<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, when TSM tx_pa_en transitions low to high, and then for the duration of the first ramp step. During PA ramp down, the contents of this register are the PA power value during the final ramp step. In both cases, PA_RAMP0 cannot exceed target power (enforced by PA ramping logic). When PA ramping is enabled, the contents of PA_RAMP0 are also presented to the PA during sequence-idle conditions. |

## 64.3.5.3.1.6 PA RAMP TABLE 1 (PA_RAMP_TBL1)

### *64.3.5.3.1.6.1 Offset*

| Register | Offset |
|---|---|
| PA_RAMP_TBL1 | 10h |

### *64.3.5.3.1.6.2 Function*

PA Ramp Table 1

### *64.3.5.3.1.6.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP7 | | | | | | 0 | | PA_RAMP6 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP5 | | | | | | 0 | | PA_RAMP4 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

### *64.3.5.3.1.6.4 Fields*

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP7 | PA_RAMP7<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the eighth ramp step. During PA ramp down, the contents of this register are the PA power value during the eighth-to-last ramp step. In both cases, PA_RAMP7 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP6 | PA_RAMP6<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the seventh ramp step. During PA ramp down, the contents of this register are the PA power value during the seventh-to-last ramp step. In both cases, PA_RAMP6 cannot exceed target power (enforced by PA ramping logic). |
| 15-14 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 13-8<br>PA_RAMP5 | PA_RAMP5<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the sixth ramp step. During PA ramp down, the contents of this register are the PA power value during the sixth-to-last ramp step. In both cases, PA_RAMP5 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP4 | PA_RAMP4<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fifth ramp step. During PA ramp down, the contents of this register are the PA power value during the fifth-to-last ramp step. In both cases, PA_RAMP4 cannot exceed target power (enforced by PA ramping logic). |

## 64.3.5.3.1.7 PA RAMP TABLE 2 (PA_RAMP_TBL2)

### *64.3.5.3.1.7.1 Offset*

| Register | Offset |
|---|---|
| PA_RAMP_TBL2 | 14h |

### *64.3.5.3.1.7.2 Function*

PA Ramp Table 2

### *64.3.5.3.1.7.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP11 | | | | | | 0 | | PA_RAMP10 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP9 | | | | | | 0 | | PA_RAMP8 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

#### *64.3.5.3.1.7.4 Fields*

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP11 | PA_RAMP11<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the twelvth ramp step. During PA ramp down, the contents of this register are the PA power value during the twelvth-to-last ramp step. In both cases, PA_RAMP11 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP10 | PA_RAMP10<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the eleventh ramp step. During PA ramp down, the contents of this register are the PA power value during the eleventh-to-last ramp step. In both cases, PA_RAMP10 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP9 | PA_RAMP9<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the tenth ramp step. During PA ramp down, the contents of this register are the PA power value during the tenth-to-last ramp step. In both cases, PA_RAMP9 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP8 | PA_RAMP8<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the ninth ramp step. During PA ramp down, the contents of this register are the PA power value during the ninth-to-last ramp step. In both cases, PA_RAMP8 cannot exceed target power (enforced by PA ramping logic). |

## 64.3.5.3.1.8 PA RAMP TABLE 3 (PA_RAMP_TBL3)

#### *64.3.5.3.1.8.1 Offset*

| Register | Offset |
|---|---|
| PA_RAMP_TBL3 | 18h |

#### *64.3.5.3.1.8.2 Function*

PA Ramp Table 3

### 64.3.5.3.1.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP15 | | | | | | 0 | | PA_RAMP14 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP13 | | | | | | 0 | | PA_RAMP12 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

### 64.3.5.3.1.8.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP15 | PA_RAMP15<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the sixteenth (final) ramp step. During PA ramp down, the contents of this register are the PA power value during the sixteenth-to-last (first) ramp step. In both cases, PA_RAMP15 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP14 | PA_RAMP14<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fifteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the fifteenth-to-last ramp step. In both cases, PA_RAMP14 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP13 | PA_RAMP13<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fourteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the fourteenth-to-last ramp step. In both cases, PA_RAMP13 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP12 | PA_RAMP12<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the thirteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the thirteenth-to-last ramp step. In both cases, PA_RAMP12 cannot exceed target power (enforced by PA ramping logic). |

## 64.3.5.3.1.9 TSM RECYCLE COUNT (RECYCLE_COUNT)

### *64.3.5.3.1.9.1 Offset*

| Register | Offset |
|---|---|
| RECYCLE_COUNT | 24h |

### *64.3.5.3.1.9.2 Function*

This register contains the TSM Recycle "Jump-to" points for the 3 types of TSM Recycle

### *64.3.5.3.1.9.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RECYCLE_COUNT2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RECYCLE_COUNT1 | | | | | | | | RECYCLE_COUNT0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

### *64.3.5.3.1.9.4 Fields*

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>RECYCLE_COUNT2 | TSM RX Recycle Count 2<br>The RECYCLE_COUNT2[7:0] register determines the TSM count value to which the TSM "recycles" when, in LPPS mode, an LPPS power save cycle completes and the receiver is switched back on, necessitating a foreshortened RX warmup. The intention is for this register to be programmed to a TSM count value that is just prior to the assertion of rx_lna_en, but there are no restrictions on programming this register. An LPPS recycle is a command from the LPPS state machine to TSM, requesting a jump from the TSM ON phase back to a programmable point in the WARMUP phase, and resume counting from there. |
| 15-8<br>RECYCLE_COUNT1 | TSM RX Recycle Count 1<br>The RECYCLE_COUNT1[7:0] register determines the TSM count value to which the TSM "recycles" when the 802.15.4 Sequence Manager (ZSM) state "RX_PAN1" is reached and the ZSM asserts tsm_recycle[1] to TSM. The intention is for this register to be programmed to a TSM count value such that the TSM de-asserts, and then re-asserts its "pll_dig_en" output, to effectuate a Dual PAN on-the-fly channel change, but there are no restrictions on programming this register. An RX recycle is a command from ZSM to TSM, requesting a jump from the TSM ON phase back to a programmable point in the |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | WARMUP phase, and resume counting from there. A recycle will result from the expiration of the Dual PAN Dwell Timer, at which point an RF-channel change is required. This necessitates the desassertion and reassertion of pll_dig_en, hence the return to the WARMUP phase at the appropriate point. |
| 7-0<br>RECYCLE_COUNT0 | TSM RX Recycle Count 0<br>The RECYCLE_COUNT0[7:0] register determines the TSM count value to which the TSM "recycles" when the 802.15.4 Sequence Manager (ZSM) state "RX_CYC" is reached and the ZSM asserts tsm_recycle[0] to TSM. This register also determines the TSM count value to which the TSM recycles when the ZSM state RX_CCCA is reached because tsm_recycle[0] is also asserted in this state. This register also determines the TSM count value to which the TSM recycles when theGeneric_FSK Link Layer asserts a tsm_recycle to TSM. The intention is for this register to be programmed to a TSM count value such that the TSM re-asserts its "rx_init" output, but there are no restrictions on programming this register. An RX recycle is a command from any protocol engine to TSM, requesting a jump from the TSM ON phase back to a programmable point in the WARMUP phase, and resume counting from there. A recycle will result from the reception of a packet with bad CRC or one which fails packet-filtering rules, or the end of a CCA operation in Continuous CCA mode which results in a channel indicating "busy". |

## 64.3.5.3.1.10 TSM FAST WARMUP CONTROL 1 (FAST_CTRL1)

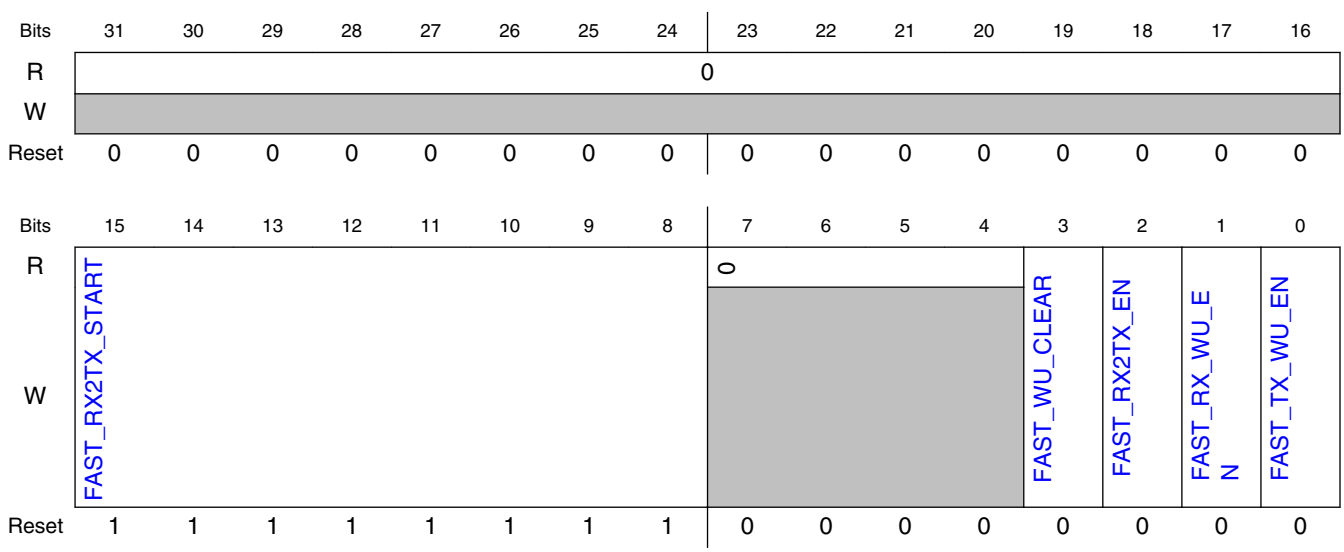
### *64.3.5.3.1.10.1 Offset*

| Register | Offset |
|---|---|
| FAST_CTRL1 | 28h |

### *64.3.5.3.1.10.2 Function*

This register provides enabling and control for Fast TSM Mode

### *64.3.5.3.1.10.3 Diagram*



#### *64.3.5.3.1.10.4 Fields*

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>FAST_RX2TX_START | TSM "Jump-to" point for a Fast TSM RX-to-TX Transition.<br>During a Fast RX-to-TX transition (FAST_RX2TX_EN=1) the contents of this register represent the entry point (TSM count) in the TX warmup sequence to which the TSM jumps to when 802.15.4 ZSM asserts fast_rx2tx_wu. The TSM will jump from the RX “ON” phase to the TX “WARMUP” phase, starting at this count. |
| 7-4<br>— | Reserved |
| 3<br>FAST_WU_CLEAR | Fast TSM Warmup Clear State<br>This bit clears the RF channel memory in the PLL Digital Block, forcing the next TSM TX Warmup to be normal (not fast), and the next TSM RX Warmup to be normal (not fast), regardless of whether the RF channel has actually changed or not. This bit is not self-clearing. Write '1' to clear channel memory, then write '0' to proceed with TSM operations. |
| 2<br>FAST_RX2TX_EN | Fast TSM RX-to-TX Transition Enable<br>Intended for 802.15.4 only, this bit enables a special state transition in the 802.15.4 ZSM (Sequence Manager), combined with a TSM jump from the RX "ON" phase to the TX "Warmup" phase, to accelerate the CCA-to-TX transition in the 802.15.4 Sequence T and Sequence TR autosequences<br>0b - Disable Fast RX-to-TX transitions<br>1b - Enable Fast RX-to-TX transitions (if fast_rx2tx_wu is asserted by 802.15.4 ZSM) |
| 1<br>FAST_RX_WU_EN | Fast TSM RX Warmup Enable<br>0b - Fast TSM RX Warmups are disabled<br>1b - Fast TSM RX Warmups are enabled, if the RF channel has not changed since the last RX warmup, and for BLE mode, the RF channel is not an advertising channel. |
| 0<br>FAST_TX_WU_EN | Fast TSM TX Warmup Enable<br>0b - Fast TSM TX Warmups are disabled<br>1b - Fast TSM TX Warmups are enabled, if the RF channel has not changed since the last TX warmup, and for BLE mode, the RF channel is not an advertising channel. |

## 64.3.5.3.1.11 TSM FAST WARMUP CONTROL 2 (FAST_CTRL2)

#### *64.3.5.3.1.11.1 Offset*

| Register | Offset |
|---|---|
| FAST_CTRL2 | 2Ch |

#### *64.3.5.3.1.11.2 Function*

This register provides configuration for Fast TSM Mode

### 64.3.5.3.1.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FAST_DEST_RX | | | | | | | | FAST_START_RX | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FAST_DEST_TX | | | | | | | | FAST_START_TX | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>FAST_DEST_RX | Fast TSM RX "Jump-to" Point<br>During a Fast RX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, FAST_START_RX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>FAST_START_RX = 10<br>FAST_DEST_RX = 15<br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| 23-16<br>FAST_START_RX | Fast TSM RX "Jump-from" Point<br>During a Fast RX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_RX[7:0], thereby executing the jump. |
| 15-8<br>FAST_DEST_TX | Fast TSM TX "Jump-to" Point<br>During a Fast TX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, FAST_START_TX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>FAST_START_TX = 10<br>FAST_DEST_TX = 15<br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| 7-0<br>FAST_START_TX | Fast TSM TX "Jump-from" Point<br>During a Fast TX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_TX[7:0], thereby executing the jump. |

## 64.3.5.3.1.12 TSM_TIMING00 (TIMING00)

#### 64.3.5.3.1.12.1 Offset

| Register | Offset |
|---|---|
| TIMING00 | 30h |

#### 64.3.5.3.1.12.2 Function

This register contains the timing values to control the assertion and deassertion times for both TX and RX sequences for the BB_LDO_HF_EN TSM signal or signal group.

#### 64.3.5.3.1.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_HF_EN_RX_LO | | | | | | | | BB_LDO_HF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_HF_EN_TX_LO | | | | | | | | BB_LDO_HF_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_HF_EN_RX_LO | De-assertion time setting for BB_LDO_HF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_HF_EN_RX_HI | Assertion time setting for BB_LDO_HF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_HF_EN_TX_LO | De-assertion time setting for BB_LDO_HF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_HF_EN_TX_HI | Assertion time setting for BB_LDO_HF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from LO to HI. |

### 64.3.5.3.1.13 TSM_TIMING01 (TIMING01)

#### 64.3.5.3.1.13.1 Offset

| Register | Offset |
|---|---|
| TIMING01 | 34h |

#### 64.3.5.3.1.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_ADCDAC_EN_RX_LO | | | | | | | | BB_LDO_ADCDAC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_ADCDAC_EN_TX_LO | | | | | | | | BB_LDO_ADCDAC_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_ADCDAC_EN_RX_LO | De-assertion time setting for BB_LDO_ADCDAC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_ADCDAC_EN_RX_HI | Assertion time setting for BB_LDO_ADCDAC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_ADCDAC_EN_TX_LO | De-assertion time setting for BB_LDO_ADCDAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_ADCDAC_EN_TX_HI | Assertion time setting for BB_LDO_ADCDAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from LO to HI. |

### 64.3.5.3.1.14 TSM_TIMING02 (TIMING02)

#### 64.3.5.3.1.14.1 Offset

| Register | Offset |
|---|---|
| TIMING02 | 38h |

#### 64.3.5.3.1.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_BBA_EN_RX_LO | | | | | | | | BB_LDO_BBA_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_BBA_EN_RX_LO | De-assertion time setting for BB_LDO_BBA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_BBA_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_BBA_EN_RX_HI | Assertion time setting for BB_LDO_BBA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_BBA_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

### 64.3.5.3.1.15 TSM_TIMING03 (TIMING03)

#### 64.3.5.3.1.15.1 Offset

| Register | Offset |
|---|---|
| TIMING03 | 3Ch |

#### 64.3.5.3.1.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_PD_EN_RX_LO | | | | | | | | BB_LDO_PD_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_PD_EN_TX_LO | | | | | | | | BB_LDO_PD_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.15.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_PD_EN_RX_LO | De-assertion time setting for BB_LDO_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_PD_EN_RX_HI | Assertion time setting for BB_LDO_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_PD_EN_TX_LO | De-assertion time setting for BB_LDO_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_PD_EN_TX_HI | Assertion time setting for BB_LDO_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.16 TSM_TIMING04 (TIMING04)

#### 64.3.5.3.1.16.1 Offset

| Register | Offset |
|---|---|
| TIMING04 | 40h |

#### 64.3.5.3.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_FDBK_EN_RX_LO | | | | | | | | BB_LDO_FDBK_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_FDBK_EN_TX_LO | | | | | | | | BB_LDO_FDBK_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.16.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_FDBK_EN_RX_LO | De-assertion time setting for BB_LDO_FDBK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_FDBK_EN_RX_HI | Assertion time setting for BB_LDO_FDBK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_FDBK_EN_TX_LO | De-assertion time setting for BB_LDO_FDBK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_FDBK_EN_TX_HI | Assertion time setting for BB_LDO_FDBK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.17 TSM_TIMING05 (TIMING05)

#### 64.3.5.3.1.17.1 Offset

| Register | Offset |
|---|---|
| TIMING05 | 44h |

#### 64.3.5.3.1.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VCOLO_EN_RX_LO | | | | | | | | BB_LDO_VCOLO_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VCOLO_EN_TX_LO | | | | | | | | BB_LDO_VCOLO_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.17.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VCOLO_EN_RX_HI | Assertion time setting for BB_LDO_VCOLO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VCOLO_EN_TX_LO | De-assertion time setting for BB_LDO_VCOLO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.18 TSM_TIMING06 (TIMING06)

#### 64.3.5.3.1.18.1 Offset

| Register | Offset |
|---|---|
| TIMING06 | 48h |

#### 64.3.5.3.1.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VTREF_EN_RX_LO | | | | | | | | BB_LDO_VTREF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VTREF_EN_TX_LO | | | | | | | | BB_LDO_VTREF_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.18.3 Fields

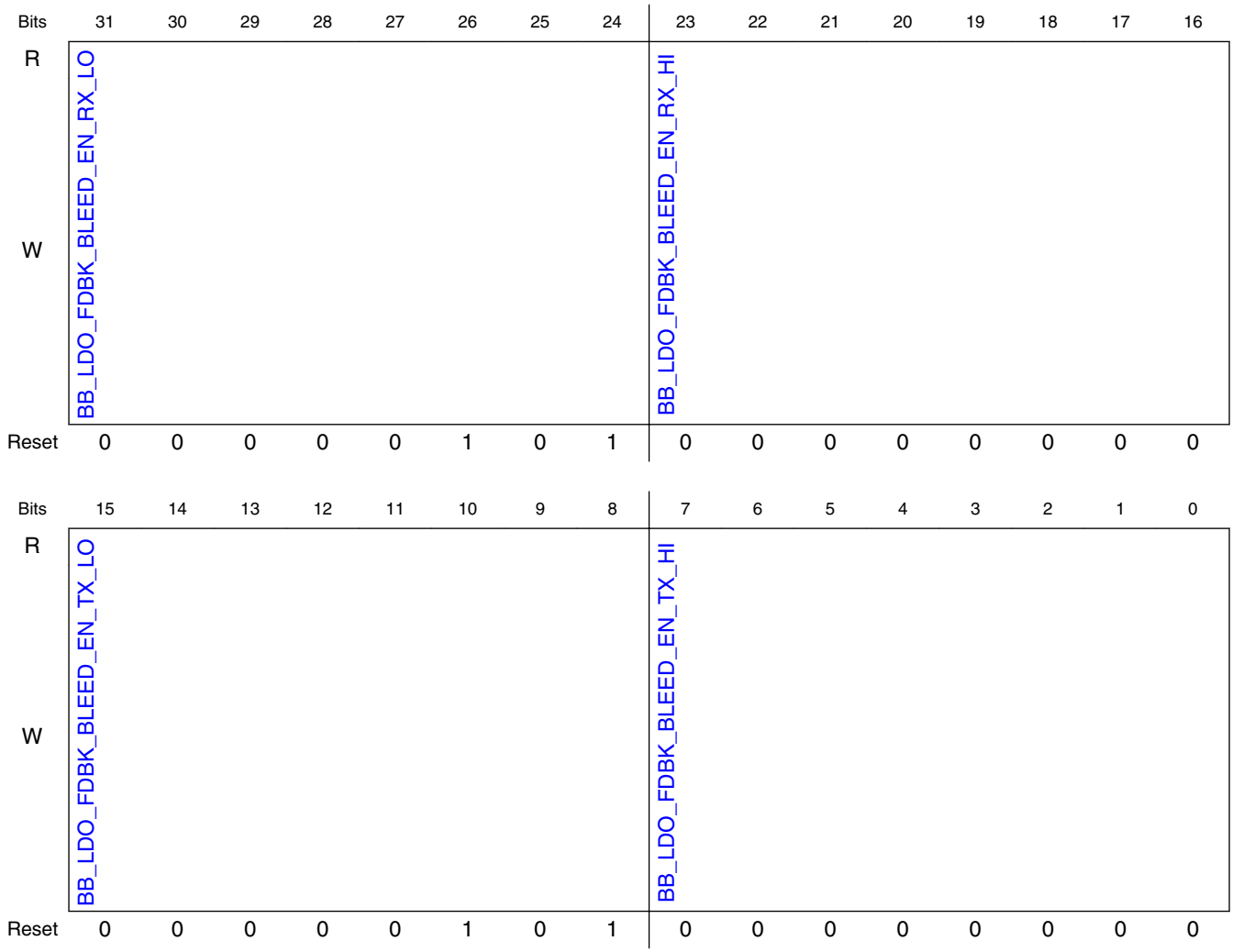
| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VTREF_EN_RX_LO | De-assertion time setting for BB_LDO_VTREF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VTREF_EN_RX_HI | Assertion time setting for BB_LDO_VTREF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VTREF_EN_TX_LO | De-assertion time setting for BB_LDO_VTREF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VTREF_EN_TX_HI | Assertion time setting for BB_LDO_VTREF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.19 TSM_TIMING07 (TIMING07)

#### 64.3.5.3.1.19.1 Offset

| Register | Offset |
|---|---|
| TIMING07 | 4Ch |

### 64.3.5.3.1.19.2 Diagram




### 64.3.5.3.1.19.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_FDBK_BLEED_EN_RX_LO | De-assertion time setting for BB_LDO_FDBK_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_FDBK_BLEED_EN_RX_HI | Assertion time setting for BB_LDO_FDBK_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_FDBK_BLEED_EN_TX_LO | De-assertion time setting for BB_LDO_FDBK_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

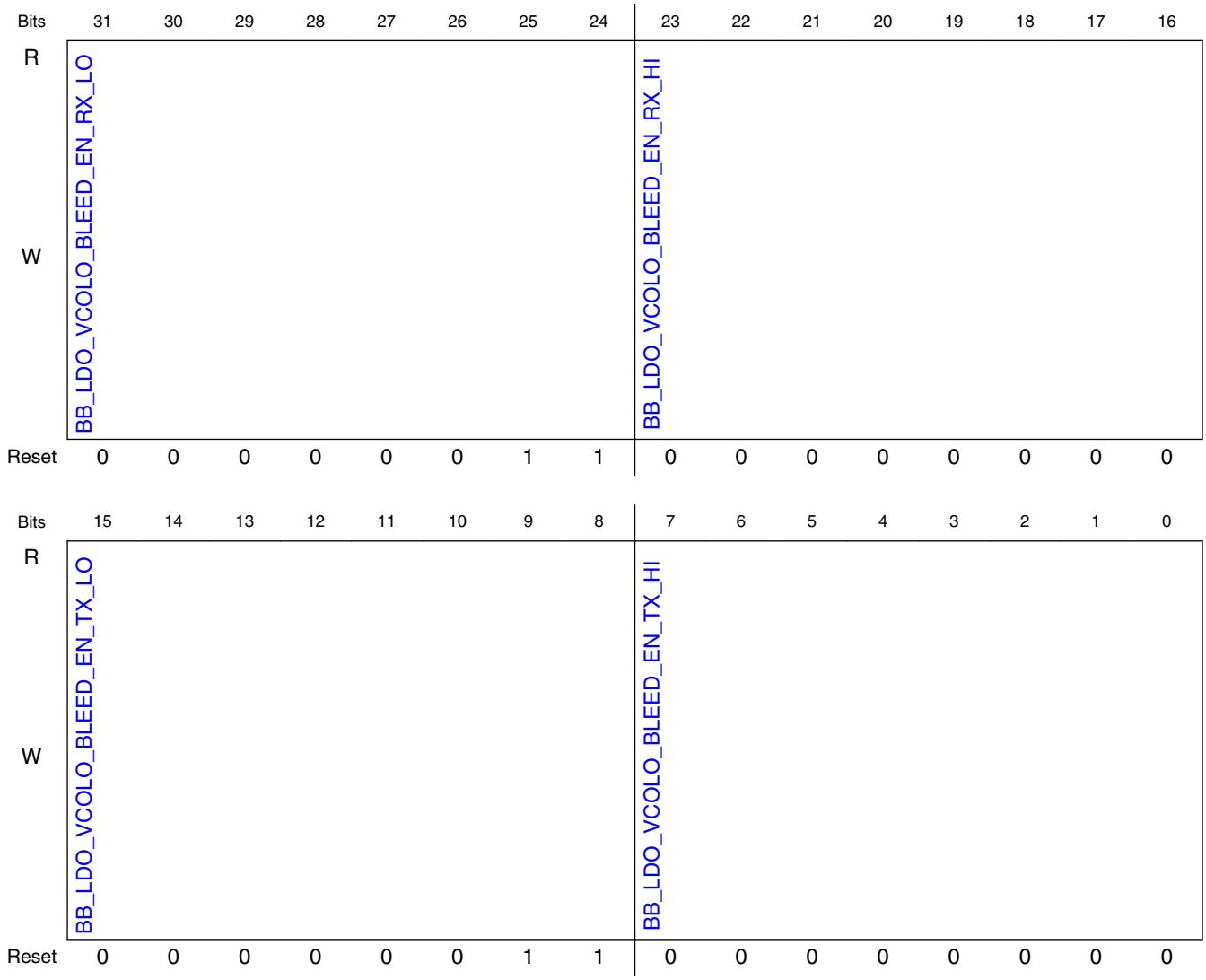
| Field | Function |
|---|---|
| 7-0<br>BB_LDO_FDBK_BLEED_EN_TX_HI | Assertion time setting for BB_LDO_FDBK_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.20 TSM_TIMING08 (TIMING08)

### *64.3.5.3.1.20.1 Offset*

| Register | Offset |
|---|---|
| TIMING08 | 50h |

### 64.3.5.3.1.20.2 Diagram




### 64.3.5.3.1.20.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_BLEED_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VCOLO_BLEED_EN_RX_HI | Assertion time setting for BB_LDO_VCOLO_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for BB_LDO_VCOLO_BLEED_EN (TX) |

*Table continues on the next page...*

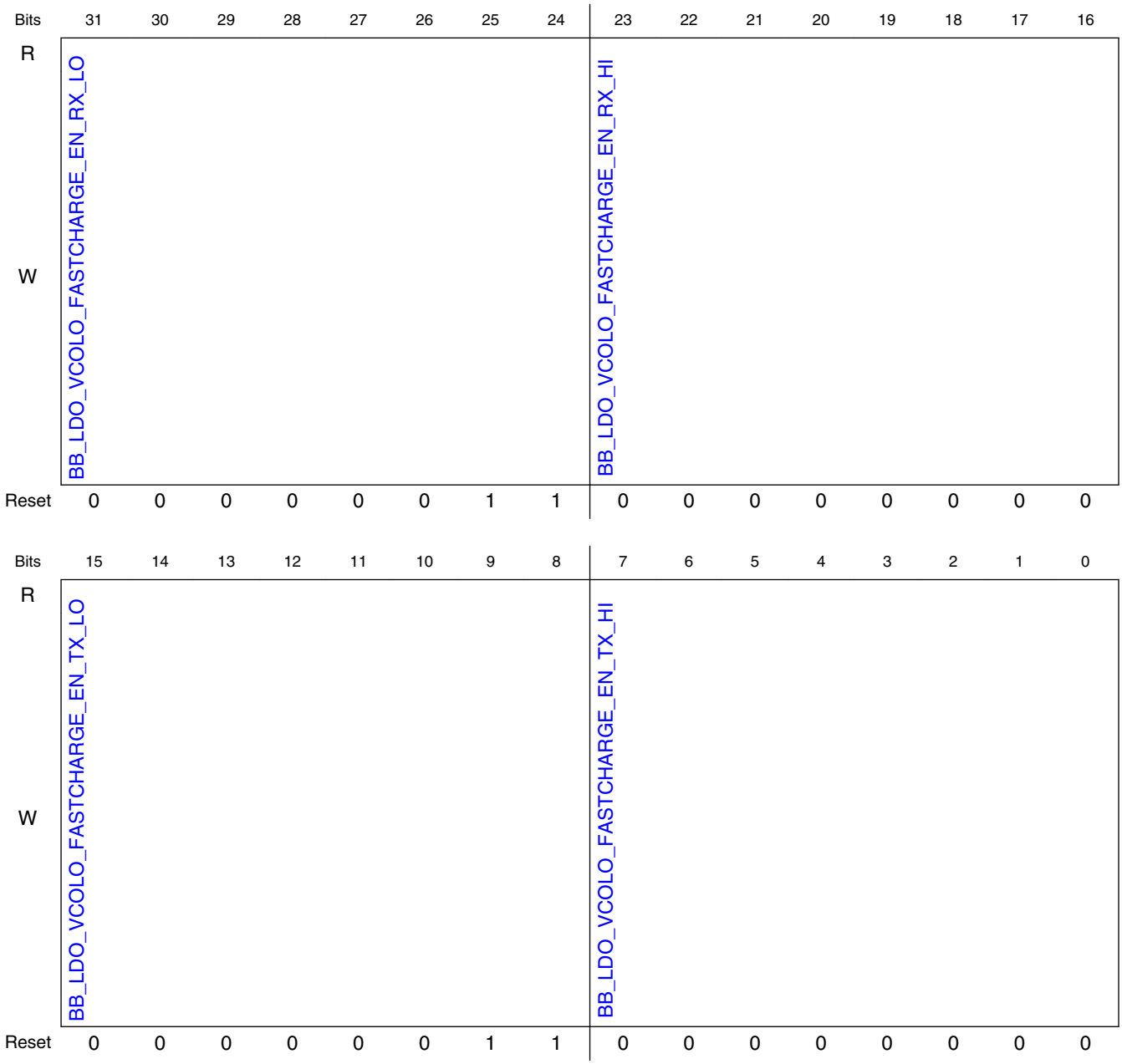
| Field | Function |
|---|---|
| BB_LDO_VCOLO_BLEED_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_BLEED_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.21 TSM_TIMING09 (TIMING09)

### *64.3.5.3.1.21.1 Offset*

| Register | Offset |
|---|---|
| TIMING09 | 54h |

### 64.3.5.3.1.21.2 Diagram




### 64.3.5.3.1.21.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_FASTCHARGE_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (RX) |

*Table continues on the next page...*

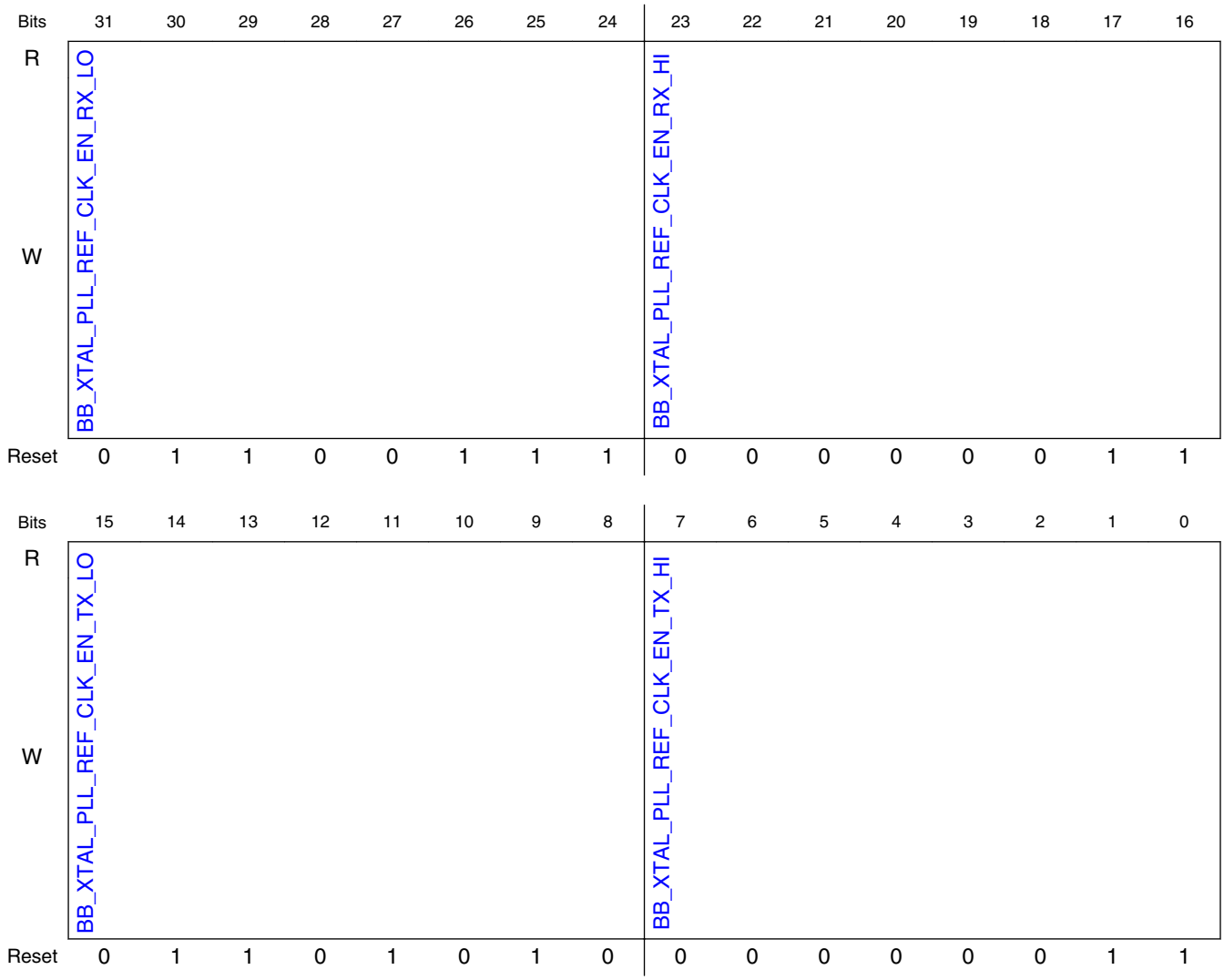
| Field | Function |
|---|---|
| BB_LDO_VCOLO_FASTCHARGE_EN_RX_HI | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VCOLO_FASTCHARGE_EN_TX_LO | De-assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_FASTCHARGE_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.22 TSM_TIMING10 (TIMING10)

### *64.3.5.3.1.22.1 Offset*

| Register | Offset |
|---|---|
| TIMING10 | 58h |

### 64.3.5.3.1.22.2 Diagram




### 64.3.5.3.1.22.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_XTAL_PLL_REF_CLK_EN_RX_LO | De-assertion time setting for BB_XTAL_PLL_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_XTAL_PLL_REF_CLK_EN_RX_HI | Assertion time setting for BB_XTAL_PLL_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for BB_XTAL_PLL_REF_CLK_EN (TX) |

*Table continues on the next page...*

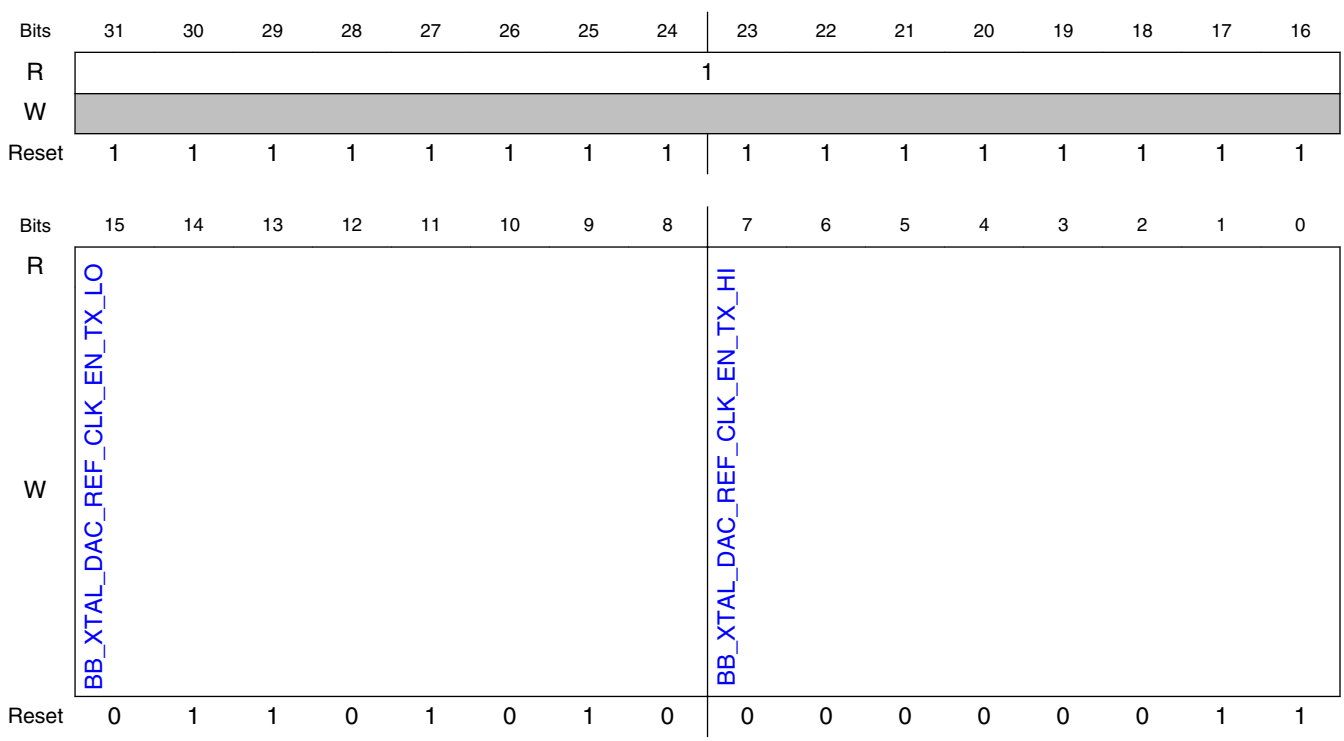
| Field | Function |
|---|---|
| BB_XTAL_PLL_REF_CLK_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_XTAL_PLL_REF_CLK_EN_TX_HI | Assertion time setting for BB_XTAL_PLL_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.23 TSM_TIMING11 (TIMING11)

### *64.3.5.3.1.23.1 Offset*

| Register | Offset |
|---|---|
| TIMING11 | 5Ch |

### *64.3.5.3.1.23.2 Diagram*



#### *64.3.5.3.1.23.3 Fields*

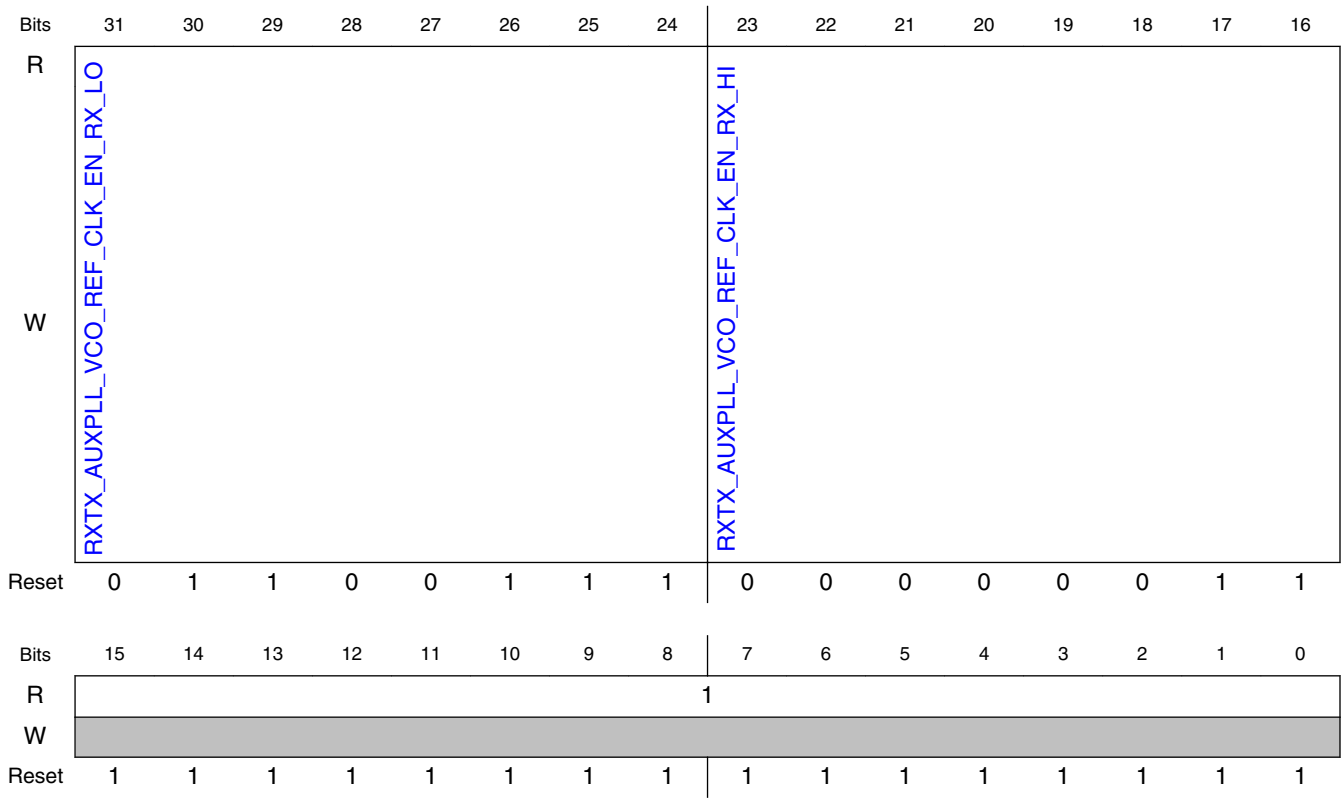
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>BB_XTAL_DAC_REF_CLK_EN_TX_LO | De-assertion time setting for BB_XTAL_DAC_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_DAC_REF_CLK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_XTAL_DAC_REF_CLK_EN_TX_HI | Assertion time setting for BB_XTAL_DAC_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_DAC_REF_CLK_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.24 TSM_TIMING12 (TIMING12)

#### *64.3.5.3.1.24.1 Offset*

| Register | Offset |
|---|---|
| TIMING12 | 60h |

### 64.3.5.3.1.24.2 Diagram




### 64.3.5.3.1.24.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_VCO_REF_CLK_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_VCO_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_VCO_REF_CLK_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_VCO_REF_CLK_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_VCO_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_VCO_REF_CLK_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.25 TSM_TIMING13 (TIMING13)

#### 64.3.5.3.1.25.1 Offset

| Register | Offset |
|---|---|
| TIMING13 | 64h |

#### 64.3.5.3.1.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_LOOP_IS_OPEN_RX_LO | | | | | | | | PLL_LOOP_IS_OPEN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_LOOP_IS_OPEN_TX_LO | | | | | | | | PLL_LOOP_IS_OPEN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.25.3 Fields

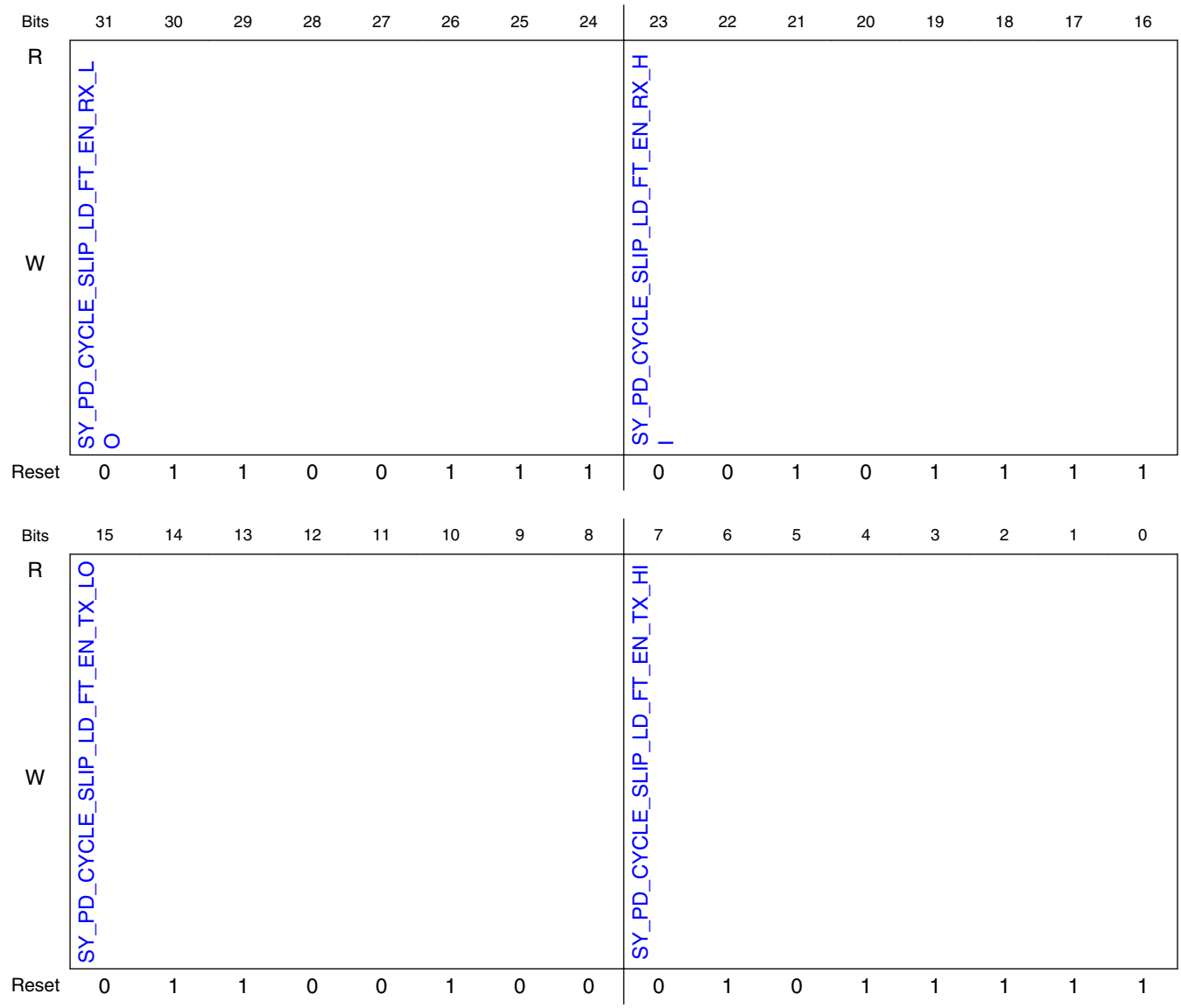
| Field | Function |
|---|---|
| 31-24<br>PLL_LOOP_IS_OPEN_RX_LO | De-assertion time setting for PLL_LOOP_IS_OPEN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from HI to LO. |
| 23-16<br>PLL_LOOP_IS_OPEN_RX_HI | Assertion time setting for PLL_LOOP_IS_OPEN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from LO to HI. |
| 15-8<br>PLL_LOOP_IS_OPEN_TX_LO | De-assertion time setting for PLL_LOOP_IS_OPEN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from HI to LO. |
| 7-0<br>PLL_LOOP_IS_OPEN_TX_HI | Assertion time setting for PLL_LOOP_IS_OPEN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from LO to HI. |

### 64.3.5.3.1.26 TSM_TIMING14 (TIMING14)

#### 64.3.5.3.1.26.1 Offset

| Register | Offset |
|---|---|
| TIMING14 | 68h |

### 64.3.5.3.1.26.2 Diagram




### 64.3.5.3.1.26.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_PD_CYCLE_SLIP_LD_FT_EN_RX_LO | De-assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from LO to HI. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_CYCLE_SLIP_LD_FT_EN_RX_HI | |
| 15-8<br>SY_PD_CYCLE_SLIP_LD_FT_EN_TX_LO | De-assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_PD_CYCLE_SLIP_LD_FT_EN_TX_HI | Assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.27 TSM_TIMING15 (TIMING15)

### *64.3.5.3.1.27.1 Offset*

| Register | Offset |
|---|---|
| TIMING15 | 6Ch |

### *64.3.5.3.1.27.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_VCO_EN_RX_LO | | | | | | | | SY_VCO_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_VCO_EN_TX_LO | | | | | | | | SY_VCO_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### *64.3.5.3.1.27.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>SY_VCO_EN_RX_LO | De-assertion time setting for SY_VCO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for SY_VCO_EN (RX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_VCO_EN_RX_HI | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_VCO_EN_TX_LO | De-assertion time setting for SY_VCO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_VCO_EN_TX_HI | Assertion time setting for SY_VCO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.28 TSM_TIMING16 (TIMING16)

### *64.3.5.3.1.28.1 Offset*

| Register | Offset |
|---|---|
| TIMING16 | 70h |

### *64.3.5.3.1.28.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_RX_BUF_EN_RX_LO | | | | | | | | SY_LO_RX_BUF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### *64.3.5.3.1.28.3 Fields*

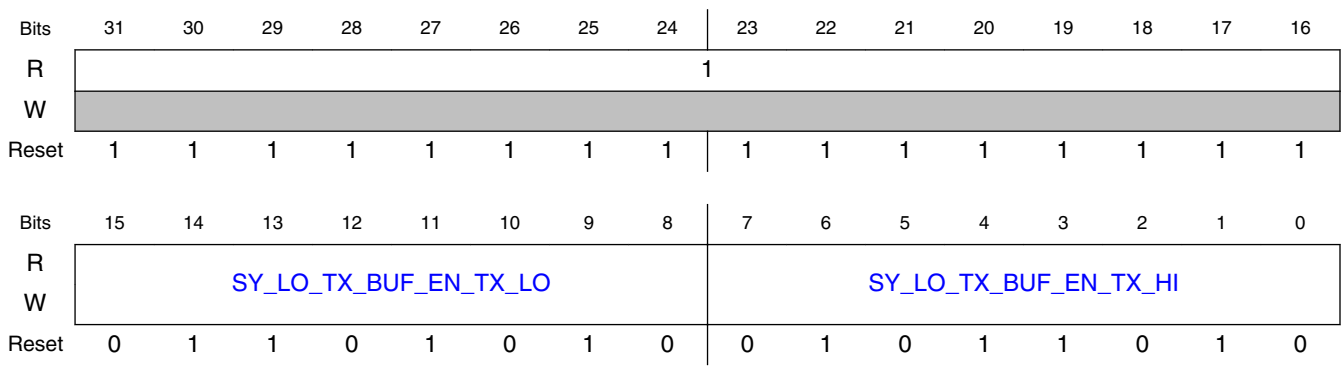
| Field | Function |
|---|---|
| 31-24<br>SY_LO_RX_BUF_EN_RX_LO | De-assertion time setting for SY_LO_RX_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_BUF_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_LO_RX_BUF_EN_RX_HI | Assertion time setting for SY_LO_RX_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_BUF_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.29 TSM_TIMING17 (TIMING17)

### *64.3.5.3.1.29.1 Offset*

| Register | Offset |
|---|---|
| TIMING17 | 74h |

### *64.3.5.3.1.29.2 Diagram*




### *64.3.5.3.1.29.3 Fields*

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>SY_LO_TX_BUF_EN_TX_LO | De-assertion time setting for SY_LO_TX_BUF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_BUF_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_LO_TX_BUF_EN_TX_HI | Assertion time setting for SY_LO_TX_BUF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_BUF_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.30 TSM_TIMING18 (TIMING18)

### *64.3.5.3.1.30.1 Offset*

| Register | Offset |
|---|---|
| TIMING18 | 78h |

#### 64.3.5.3.1.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_EN_RX_LO | | | | | | | | SY_DIVN_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_EN_TX_LO | | | | | | | | SY_DIVN_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

#### 64.3.5.3.1.30.3 Fields

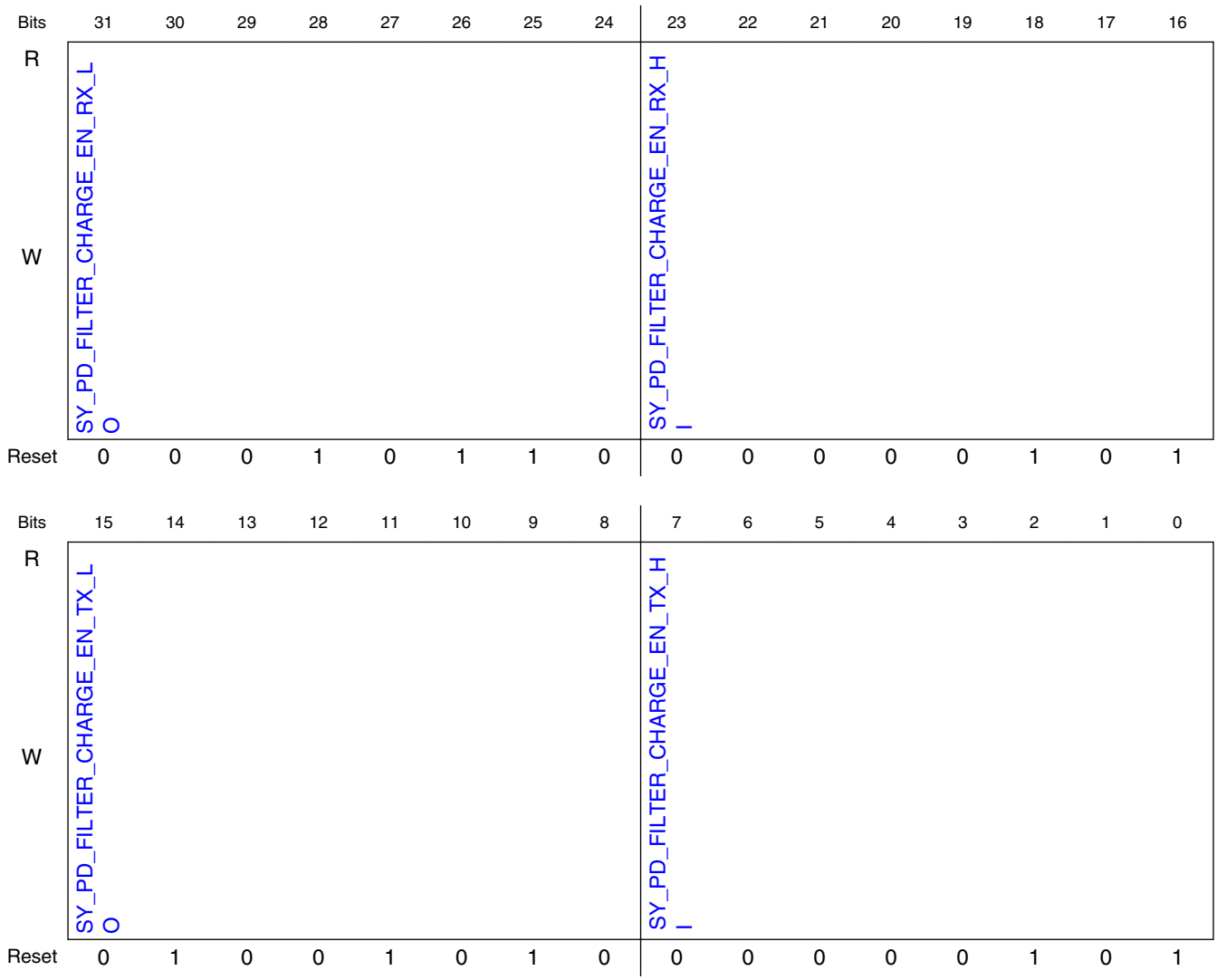
| Field | Function |
|---|---|
| 31-24<br>SY_DIVN_EN_RX_LO | De-assertion time setting for SY_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_DIVN_EN_RX_HI | Assertion time setting for SY_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_DIVN_EN_TX_LO | De-assertion time setting for SY_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_DIVN_EN_TX_HI | Assertion time setting for SY_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.31 TSM_TIMING19 (TIMING19)

#### 64.3.5.3.1.31.1 Offset

| Register | Offset |
|---|---|
| TIMING19 | 7Ch |

### 64.3.5.3.1.31.2 Diagram




### 64.3.5.3.1.31.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_PD_FILTER_CHARGE_EN_RX_LO | De-assertion time setting for SY_PD_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_PD_FILTER_CHARGE_EN_RX_HI | Assertion time setting for SY_PD_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for SY_PD_FILTER_CHARGE_EN (TX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_FILTER_CHARGE_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_PD_FILTER_CHARGE_EN_TX_HI | Assertion time setting for SY_PD_FILTER_CHARGE_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.32 TSM_TIMING20 (TIMING20)

### *64.3.5.3.1.32.1 Offset*

| Register | Offset |
|---|---|
| TIMING20 | 80h |

### *64.3.5.3.1.32.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_PD_EN_RX_LO | | | | | | | | SY_PD_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_PD_EN_TX_LO | | | | | | | | SY_PD_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

### *64.3.5.3.1.32.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>SY_PD_EN_RX_LO | De-assertion time setting for SY_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_PD_EN_RX_HI | Assertion time setting for SY_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_PD_EN_TX_LO | De-assertion time setting for SY_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-0<br>SY_PD_EN_TX_HI | Assertion time setting for SY_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.33 TSM_TIMING21 (TIMING21)

### *64.3.5.3.1.33.1 Offset*

| Register | Offset |
|---|---|
| TIMING21 | 84h |

### *64.3.5.3.1.33.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_DIVN_EN_RX_LO | | | | | | | | SY_LO_DIVN_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_DIVN_EN_TX_LO | | | | | | | | SY_LO_DIVN_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### *64.3.5.3.1.33.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>SY_LO_DIVN_EN_RX_LO | De-assertion time setting for SY_LO_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_LO_DIVN_EN_RX_HI | Assertion time setting for SY_LO_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_LO_DIVN_EN_TX_LO | De-assertion time setting for SY_LO_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_LO_DIVN_EN_TX_HI | Assertion time setting for SY_LO_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.34 TSM_TIMING22 (TIMING22)

### *64.3.5.3.1.34.1 Offset*

| Register | Offset |
|---|---|
| TIMING22 | 88h |

### *64.3.5.3.1.34.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_RX_EN_RX_LO | | | | | | | | SY_LO_RX_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### *64.3.5.3.1.34.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>SY_LO_RX_EN_RX_LO | De-assertion time setting for SY_LO_RX_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_LO_RX_EN_RX_HI | Assertion time setting for SY_LO_RX_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.35 TSM_TIMING23 (TIMING23)

### *64.3.5.3.1.35.1 Offset*

| Register | Offset |
|---|---|
| TIMING23 | 8Ch |

#### 64.3.5.3.1.35.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SY_LO_TX_EN_TX_LO | | | | | | | | SY_LO_TX_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

#### 64.3.5.3.1.35.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>SY_LO_TX_EN_TX_LO | De-assertion time setting for SY_LO_TX_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_LO_TX_EN_TX_HI | Assertion time setting for SY_LO_TX_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.36 TSM_TIMING24 (TIMING24)

#### 64.3.5.3.1.36.1 Offset

| Register | Offset |
|---|---|
| TIMING24 | 90h |

#### 64.3.5.3.1.36.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_CAL_EN_RX_LO | | | | | | | | SY_DIVN_CAL_EN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_CAL_EN_TX_LO | | | | | | | | SY_DIVN_CAL_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.36.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_DIVN_CAL_EN_RX_LO | De-assertion time setting for SY_DIVN_CAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_DIVN_CAL_EN_RX_HI | Assertion time setting for SY_DIVN_CAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_DIVN_CAL_EN_TX_LO | De-assertion time setting for SY_DIVN_CAL_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_DIVN_CAL_EN_TX_HI | Assertion time setting for SY_DIVN_CAL_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.37 TSM_TIMING25 (TIMING25)

#### 64.3.5.3.1.37.1 Offset

| Register | Offset |
|---|---|
| TIMING25 | 94h |

#### 64.3.5.3.1.37.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_LNA_MIXER_EN_RX_LO | | | | | | | | RX_LNA_MIXER_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.37.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_LNA_MIXER_EN_RX_LO | De-assertion time setting for RX_LNA_MIXER_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_LNA_MIXER_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_LNA_MIXER_EN_RX_HI | Assertion time setting for RX_LNA_MIXER_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_LNA_MIXER_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.38 TSM_TIMING26 (TIMING26)

#### 64.3.5.3.1.38.1 Offset

| Register | Offset |
|---|---|
| TIMING26 | 98h |

#### 64.3.5.3.1.38.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_PA_EN_TX_LO | | | | | | | | TX_PA_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

#### 64.3.5.3.1.38.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_PA_EN_TX_LO | De-assertion time setting for TX_PA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_PA_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_PA_EN_TX_HI | Assertion time setting for TX_PA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_PA_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.39 TSM_TIMING27 (TIMING27)

#### 64.3.5.3.1.39.1 Offset

| Register | Offset |
|---|---|
| TIMING27 | 9Ch |

#### 64.3.5.3.1.39.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_ADC_I_Q_EN_RX_LO | | | | | | | | RX_ADC_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.39.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_ADC_I_Q_EN_RX_LO | De-assertion time setting for RX_ADC_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_ADC_I_Q_EN_RX_HI | Assertion time setting for RX_ADC_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.40 TSM_TIMING28 (TIMING28)

#### 64.3.5.3.1.40.1 Offset

| Register | Offset |
|---|---|
| TIMING28 | A0h |

#### 64.3.5.3.1.40.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_ADC_RESET_EN_RX_LO | | | | | | | | RX_ADC_RESET_EN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.40.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_ADC_RESET_EN_RX_LO | De-assertion time setting for RX_ADC_RESET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_RESET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_ADC_RESET_EN_RX_HI | Assertion time setting for RX_ADC_RESET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_RESET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.41 TSM_TIMING29 (TIMING29)

#### 64.3.5.3.1.41.1 Offset

| Register | Offset |
|---|---|
| TIMING29 | A4h |

#### 64.3.5.3.1.41.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_I_Q_EN_RX_LO | | | | | | | | RX_BBA_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.41.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_I_Q_EN_RX_LO | De-assertion time setting for RX_BBA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_I_Q_EN_RX_HI | Assertion time setting for RX_BBA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.42 TSM_TIMING30 (TIMING30)

#### 64.3.5.3.1.42.1 Offset

| Register | Offset |
|---|---|
| TIMING30 | A8h |

#### 64.3.5.3.1.42.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_PDET_EN_RX_LO | | | | | | | | RX_BBA_PDET_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.42.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_PDET_EN_RX_LO | De-assertion time setting for RX_BBA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_PDET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_PDET_EN_RX_HI | Assertion time setting for RX_BBA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_PDET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.43 TSM_TIMING31 (TIMING31)

#### 64.3.5.3.1.43.1 Offset

| Register | Offset |
|---|---|
| TIMING31 | ACh |

#### 64.3.5.3.1.43.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_TZA_DCOC_EN_RX_LO | | | | | | | | RX_BBA_TZA_DCOC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.43.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_TZA_DCOC_EN_RX_LO | De-assertion time setting for RX_BBA_TZA_DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_TZA_DCOC_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_TZA_DCOC_EN_RX_HI | Assertion time setting for RX_BBA_TZA_DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_TZA_DCOC_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.44 TSM_TIMING32 (TIMING32)

#### 64.3.5.3.1.44.1 Offset

| Register | Offset |
|---|---|
| TIMING32 | B0h |

#### 64.3.5.3.1.44.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_TZA_I_Q_EN_RX_LO | | | | | | | | RX_TZA_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.44.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_TZA_I_Q_EN_RX_LO | De-assertion time setting for RX_TZA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_TZA_I_Q_EN_RX_HI | Assertion time setting for RX_TZA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.45 TSM_TIMING33 (TIMING33)

#### 64.3.5.3.1.45.1 Offset

| Register | Offset |
|---|---|
| TIMING33 | B4h |

### 64.3.5.3.1.45.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_TZA_PDET_EN_RX_LO | | | | | | | | RX_TZA_PDET_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.45.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_TZA_PDET_EN_RX_LO | De-assertion time setting for RX_TZA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_PDET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_TZA_PDET_EN_RX_HI | Assertion time setting for RX_TZA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_PDET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.46 TSM_TIMING34 (TIMING34)

### 64.3.5.3.1.46.1 Offset

| Register | Offset |
|---|---|
| TIMING34 | B8h |

#### 64.3.5.3.1.46.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_DIG_EN_RX_LO | | | | | | | | PLL_DIG_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_DIG_EN_TX_LO | | | | | | | | PLL_DIG_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

#### 64.3.5.3.1.46.3 Fields

| Field | Function |
|---|---|
| 31-24<br>PLL_DIG_EN_RX_LO | De-assertion time setting for PLL_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from HI to LO. |
| 23-16<br>PLL_DIG_EN_RX_HI | Assertion time setting for PLL_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from LO to HI. |
| 15-8<br>PLL_DIG_EN_TX_LO | De-assertion time setting for PLL_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from HI to LO. |
| 7-0<br>PLL_DIG_EN_TX_HI | Assertion time setting for PLL_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.47 TSM_TIMING35 (TIMING35)

#### 64.3.5.3.1.47.1 Offset

| Register | Offset |
|---|---|
| TIMING35 | BCh |

### 64.3.5.3.1.47.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_DIG_EN_TX_LO | | | | | | | | TX_DIG_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

### 64.3.5.3.1.47.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_DIG_EN_TX_LO | De-assertion time setting for TX_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_DIG_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_DIG_EN_TX_HI | Assertion time setting for TX_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_DIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.48 TSM_TIMING36 (TIMING36)

### 64.3.5.3.1.48.1 Offset

| Register | Offset |
|---|---|
| TIMING36 | C0h |

#### 64.3.5.3.1.48.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_DIG_EN_RX_LO | | | | | | | | RX_DIG_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.48.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_DIG_EN_RX_LO | De-assertion time setting for RX_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_DIG_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_DIG_EN_RX_HI | Assertion time setting for RX_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_DIG_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.49 TSM_TIMING37 (TIMING37)

#### 64.3.5.3.1.49.1 Offset

| Register | Offset |
|---|---|
| TIMING37 | C4h |

### 64.3.5.3.1.49.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_INIT_RX_LO | | | | | | | | RX_INIT_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.49.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_INIT_RX_LO | De-assertion time setting for RX_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_INIT signal or group will transition from HI to LO. |
| 23-16<br>RX_INIT_RX_HI | Assertion time setting for RX_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_INIT signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.50 TSM_TIMING38 (TIMING38)

### 64.3.5.3.1.50.1 Offset

| Register | Offset |
|---|---|
| TIMING38 | C8h |

#### 64.3.5.3.1.50.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SIGMA_DELTA_EN_RX_LO | | | | | | | | SIGMA_DELTA_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SIGMA_DELTA_EN_TX_LO | | | | | | | | SIGMA_DELTA_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.5.3.1.50.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SIGMA_DELTA_EN_RX_LO | De-assertion time setting for SIGMA_DELTA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from HI to LO. |
| 23-16<br>SIGMA_DELTA_EN_RX_HI | Assertion time setting for SIGMA_DELTA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from LO to HI. |
| 15-8<br>SIGMA_DELTA_EN_TX_LO | De-assertion time setting for SIGMA_DELTA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from HI to LO. |
| 7-0<br>SIGMA_DELTA_EN_TX_HI | Assertion time setting for SIGMA_DELTA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.51 TSM_TIMING39 (TIMING39)

#### 64.3.5.3.1.51.1 Offset

| Register | Offset |
|---|---|
| TIMING39 | CCh |

### 64.3.5.3.1.51.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_PHY_EN_RX_LO | | | | | | | | RX_PHY_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.51.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_PHY_EN_RX_LO | De-assertion time setting for RX_PHY_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_PHY_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_PHY_EN_RX_HI | Assertion time setting for RX_PHY_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_PHY_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.52 TSM_TIMING40 (TIMING40)

### 64.3.5.3.1.52.1 Offset

| Register | Offset |
|---|---|
| TIMING40 | D0h |

#### 64.3.5.3.1.52.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCOC_EN_RX_LO | | | | | | | | DCOC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.52.3 Fields

| Field | Function |
|---|---|
| 31-24<br>DCOC_EN_RX_LO | De-assertion time setting for DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_EN signal or group will transition from HI to LO. |
| 23-16<br>DCOC_EN_RX_HI | Assertion time setting for DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

### 64.3.5.3.1.53 TSM_TIMING41 (TIMING41)

#### 64.3.5.3.1.53.1 Offset

| Register | Offset |
|---|---|
| TIMING41 | D4h |

### 64.3.5.3.1.53.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCOC_INIT_RX_LO | | | | | | | | DCOC_INIT_RX_HI | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.53.3 Fields

| Field | Function |
|---|---|
| 31-24<br>DCOC_INIT_RX_LO | De-assertion time setting for DCOC_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_INIT signal or group will transition from HI to LO. |
| 23-16<br>DCOC_INIT_RX_HI | Assertion time setting for DCOC_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_INIT signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.54 TSM_TIMING42 (TIMING42)

### 64.3.5.3.1.54.1 Offset

| Register | Offset |
|---|---|
| TIMING42 | D8h |

#### 64.3.5.3.1.54.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAR_ADC_TRIG_EN_RX_LO | | | | | | | | SAR_ADC_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAR_ADC_TRIG_EN_TX_LO | | | | | | | | SAR_ADC_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.54.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SAR_ADC_TRIG_EN_RX_LO | De-assertion time setting for SAR_ADC_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>SAR_ADC_TRIG_EN_RX_HI | Assertion time setting for SAR_ADC_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>SAR_ADC_TRIG_EN_TX_LO | De-assertion time setting for SAR_ADC_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>SAR_ADC_TRIG_EN_TX_HI | Assertion time setting for SAR_ADC_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.55 TSM_TIMING43 (TIMING43)

#### 64.3.5.3.1.55.1 Offset

| Register | Offset |
|---|---|
| TIMING43 | DCh |

#### 64.3.5.3.1.55.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE0_EN_RX_LO | | | | | | | | TSM_SPARE0_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE0_EN_TX_LO | | | | | | | | TSM_SPARE0_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.55.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE0_EN_RX_LO | De-assertion time setting for TSM_SPARE0_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE0_EN_RX_HI | Assertion time setting for TSM_SPARE0_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE0_EN_TX_LO | De-assertion time setting for TSM_SPARE0_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE0_EN_TX_HI | Assertion time setting for TSM_SPARE0_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.56 TSM_TIMING44 (TIMING44)

#### 64.3.5.3.1.56.1 Offset

| Register | Offset |
|---|---|
| TIMING44 | E0h |

#### 64.3.5.3.1.56.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE1_EN_RX_LO | | | | | | | | TSM_SPARE1_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE1_EN_TX_LO | | | | | | | | TSM_SPARE1_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.56.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE1_EN_RX_LO | De-assertion time setting for TSM_SPARE1_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE1_EN_RX_HI | Assertion time setting for TSM_SPARE1_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE1_EN_TX_LO | De-assertion time setting for TSM_SPARE1_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE1_EN_TX_HI | Assertion time setting for TSM_SPARE1_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.57 TSM_TIMING45 (TIMING45)

#### 64.3.5.3.1.57.1 Offset

| Register | Offset |
|---|---|
| TIMING45 | E4h |

### 64.3.5.3.1.57.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE2_EN_RX_LO | | | | | | | | TSM_SPARE2_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE2_EN_TX_LO | | | | | | | | TSM_SPARE2_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 64.3.5.3.1.57.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE2_EN_RX_LO | De-assertion time setting for TSM_SPARE2_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE2_EN_RX_HI | Assertion time setting for TSM_SPARE2_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE2_EN_TX_LO | De-assertion time setting for TSM_SPARE2_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE2_EN_TX_HI | Assertion time setting for TSM_SPARE2_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.58 TSM_TIMING46 (TIMING46)

### 64.3.5.3.1.58.1 Offset

| Register | Offset |
|---|---|
| TIMING46 | E8h |

#### 64.3.5.3.1.58.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE3_EN_RX_LO | | | | | | | | TSM_SPARE3_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE3_EN_TX_LO | | | | | | | | TSM_SPARE3_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.58.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE3_EN_RX_LO | De-assertion time setting for TSM_SPARE3_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE3_EN_RX_HI | Assertion time setting for TSM_SPARE3_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE3_EN_TX_LO | De-assertion time setting for TSM_SPARE3_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE3_EN_TX_HI | Assertion time setting for TSM_SPARE3_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.59 TSM_TIMING47 (TIMING47)

#### 64.3.5.3.1.59.1 Offset

| Register | Offset |
|---|---|
| TIMING47 | ECh |

#### 64.3.5.3.1.59.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO0_TRIG_EN_RX_LO | | | | | | | | GPIO0_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO0_TRIG_EN_TX_LO | | | | | | | | GPIO0_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.59.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO0_TRIG_EN_RX_LO | De-assertion time setting for GPIO0_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO0_TRIG_EN_RX_HI | Assertion time setting for GPIO0_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO0_TRIG_EN_TX_LO | De-assertion time setting for GPIO0_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO0_TRIG_EN_TX_HI | Assertion time setting for GPIO0_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.60 TSM_TIMING48 (TIMING48)

#### 64.3.5.3.1.60.1 Offset

| Register | Offset |
|---|---|
| TIMING48 | F0h |

#### 64.3.5.3.1.60.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO1_TRIG_EN_RX_LO | | | | | | | | GPIO1_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO1_TRIG_EN_TX_LO | | | | | | | | GPIO1_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.60.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO1_TRIG_EN_RX_LO | De-assertion time setting for GPIO1_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO1_TRIG_EN_RX_HI | Assertion time setting for GPIO1_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO1_TRIG_EN_TX_LO | De-assertion time setting for GPIO1_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO1_TRIG_EN_TX_HI | Assertion time setting for GPIO1_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.61 TSM_TIMING49 (TIMING49)

#### 64.3.5.3.1.61.1 Offset

| Register | Offset |
|---|---|
| TIMING49 | F4h |

#### 64.3.5.3.1.61.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO2_TRIG_EN_RX_LO | | | | | | | | GPIO2_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO2_TRIG_EN_TX_LO | | | | | | | | GPIO2_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.61.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO2_TRIG_EN_RX_LO | De-assertion time setting for GPIO2_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO2_TRIG_EN_RX_HI | Assertion time setting for GPIO2_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO2_TRIG_EN_TX_LO | De-assertion time setting for GPIO2_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO2_TRIG_EN_TX_HI | Assertion time setting for GPIO2_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.62 TSM_TIMING50 (TIMING50)

#### 64.3.5.3.1.62.1 Offset

| Register | Offset |
|---|---|
| TIMING50 | F8h |

#### 64.3.5.3.1.62.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO3_TRIG_EN_RX_LO | | | | | | | | GPIO3_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO3_TRIG_EN_TX_LO | | | | | | | | GPIO3_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.3.5.3.1.62.3 Fields

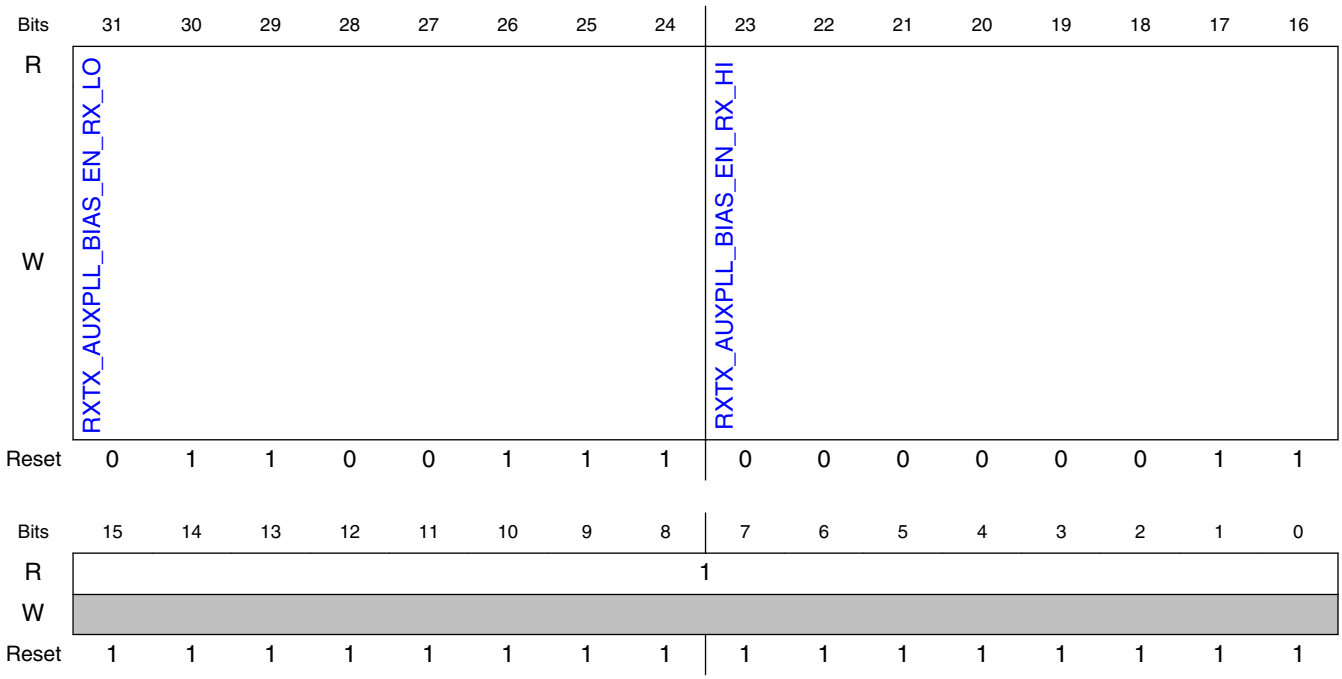
| Field | Function |
|---|---|
| 31-24<br>GPIO3_TRIG_EN_RX_LO | De-assertion time setting for GPIO3_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO3_TRIG_EN_RX_HI | Assertion time setting for GPIO3_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO3_TRIG_EN_TX_LO | De-assertion time setting for GPIO3_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO3_TRIG_EN_TX_HI | Assertion time setting for GPIO3_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.63 TSM_TIMING51 (TIMING51)

#### 64.3.5.3.1.63.1 Offset

| Register | Offset |
|---|---|
| TIMING51 | FCh |

#### 64.3.5.3.1.63.2 Diagram




#### 64.3.5.3.1.63.3 Fields

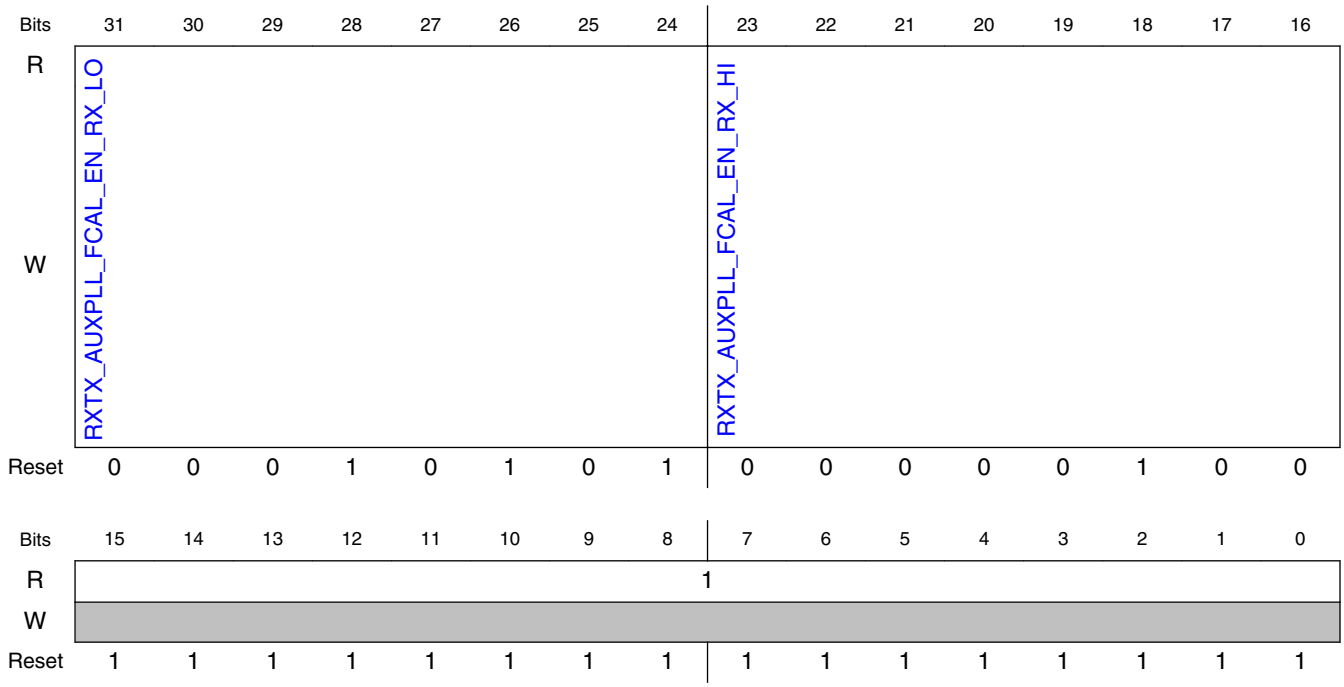
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_BIAS_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_BIAS_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_BIAS_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_BIAS_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_BIAS_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_BIAS_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.64 TSM_TIMING52 (TIMING52)

#### 64.3.5.3.1.64.1 Offset

| Register | Offset |
|---|---|
| TIMING52 | 100h |

#### 64.3.5.3.1.64.2 Diagram




#### 64.3.5.3.1.64.3 Fields

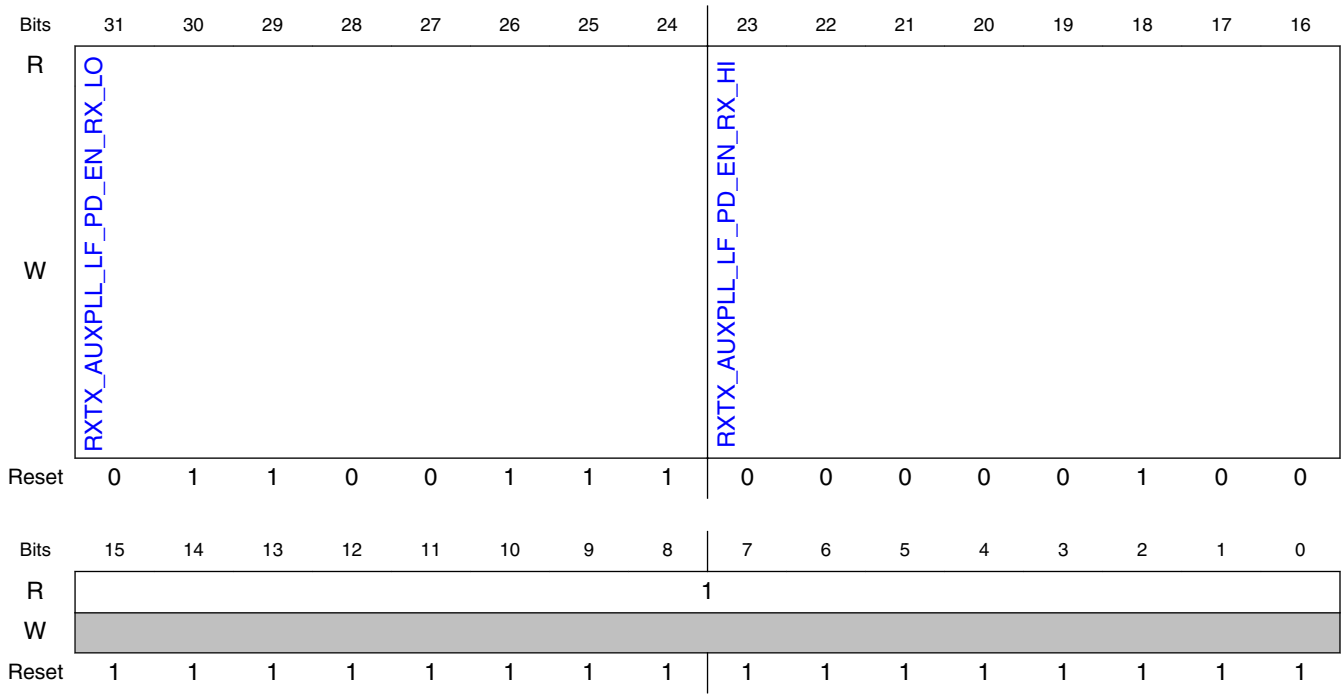
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_FCAL_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_FCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_FCAL_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_FCAL_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_FCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_FCAL_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.65 TSM_TIMING53 (TIMING53)

#### 64.3.5.3.1.65.1 Offset

| Register | Offset |
|---|---|
| TIMING53 | 104h |

### 64.3.5.3.1.65.2 Diagram




### 64.3.5.3.1.65.3 Fields

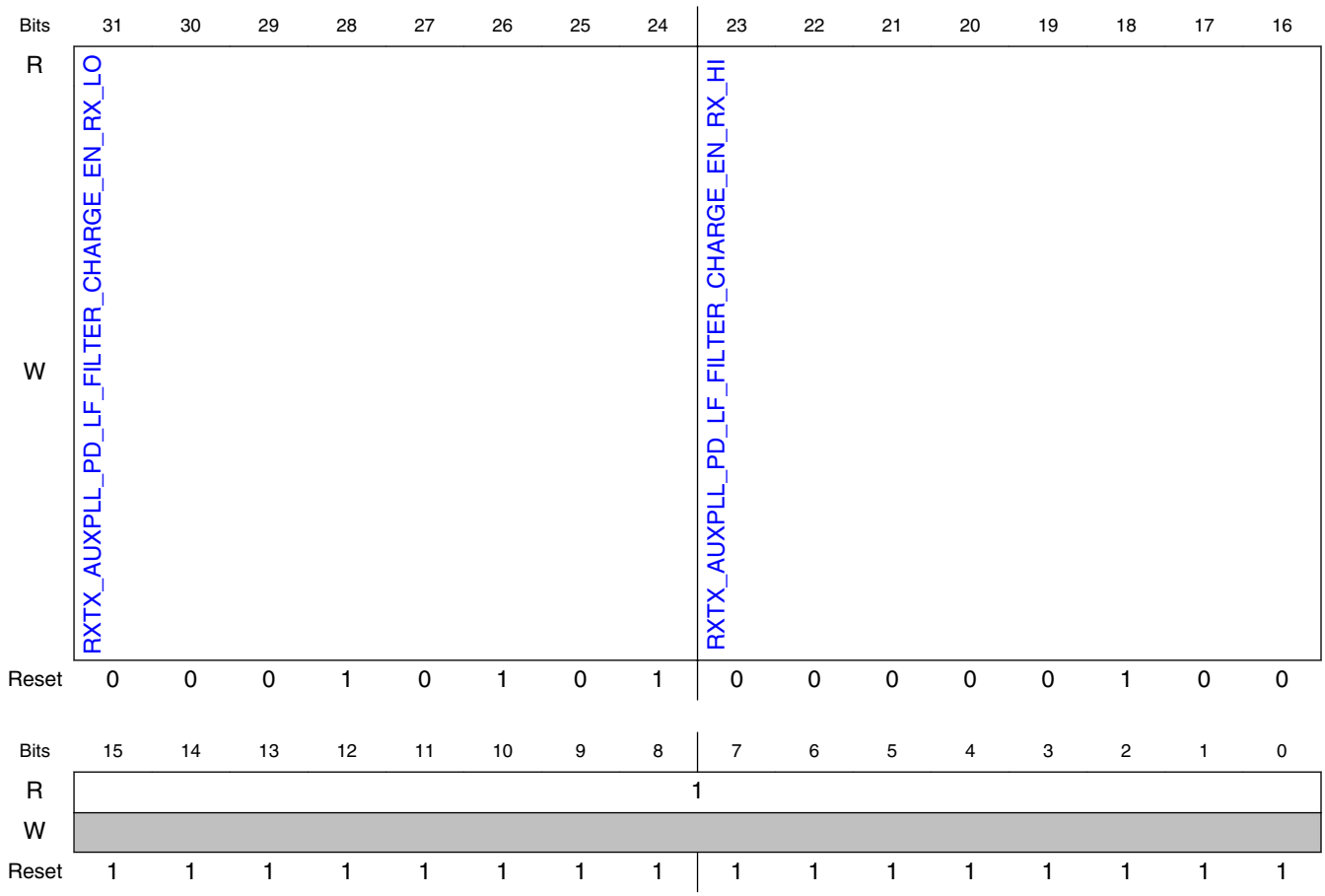
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_LF_PD_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_LF_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_LF_PD_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_LF_PD_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_LF_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_LF_PD_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.66 TSM_TIMING54 (TIMING54)

### 64.3.5.3.1.66.1 Offset

| Register | Offset |
|---|---|
| TIMING54 | 108h |

### 64.3.5.3.1.66.2 Diagram




### 64.3.5.3.1.66.3 Fields

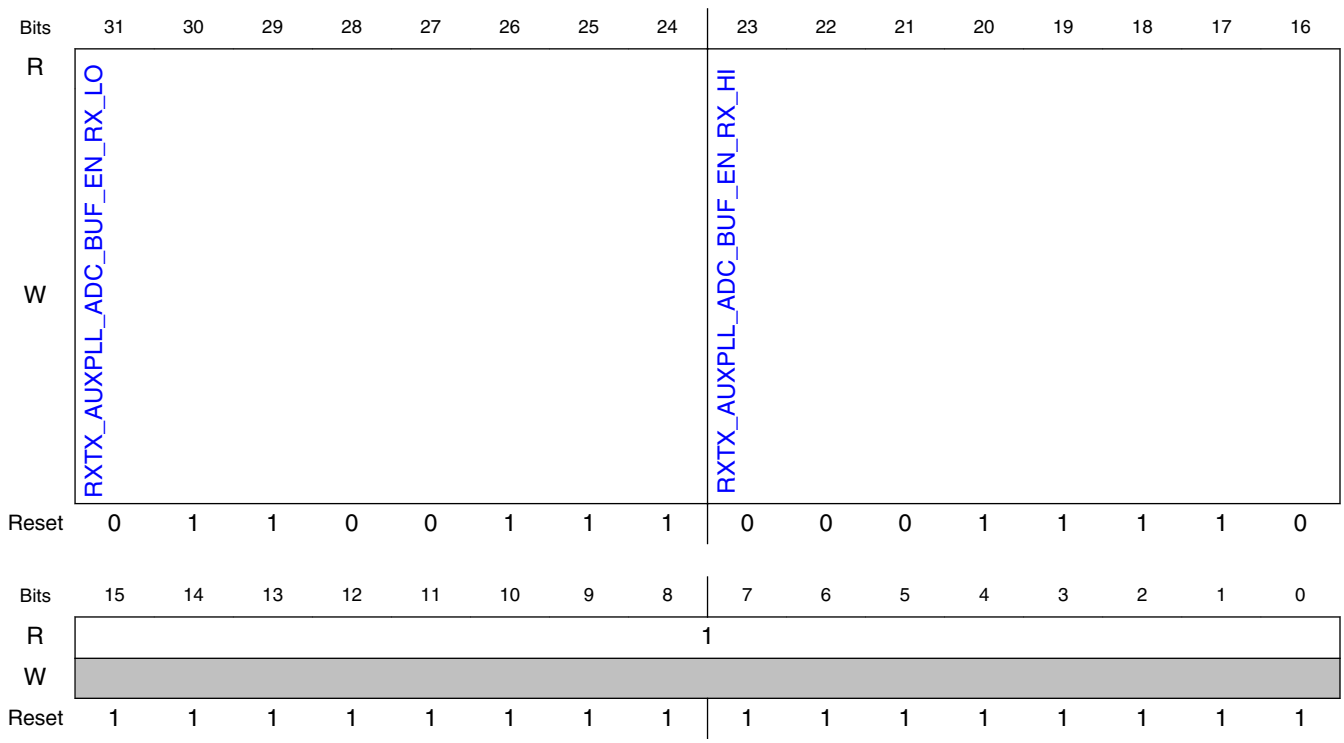
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.67 TSM_TIMING55 (TIMING55)

### *64.3.5.3.1.67.1 Offset*

| Register | Offset |
|---|---|
| TIMING55 | 10Ch |

### *64.3.5.3.1.67.2 Diagram*




### *64.3.5.3.1.67.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_ADC_BUF_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_ADC_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_ADC_BUF_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_ADC_BUF_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_ADC_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_ADC_BUF_EN signal or group will transition from LO to HI. |

*Table continues on the next page...*

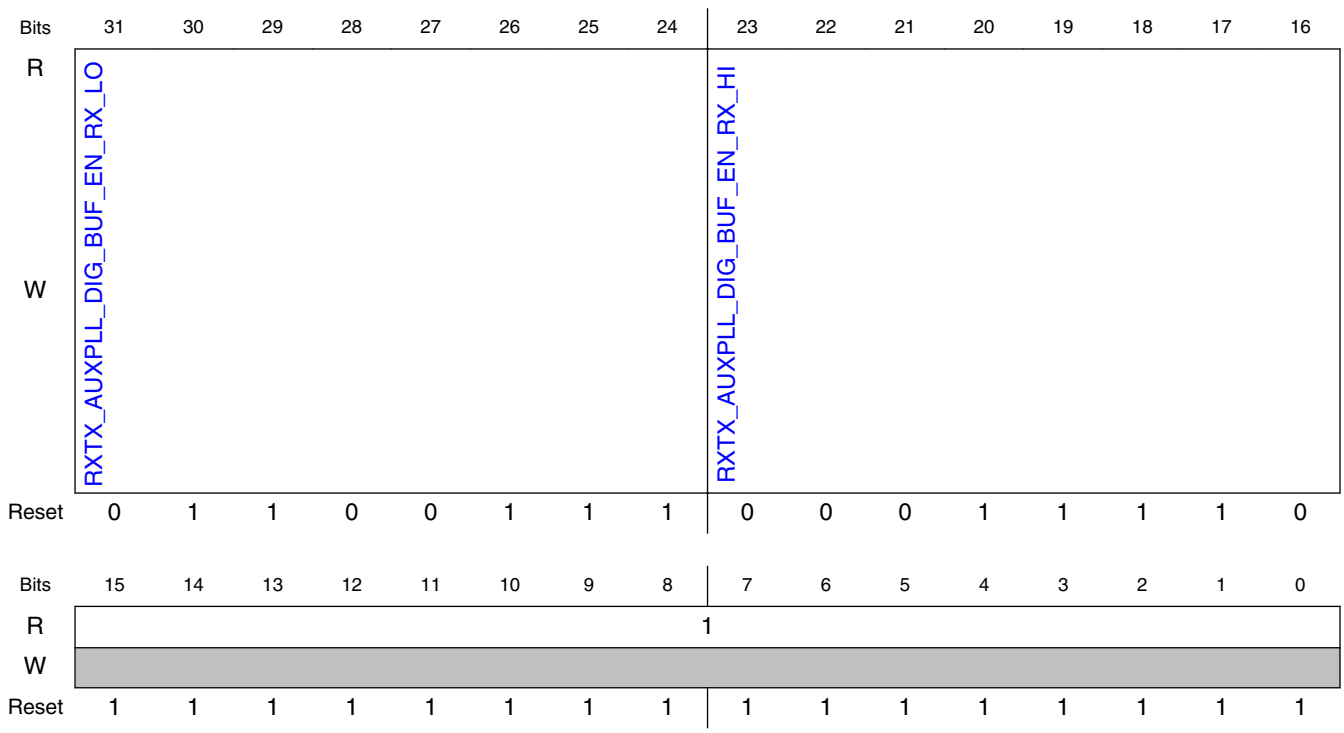
| Field | Function |
|---|---|
| 15-0<br>— | Reserved |

## 64.3.5.3.1.68 TSM_TIMING56 (TIMING56)

### *64.3.5.3.1.68.1 Offset*

| Register | Offset |
|---|---|
| TIMING56 | 110h |

### *64.3.5.3.1.68.2 Diagram*




### *64.3.5.3.1.68.3 Fields*

| Field | Function |
|---|---|
| 31-24 | De-assertion time setting for RXTX_AUXPLL_DIG_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_DIG_BUF_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXTX_AUXPLL_DIG_BUF_EN_RX_LO | |
| 23-16<br>RXTX_AUXPLL_DIG_BUF_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_DIG_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_DIG_BUF_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## 64.3.5.3.1.69 TSM_TIMING57 (TIMING57)

### *64.3.5.3.1.69.1 Offset*

| Register | Offset |
|---|---|
| TIMING57 | 114h |

### *64.3.5.3.1.69.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RXTX_RCCAL_EN_RX_LO | | | | | | | | RXTX_RCCAL_EN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### *64.3.5.3.1.69.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>RXTX_RCCAL_EN_RX_LO | De-assertion time setting for RXTX_RCCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_RCCAL_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_RCCAL_EN_RX_HI | Assertion time setting for RXTX_RCCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_RCCAL_EN signal or group will transition from LO to HI. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-0<br>— | Reserved |

## 64.3.5.3.1.70 TSM_TIMING58 (TIMING58)

### *64.3.5.3.1.70.1 Offset*

| Register | Offset |
|---|---|
| TIMING58 | 118h |

### *64.3.5.3.1.70.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_HPM_DAC_EN_TX_LO | | | | | | | | TX_HPM_DAC_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### *64.3.5.3.1.70.3 Fields*

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_HPM_DAC_EN_TX_LO | De-assertion time setting for TX_HPM_DAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_HPM_DAC_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_HPM_DAC_EN_TX_HI | Assertion time setting for TX_HPM_DAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_HPM_DAC_EN signal or group will transition from LO to HI. |

## 64.3.5.3.1.71 TSM OVERRIDE REGISTER 0 (OVRD0)

### 64.3.5.3.1.71.1 Offset

| Register | Offset |
|---|---|
| OVRD0 | 11Ch |

### 64.3.5.3.1.71.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | SY_VCO_EN_OVRD | SY_VCO_EN_OVRD_EN | SY_PD_CYCLE_SLIP_LD_EN_OVRD | SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN | PLL_LOOP_IS_OPEN_OVRD | PLL_LOOP_IS_OPEN_OVRD_EN | BB_XTAL_AUXPLL_REF_CLK_EN_OVRD | BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN | BB_XTAL_DAC_REF_CLK_EN_OVRD | BB_XTAL_DAC_REF_CLK_EN_OVRD_EN | BB_XTAL_PLL_REF_CLK_EN_OVRD | BB_XTAL_PLL_REF_CLK_EN_OVRD_EN | BB_LDO_VCOLO_FASTCHARGE_EN_OVRD | BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN | BB_LDO_VCOLO_BLEED_EN_OVRD | BB_LDO_VCOLO_BLEED_EN_OVRD_EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | BB_LDO_FDBK_BLEED_EN_OVRD | BB_LDO_FDBK_BLEED_EN_OVRD_EN | BB_LDO_VTREF_EN_OVRD | BB_LDO_VTREF_EN_OVRD_EN | BB_LDO_VCOLO_EN_OVRD | BB_LDO_VCOLO_EN_OVRD_EN | BB_LDO_FDBK_EN_OVRD | BB_LDO_FDBK_EN_OVRD_EN | BB_LDO_PD_EN_OVRD | BB_LDO_PD_EN_OVRD_EN | BB_LDO_BBA_EN_OVRD | BB_LDO_BBA_EN_OVRD_EN | BB_LDO_ADCDAC_EN_OVRD | BB_LDO_ADCDAC_EN_OVRD_EN | BB_LDO_HF_EN_OVRD | BB_LDO_HF_EN_OVRD_EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.5.3.1.71.3 Fields*

| Field | Function |
|---|---|
| 31<br>SY_VCO_EN_OVRD | Override value for SY_VCO_EN<br>When SY_VCO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_vco_en". This bit is ignored when SY_VCO_EN_OVRD_EN==0. |
| 30<br>SY_VCO_EN_OVRD_EN | Override control for SY_VCO_EN<br>0b - Normal operation.<br>1b - Use the state of SY_VCO_EN_OVRD to override the signal "sy_vco_en". |
| 29<br>SY_PD_CYCLE_SLIP_LD_EN_OVRD | Override value for SY_PD_CYCLE_SLIP_LD_EN<br>When SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_cycle_slip_ld_en". This bit is ignored when SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN==0. |
| 28<br>SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN | Override control for SY_PD_CYCLE_SLIP_LD_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_CYCLE_SLIP_LD_EN_OVRD to override the signal "sy_pd_cycle_slip_ld_en". |
| 27<br>PLL_LOOP_IS_OPEN_OVRD | Override value for PLL_LOOP_IS_OPEN<br>When PLL_LOOP_IS_OPEN_OVRD_EN=1, this value overrides the mission mode state of the signal "pll_loop_is_open". This bit is ignored when PLL_LOOP_IS_OPEN_OVRD_EN==0. |
| 26<br>PLL_LOOP_IS_OPEN_OVRD_EN | Override control for PLL_LOOP_IS_OPEN<br>0b - Normal operation.<br>1b - Use the state of PLL_LOOP_IS_OPEN_OVRD to override the signal "pll_loop_is_open". |
| 25<br>BB_XTAL_AUXPLL_REF_CLK_EN_OVRD | Override value for BB_XTAL_AUXPLL_REF_CLK_EN<br>When BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_auxpll_ref_clk_en". This bit is ignored when BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN==0. |
| 24<br>BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN | Override control for BB_XTAL_AUXPLL_REF_CLK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_XTAL_AUXPLL_REF_CLK_EN_OVRD to override the signal "bb_xtal_auxpll_ref_clk_en". |
| 23<br>BB_XTAL_DAC_REF_CLK_EN_OVRD | Override value for BB_XTAL_DAC_REF_CLK_EN<br>When BB_XTAL_DAC_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_dac_ref_clk_en". This bit is ignored when BB_XTAL_DAC_REF_CLK_EN_OVRD_EN==0. |
| 22<br>BB_XTAL_DAC_REF_CLK_EN_OVRD_EN | Override control for BB_XTAL_DAC_REF_CLK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_XTAL_DAC_REF_CLK_EN_OVRD to override the signal "bb_xtal_dac_ref_clk_en". |
| 21<br>BB_XTAL_PLL_REF_CLK_EN_OVRD | Override value for BB_XTAL_PLL_REF_CLK_EN<br>When BB_XTAL_PLL_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_pll_ref_clk_en". This bit is ignored when BB_XTAL_PLL_REF_CLK_EN_OVRD_EN==0. |
| 20 | Override control for BB_XTAL_PLL_REF_CLK_EN<br>0b - Normal operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BB_XTAL_PLL_REF_CLK_EN_OVRD_EN | 1b - Use the state of BB_XTAL_PLL_REF_CLK_EN_OVRD to override the signal "bb_xtal_pll_ref_clk_en". |
| 19<br>BB_LDO_VCOLO_FASTCHARGE_EN_OVRD | Override value for BB_LDO_VCOLO_FASTCHARGE_EN<br>When BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_fastcharge_en". This bit is ignored when BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN==0. |
| 18<br>BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN | Override control for BB_LDO_VCOLO_FASTCHARGE_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VCOLO_FASTCHARGE_EN_OVRD to override the signal "bb_ldo_vcolo_fastcharge_en". |
| 17<br>BB_LDO_VCOLO_BLEED_EN_OVRD | Override value for BB_LDO_VCOLO_BLEED_EN<br>When BB_LDO_VCOLO_BLEED_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_bleed_en". This bit is ignored when BB_LDO_VCOLO_BLEED_EN_OVRD_EN==0. |
| 16<br>BB_LDO_VCOLO_BLEED_EN_OVRD_EN | Override control for BB_LDO_VCOLO_BLEED_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VCOLO_BLEED_EN_OVRD to override the signal "bb_ldo_vcolo_bleed_en". |
| 15<br>BB_LDO_FDBK_BLEED_EN_OVRD | Override value for BB_LDO_FDBK_BLEED_EN<br>When BB_LDO_FDBK_BLEED_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_fdbk_bleed_en". This bit is ignored when BB_LDO_FDBK_BLEED_EN_OVRD_EN==0. |
| 14<br>BB_LDO_FDBK_BLEED_EN_OVRD_EN | Override control for BB_LDO_FDBK_BLEED_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_FDBK_BLEED_EN_OVRD to override the signal "bb_ldo_fdbk_bleed_en". |
| 13<br>BB_LDO_VTREF_EN_OVRD | Override value for BB_LDO_VTREF_EN<br>When BB_LDO_VTREF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vtref_en". This bit is ignored when BB_LDO_VTREF_EN_OVRD_EN==0. |
| 12<br>BB_LDO_VTREF_EN_OVRD_EN | Override control for BB_LDO_VTREF_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VTREF_EN_OVRD to override the signal "bb_ldo_vtref_en". |
| 11<br>BB_LDO_VCOLO_EN_OVRD | Override value for BB_LDO_VCOLO_EN<br>When BB_LDO_VCOLO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_en". This bit is ignored when BB_LDO_VCOLO_EN_OVRD_EN==0. |
| 10<br>BB_LDO_VCOLO_EN_OVRD_EN | Override control for BB_LDO_VCOLO_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VCOLO_EN_OVRD to override the signal "bb_ldo_vcolo_en". |
| 9<br>BB_LDO_FDBK_EN_OVRD | Override value for BB_LDO_FDBK_EN<br>When BB_LDO_FDBK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_fdbk_en". This bit is ignored when BB_LDO_FDBK_EN_OVRD_EN==0. |
| 8 | Override control for BB_LDO_FDBK_EN<br>0b - Normal operation. |

*Table continues on the next page...*

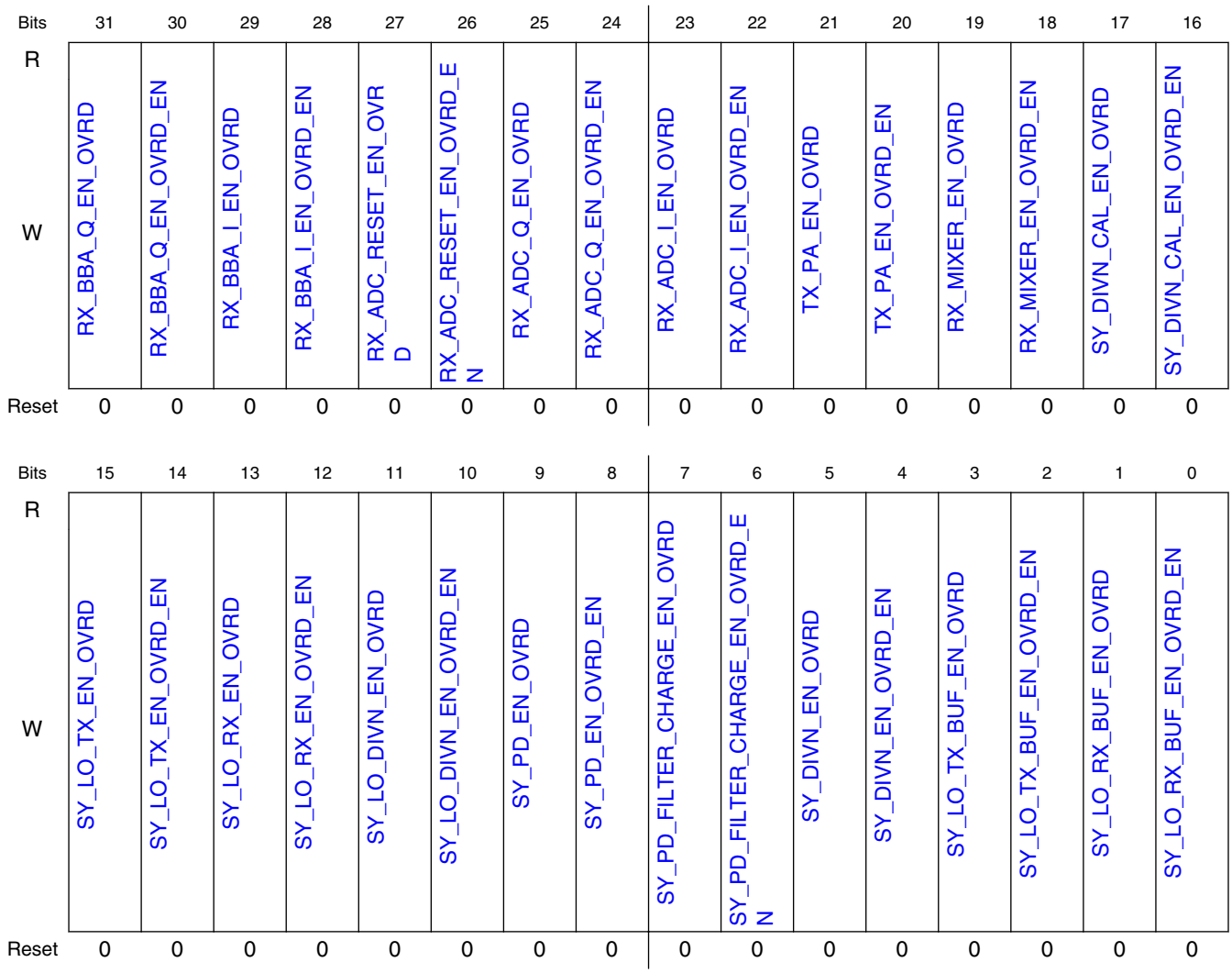
| Field | Function |
|---|---|
| BB_LDO_FDBK_EN_OVRD_EN | 1b - Use the state of BB_LDO_FDBK_EN_OVRD to override the signal "bb_ldo_fdbk_en". |
| 7<br>BB_LDO_PD_EN_OVRD | Override value for BB_LDO_PD_EN<br>When BB_LDO_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_pd_en". This bit is ignored when BB_LDO_PD_EN_OVRD_EN==0. |
| 6<br>BB_LDO_PD_EN_OVRD_EN | Override control for BB_LDO_PD_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_PD_EN_OVRD to override the signal "bb_ldo_pd_en". |
| 5<br>BB_LDO_BBA_EN_OVRD | Override value for BB_LDO_BBA_EN<br>When BB_LDO_BBA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_bba_en". This bit is ignored when BB_LDO_BBA_EN_OVRD_EN==0. |
| 4<br>BB_LDO_BBA_EN_OVRD_EN | Override control for BB_LDO_BBA_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_BBA_EN_OVRD to override the signal "bb_ldo_bba_en". |
| 3<br>BB_LDO_ADCDAC_EN_OVRD | Override value for BB_LDO_ADCDAC_EN<br>When BB_LDO_ADCDAC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_adcdac_en". This bit is ignored when BB_LDO_ADCDAC_EN_OVRD_EN==0. |
| 2<br>BB_LDO_ADCDAC_EN_OVRD_EN | Override control for BB_LDO_ADCDAC_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_ADCDAC_EN_OVRD to override the signal "bb_ldo_adcdac_en". |
| 1<br>BB_LDO_HF_EN_OVRD | Override value for BB_LDO_HF_EN<br>When BB_LDO_HF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_hf_en". This bit is ignored when BB_LDO_HF_EN_OVRD_EN==0. |
| 0<br>BB_LDO_HF_EN_OVRD_EN | Override control for BB_LDO_HF_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_HF_EN_OVRD to override the signal "bb_ldo_hf_en". |

## 64.3.5.3.1.72 TSM OVERRIDE REGISTER 1 (OVRD1)

### *64.3.5.3.1.72.1 Offset*

| Register | Offset |
|---|---|
| OVRD1 | 120h |

### 64.3.5.3.1.72.2 Diagram




### 64.3.5.3.1.72.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_BBA_Q_EN_OVRD | Override value for RX_BBA_Q_EN<br>When RX_BBA_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_q_en". This bit is ignored when RX_BBA_Q_EN_OVRD_EN==0. |
| 30<br>RX_BBA_Q_EN_OVRD_EN | Override control for RX_BBA_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_BBA_Q_EN_OVRD to override the signal "rx_bba_q_en". |
| 29<br>RX_BBA_I_EN_OVRD | Override value for RX_BBA_I_EN<br>When RX_BBA_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_i_en". This bit is ignored when RX_BBA_I_EN_OVRD_EN==0. |
| 28 | Override control for RX_BBA_I_EN<br>0b - Normal operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RX_BBA_I_EN_OVRD_EN | 1b - Use the state of RX_BBA_I_EN_OVRD to override the signal "rx_bba_i_en". |
| 27<br>RX_ADC_RESET_EN_OVRD | Override value for RX_ADC_RESET_EN<br>When RX_ADC_RESET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_reset_en". This bit is ignored when RX_ADC_RESET_EN_OVRD_EN==0. |
| 26<br>RX_ADC_RESET_EN_OVRD_EN | Override control for RX_ADC_RESET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_RESET_EN_OVRD to override the signal "rx_adc_reset_en". |
| 25<br>RX_ADC_Q_EN_OVRD | Override value for RX_ADC_Q_EN<br>When RX_ADC_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_q_en". This bit is ignored when RX_ADC_Q_EN_OVRD_EN==0. |
| 24<br>RX_ADC_Q_EN_OVRD_EN | Override control for RX_ADC_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_Q_EN_OVRD to override the signal "rx_adc_q_en". |
| 23<br>RX_ADC_I_EN_OVRD | Override value for RX_ADC_I_EN<br>When RX_ADC_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_i_en". This bit is ignored when RX_ADC_I_EN_OVRD_EN==0. |
| 22<br>RX_ADC_I_EN_OVRD_EN | Override control for RX_ADC_I_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_I_EN_OVRD to override the signal "rx_adc_i_en". |
| 21<br>TX_PA_EN_OVRD | Override value for TX_PA_EN<br>When TX_PA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_pa_en". This bit is ignored when TX_PA_EN_OVRD_EN==0. |
| 20<br>TX_PA_EN_OVRD_EN | Override control for TX_PA_EN<br>0b - Normal operation.<br>1b - Use the state of TX_PA_EN_OVRD to override the signal "tx_pa_en". |
| 19<br>RX_MIXER_EN_OVRD | Override value for RX_MIXER_EN<br>When RX_MIXER_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_mixer_en". This bit is ignored when RX_MIXER_EN_OVRD_EN==0. |
| 18<br>RX_MIXER_EN_OVRD_EN | Override control for RX_MIXER_EN<br>0b - Normal operation.<br>1b - Use the state of RX_MIXER_EN_OVRD to override the signal "rx_mixer_en". |
| 17<br>SY_DIVN_CAL_EN_OVRD | Override value for SY_DIVN_CAL_EN<br>When SY_DIVN_CAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_divn_cal_en". This bit is ignored when SY_DIVN_CAL_EN_OVRD_EN==0. |
| 16<br>SY_DIVN_CAL_EN_OVRD_EN | Override control for SY_DIVN_CAL_EN<br>0b - Normal operation.<br>1b - Use the state of SY_DIVN_CAL_EN_OVRD to override the signal "sy_divn_cal_en". |
| 15<br>SY_LO_TX_EN_OVRD | Override value for SY_LO_TX_EN<br>When SY_LO_TX_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_tx_en". This bit is ignored when SY_LO_TX_EN_OVRD_EN==0. |
| 14<br>SY_LO_TX_EN_OVRD_EN | Override control for SY_LO_TX_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_TX_EN_OVRD to override the signal "sy_lo_tx_en". |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13<br>SY_LO_RX_EN_OVRD | Override value for SY_LO_RX_EN<br>When SY_LO_RX_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_rx_en". This bit is ignored when SY_LO_RX_EN_OVRD_EN==0. |
| 12<br>SY_LO_RX_EN_OVRD_EN | Override control for SY_LO_RX_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_RX_EN_OVRD to override the signal "sy_lo_rx_en". |
| 11<br>SY_LO_DIVN_EN_OVRD | Override value for SY_LO_DIVN_EN<br>When SY_LO_DIVN_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_divn_en". This bit is ignored when SY_LO_DIVN_EN_OVRD_EN==0. |
| 10<br>SY_LO_DIVN_EN_OVRD_EN | Override control for SY_LO_DIVN_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_DIVN_EN_OVRD to override the signal "sy_lo_divn_en". |
| 9<br>SY_PD_EN_OVRD | Override value for SY_PD_EN<br>When SY_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_en". This bit is ignored when SY_PD_EN_OVRD_EN==0. |
| 8<br>SY_PD_EN_OVRD_EN | Override control for SY_PD_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_EN_OVRD to override the signal "sy_pd_en". |
| 7<br>SY_PD_FILTER_CHARGE_EN_OVRD | Override value for SY_PD_FILTER_CHARGE_EN<br>When SY_PD_FILTER_CHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_filter_charge_en". This bit is ignored when SY_PD_FILTER_CHARGE_EN_OVRD_EN==0. |
| 6<br>SY_PD_FILTER_CHARGE_EN_OVRD_EN | Override control for SY_PD_FILTER_CHARGE_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_FILTER_CHARGE_EN_OVRD to override the signal "sy_pd_filter_charge_en". |
| 5<br>SY_DIVN_EN_OVRD | Override value for SY_DIVN_EN<br>When SY_DIVN_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_divn_en". This bit is ignored when SY_DIVN_EN_OVRD_EN==0. |
| 4<br>SY_DIVN_EN_OVRD_EN | Override control for SY_DIVN_EN<br>0b - Normal operation.<br>1b - Use the state of SY_DIVN_EN_OVRD to override the signal "sy_divn_en". |
| 3<br>SY_LO_TX_BUF_EN_OVRD | Override value for SY_LO_TX_BUF_EN<br>When SY_LO_TX_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_tx_buf_en". This bit is ignored when SY_LO_TX_BUF_EN_OVRD_EN==0. |
| 2<br>SY_LO_TX_BUF_EN_OVRD_EN | Override control for SY_LO_TX_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_TX_BUF_EN_OVRD to override the signal "sy_lo_tx_buf_en". |
| 1<br>SY_LO_RX_BUF_EN_OVRD | Override value for SY_LO_RX_BUF_EN<br>When SY_LO_RX_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_rx_buf_en". This bit is ignored when SY_LO_RX_BUF_EN_OVRD_EN==0. |
| 0<br>SY_LO_RX_BUF_EN_OVRD_EN | Override control for SY_LO_RX_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_RX_BUF_EN_OVRD to override the signal "sy_lo_rx_buf_en". |

## 64.3.5.3.1.73 TSM OVERRIDE REGISTER 2 (OVRD2)

### *64.3.5.3.1.73.1 Offset*

| Register | Offset |
|---|---|
| OVRD2 | 124h |

### *64.3.5.3.1.73.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FREQ_TARG_LD_EN_OVRD | FREQ_TARG_LD_EN_OVRD_EN | DCOC_INIT_OVRD | DCOC_INIT_OVRD_EN | DCOC_EN_OVRD | DCOC_EN_OVRD_EN | RX_PHY_EN_OVRD | RX_PHY_EN_OVRD_EN | SIGMA_DELTA_EN_OVRD | SIGMA_DELTA_EN_OVRD_EN | RX_INIT_OVRD | RX_INIT_OVRD_EN | RX_DIG_EN_OVRD | RX_DIG_EN_OVRD_EN | TX_DIG_EN_OVRD | TX_DIG_EN_OVRD_EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PLL_DIG_EN_OVRD | PLL_DIG_EN_OVRD_EN | RX_TZA_DCOC_EN_OVRD | RX_TZA_DCOC_EN_OVRD_EN | RX_TZA_PDET_EN_OVRD | RX_TZA_PDET_EN_OVRD_EN | RX_TZA_Q_EN_OVRD | RX_TZA_Q_EN_OVRD_EN | RX_TZA_I_EN_OVRD | RX_TZA_I_EN_OVRD_EN | RX_LNA_EN_OVRD | RX_LNA_EN_OVRD_EN | RX_BBA_DCOC_EN_OVRD | RX_BBA_DCOC_EN_OVRD_EN | RX_BBA_PDET_EN_OVRD | RX_BBA_PDET_EN_OVRD_EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.5.3.1.73.3 Fields*

| Field | Function |
|---|---|
| 31 | Override value for FREQ_TARG_LD_EN |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FREQ_TARG_LD_EN_OVRD | When FREQ_TARG_LD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "freq_targ_ld_en". This bit is ignored when FREQ_TARG_LD_EN_OVRD_EN==0. |
| 30<br>FREQ_TARG_LD_EN_OVRD_EN | Override control for FREQ_TARG_LD_EN<br>0b - Normal operation.<br>1b - Use the state of FREQ_TARG_LD_EN_OVRD to override the signal "freq_targ_ld_en". |
| 29<br>DCOC_INIT_OVRD | Override value for DCOC_INIT<br>When DCOC_INIT_OVRD_EN=1, this value overrides the mission mode state of the signal "dcoc_init". This bit is ignored when DCOC_INIT_OVRD_EN==0. |
| 28<br>DCOC_INIT_OVRD_EN | Override control for DCOC_INIT<br>0b - Normal operation.<br>1b - Use the state of DCOC_INIT_OVRD to override the signal "dcoc_init". |
| 27<br>DCOC_EN_OVRD | Override value for DCOC_EN<br>When DCOC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "dcoc_en". This bit is ignored when DCOC_EN_OVRD_EN==0. |
| 26<br>DCOC_EN_OVRD_EN | Override control for DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of DCOC_EN_OVRD to override the signal "dcoc_en". |
| 25<br>RX_PHY_EN_OVRD | Override value for RX_PHY_EN<br>When RX_PHY_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_phy_en". This bit is ignored when RX_PHY_EN_OVRD_EN==0. |
| 24<br>RX_PHY_EN_OVRD_EN | Override control for RX_PHY_EN<br>0b - Normal operation.<br>1b - Use the state of RX_PHY_EN_OVRD to override the signal "rx_phy_en". |
| 23<br>SIGMA_DELTA_EN_OVRD | Override value for SIGMA_DELTA_EN<br>When SIGMA_DELTA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sigma_delta_en". This bit is ignored when SIGMA_DELTA_EN_OVRD_EN==0. |
| 22<br>SIGMA_DELTA_EN_OVRD_EN | Override control for SIGMA_DELTA_EN<br>0b - Normal operation.<br>1b - Use the state of SIGMA_DELTA_EN_OVRD to override the signal "sigma_delta_en". |
| 21<br>RX_INIT_OVRD | Override value for RX_INIT<br>When RX_INIT_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_init". This bit is ignored when RX_INIT_OVRD_EN==0. |
| 20<br>RX_INIT_OVRD_EN | Override control for RX_INIT<br>0b - Normal operation.<br>1b - Use the state of RX_INIT_OVRD to override the signal "rx_init". |
| 19<br>RX_DIG_EN_OVRD | Override value for RX_DIG_EN<br>When RX_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_dig_en". This bit is ignored when RX_DIG_EN_OVRD_EN==0. |
| 18<br>RX_DIG_EN_OVRD_EN | Override control for RX_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of RX_DIG_EN_OVRD to override the signal "rx_dig_en". |
| 17<br>TX_DIG_EN_OVRD | Override value for TX_DIG_EN<br>When TX_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_dig_en". This bit is ignored when TX_DIG_EN_OVRD_EN==0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 16<br>TX_DIG_EN_OVRD_EN | Override control for TX_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of TX_DIG_EN_OVRD to override the signal "tx_dig_en". |
| 15<br>PLL_DIG_EN_OVRD | Override value for PLL_DIG_EN<br>When PLL_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "pll_dig_en". This bit is ignored when PLL_DIG_EN_OVRD_EN==0. |
| 14<br>PLL_DIG_EN_OVRD_EN | Override control for PLL_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of PLL_DIG_EN_OVRD to override the signal "pll_dig_en". |
| 13<br>RX_TZA_DCOC_EN_OVRD | Override control for RX_TZA_DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_DCOC_EN_OVRD to override the signal "rx_tza_dcoc_en". |
| 12<br>RX_TZA_DCOC_EN_OVRD_EN | Override control for RX_TZA_DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_DCOC_EN_OVRD to override the signal "rx_tza_dcoc_en". |
| 11<br>RX_TZA_PDET_EN_OVRD | Override value for RX_TZA_PDET_EN<br>When RX_TZA_PDET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_pdet_en". This bit is ignored when RX_TZA_PDET_EN_OVRD_EN==0. |
| 10<br>RX_TZA_PDET_EN_OVRD_EN | Override control for RX_TZA_PDET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_PDET_EN_OVRD to override the signal "rx_tza_pdet_en". |
| 9<br>RX_TZA_Q_EN_OVRD | Override value for RX_TZA_Q_EN<br>When RX_TZA_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_q_en". This bit is ignored when RX_TZA_Q_EN_OVRD_EN==0. |
| 8<br>RX_TZA_Q_EN_OVRD_EN | Override control for RX_TZA_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_Q_EN_OVRD to override the signal "rx_tza_q_en". |
| 7<br>RX_TZA_I_EN_OVRD | Override value for RX_TZA_I_EN<br>When RX_TZA_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_i_en". This bit is ignored when RX_TZA_I_EN_OVRD_EN==0. |
| 6<br>RX_TZA_I_EN_OVRD_EN | Override control for RX_TZA_I_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_I_EN_OVRD to override the signal "rx_tza_i_en". |
| 5<br>RX_LNA_EN_OVRD | Override value for RX_LNA_EN<br>When RX_LNA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_lna_en". This bit is ignored when RX_LNA_EN_OVRD_EN==0. |
| 4<br>RX_LNA_EN_OVRD_EN | Override control for RX_LNA_EN<br>0b - Normal operation.<br>1b - Use the state of RX_LNA_EN_OVRD to override the signal "rx_lna_en". |
| 3<br>RX_BBA_DCOC_EN_OVRD | Override value for RX_BBA_DCOC_EN<br>When RX_BBA_DCOC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_dcoc_en". This bit is ignored when RX_BBA_DCOC_EN_OVRD_EN==0. |
| 2 | Override control for RX_BBA_DCOC_EN<br>0b - Normal operation. |

*Table continues on the next page...*

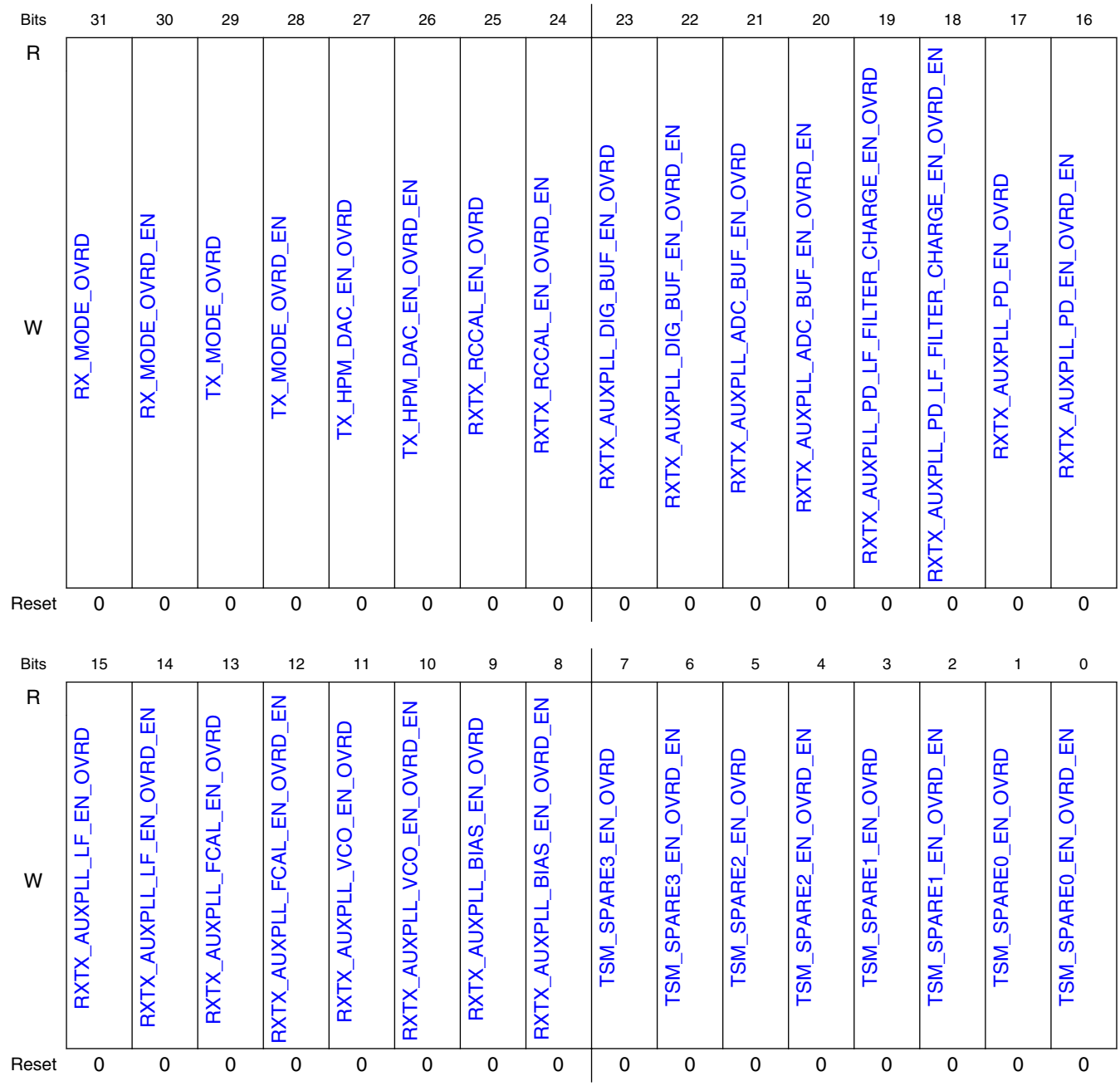
| Field | Function |
|---|---|
| RX_BBA_DCOC_EN_OVRD_EN | 1b - Use the state of RX_BBA_DCOC_EN_OVRD to override the signal "rx_bba_dcoc_en". |
| 1<br>RX_BBA_PDET_EN_OVRD | Override value for RX_BBA_PDET_EN<br>When RX_BBA_PDET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_pdet_en". This bit is ignored when RX_BBA_PDET_EN_OVRD_EN==0. |
| 0<br>RX_BBA_PDET_EN_OVRD_EN | Override control for RX_BBA_PDET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_BBA_PDET_EN_OVRD to override the signal "rx_bba_pdet_en". |

## 64.3.5.3.1.74 TSM OVERRIDE REGISTER 3 (OVRD3)

### *64.3.5.3.1.74.1 Offset*

| Register | Offset |
|---|---|
| OVRD3 | 128h |

### 64.3.5.3.1.74.2 Diagram




### 64.3.5.3.1.74.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_MODE_OVRD | Override value for RX_MODE<br>When RX_MODE_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_mode". This bit is ignored when RX_MODE_OVRD_EN==0. |
| 30 | Override control for RX_MODE |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RX_MODE_OVRD_EN | 0b - Normal operation.<br>1b - Use the state of RX_MODE_OVRD to override the signal "rx_mode". |
| 29<br>TX_MODE_OVRD | Override value for TX_MODE<br>When TX_MODE_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_mode". This bit is ignored when TX_MODE_OVRD_EN==0. |
| 28<br>TX_MODE_OVRD_EN | Override control for TX_MODE<br>0b - Normal operation.<br>1b - Use the state of TX_MODE_OVRD to override the signal "tx_mode". |
| 27<br>TX_HPM_DAC_EN_OVRD | Override value for TX_HPM_DAC_EN<br>When TX_HPM_DAC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_hpm_dac_en". This bit is ignored when TX_HPM_DAC_EN_OVRD_EN==0. |
| 26<br>TX_HPM_DAC_EN_OVRD_EN | Override control for TX_HPM_DAC_EN<br>0b - Normal operation.<br>1b - Use the state of TX_HPM_DAC_EN_OVRD to override the signal "tx_hpm_dac_en". |
| 25<br>RXTX_RCCAL_EN_OVRD | Override value for RXTX_RCCAL_EN<br>When RXTX_RCCAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_rccal_en". This bit is ignored when RXTX_RCCAL_EN_OVRD_EN==0. |
| 24<br>RXTX_RCCAL_EN_OVRD_EN | Override control for RXTX_RCCAL_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_RCCAL_EN_OVRD to override the signal "rxtx_rccal_en". |
| 23<br>RXTX_AUXPLL_DIG_BUF_EN_OVRD | Override value for RXTX_AUXPLL_DIG_BUF_EN<br>When RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_dig_buf_en". This bit is ignored when RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN==0. |
| 22<br>RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN | Override control for RXTX_AUXPLL_DIG_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_DIG_BUF_EN_OVRD to override the signal "rxtx_auxpll_dig_buf_en". |
| 21<br>RXTX_AUXPLL_ADC_BUF_EN_OVRD | Override value for RXTX_AUXPLL_ADC_BUF_EN<br>When RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_adc_buf_en". This bit is ignored when RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN==0. |
| 20<br>RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN | Override control for RXTX_AUXPLL_ADC_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_ADC_BUF_EN_OVRD to override the signal "rxtx_auxpll_adc_buf_en". |
| 19<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD | Override value for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN<br>When RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_pd_lf_filter_charge_en". This bit is ignored when RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN==0. |
| 18<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN | Override control for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD to override the signal "rxtx_auxpll_pd_lf_filter_charge_en". |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 17<br>RXTX_AUXPLL_PD_EN_OVRD | Override value for RXTX_AUXPLL_PD_EN<br>When RXTX_AUXPLL_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_pd_en". This bit is ignored when RXTX_AUXPLL_PD_EN_OVRD_EN==0. |
| 16<br>RXTX_AUXPLL_PD_EN_OVRD_EN | Override control for RXTX_AUXPLL_PD_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_PD_EN_OVRD to override the signal "rxtx_auxpll_pd_en". |
| 15<br>RXTX_AUXPLL_LF_EN_OVRD | Override value for RXTX_AUXPLL_LF_EN<br>When RXTX_AUXPLL_LF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_lf_en". This bit is ignored when RXTX_AUXPLL_LF_EN_OVRD_EN==0. |
| 14<br>RXTX_AUXPLL_LF_EN_OVRD_EN | Override control for RXTX_AUXPLL_LF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_LF_EN_OVRD to override the signal "rxtx_auxpll_lf_en". |
| 13<br>RXTX_AUXPLL_FCAL_EN_OVRD | Override value for RXTX_AUXPLL_FCAL_EN<br>When RXTX_AUXPLL_FCAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_fcal_en". This bit is ignored when RXTX_AUXPLL_FCAL_EN_OVRD_EN==0. |
| 12<br>RXTX_AUXPLL_FCAL_EN_OVRD_EN | Override control for RXTX_AUXPLL_FCAL_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_FCAL_EN_OVRD to override the signal "rxtx_auxpll_fcal_en". |
| 11<br>RXTX_AUXPLL_VCO_EN_OVRD | Override value for RXTX_AUXPLL_VCO_EN<br>When RXTX_AUXPLL_VCO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_vco_en". This bit is ignored when RXTX_AUXPLL_VCO_EN_OVRD_EN==0. |
| 10<br>RXTX_AUXPLL_VCO_EN_OVRD_EN | Override control for RXTX_AUXPLL_VCO_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_VCO_EN_OVRD to override the signal "rxtx_auxpll_vco_en". |
| 9<br>RXTX_AUXPLL_BIAS_EN_OVRD | Override value for RXTX_AUXPLL_BIAS_EN<br>When RXTX_AUXPLL_BIAS_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_bias_en". This bit is ignored when RXTX_AUXPLL_BIAS_EN_OVRD_EN==0. |
| 8<br>RXTX_AUXPLL_BIAS_EN_OVRD_EN | Override control for RXTX_AUXPLL_BIAS_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_BIAS_EN_OVRD to override the signal "rxtx_auxpll_bias_en". |
| 7<br>TSM_SPARE3_EN_OVRD | Override value for TSM_SPARE3_EN<br>When TSM_SPARE3_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare3_en". This bit is ignored when TSM_SPARE3_EN_OVRD_EN==0. |
| 6<br>TSM_SPARE3_EN_OVRD_EN | Override control for TSM_SPARE3_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE3_EN_OVRD to override the signal "tsm_spare3_en". |
| 5 | Override value for TSM_SPARE2_EN |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TSM_SPARE2_EN_OVRD | When TSM_SPARE2_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare2_en". This bit is ignored when TSM_SPARE2_EN_OVRD_EN==0. |
| 4<br>TSM_SPARE2_EN_OVRD_EN | Override control for TSM_SPARE2_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE2_EN_OVRD to override the signal "tsm_spare2_en". |
| 3<br>TSM_SPARE1_EN_OVRD | Override value for TSM_SPARE1_EN<br>When TSM_SPARE1_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare1_en". This bit is ignored when TSM_SPARE1_EN_OVRD_EN==0. |
| 2<br>TSM_SPARE1_EN_OVRD_EN | Override control for TSM_SPARE1_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE1_EN_OVRD to override the signal "tsm_spare1_en". |
| 1<br>TSM_SPARE0_EN_OVRD | Override value for TSM_SPARE0_EN<br>When TSM_SPARE0_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare0_en". This bit is ignored when TSM_SPARE0_EN_OVRD_EN==0. |
| 0<br>TSM_SPARE0_EN_OVRD_EN | Override control for TSM_SPARE0_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE0_EN_OVRD to override the signal "tsm_spare0_en". |

## 64.3.6 Transceiver DMA and Packet RAM Debug Modes

### 64.3.6.1 Introduction

For debug and evaluation purposes, the 2.4GHz Radio provides the capability to capture a variety of internal data during the reception process, and store it to either system memory (using Transceiver DMA mode) or radio Packet RAM (using Packet RAM Debug mode), for analysis and post-processing

#### 64.3.6.1.1 Overview

During silicon debug and evaluation, it is often desirable to gain access to data internal to the digital receiver, in order to, for example:

- troubleshoot packet loss issues
- troubleshoot unexpectedly high Bit Error Rate
- fine-tune receiver initialization and configuration parameters

The 2.4GHz radio offers 3 methods of accessing such internal data:

1. Transceiver DMA mode
2. Packet RAM Debug mode
3. Digital Teset Mux (DTEST)

This chapter describes the first 2 methods.

Examples of internal data desirable for capture are raw ADC samples, RX digital I and Q outputs, and PHY soft decision data. Transceiver DMA and Packet RAM Debug modes allow a source of internal data to be selected, automate the acquisition of the data, pack the data into a format suitable and optimized for its destination (system or radio memory), implement the interfaces required to access those memories, store the data to memory under program control, and subsequently allow host MCU access to the acquired data for analysis and post-processing. Selected data are organized into pages, to allow contextually related data to be simultaneously captured wherever possible, and allow for maximum utilization of available bandwidth to memory. For DMA mode, two transfer protocols are provided: a low-bandwidth protocol where the radio requests access to memory only when it has a new sample ready, and a high-bandwidth protocol that allows an entire block of radio data to be transferred to memory with a single radio request. For RAM debug mode, the entire radio RAM is made available for this debug mode, with hardware to pack data, generate RAM addressing, and detect RAM-full condition to prevent overwriting and loss of data.

#### 64.3.6.1.2 Features

- Captures any of 4 sources of RX_DIG data and 4 sources of PHY data
- DMA engine communicates with SoC DMA2 controller to access system memory
- DMA signalling compliant with IP Interface SRS DarkBlue Line convention
- DMA engine provides 2 options to optimize request signalling to required data rate
- Packet RAM Debug engine automates transfers to radio memory
- Efficient packing of source data optimized to destination memory dimensions
- Efficient packing of source data optimized to make best use of available bandwidth
- DMA and Packet RAM engine clocking disabled in mission mode to reduce power
- Simultaneous DMA and Packet RAM Debug operation possible
- Simultaneous Mission Mode and DMA operation possible
- Hardware start-triggering capability with 7 hardware sources
- Stop-on-trigger capability for Packet RAM Debug with 11 hardware sources

#### 64.3.6.1.3 Block Diagram

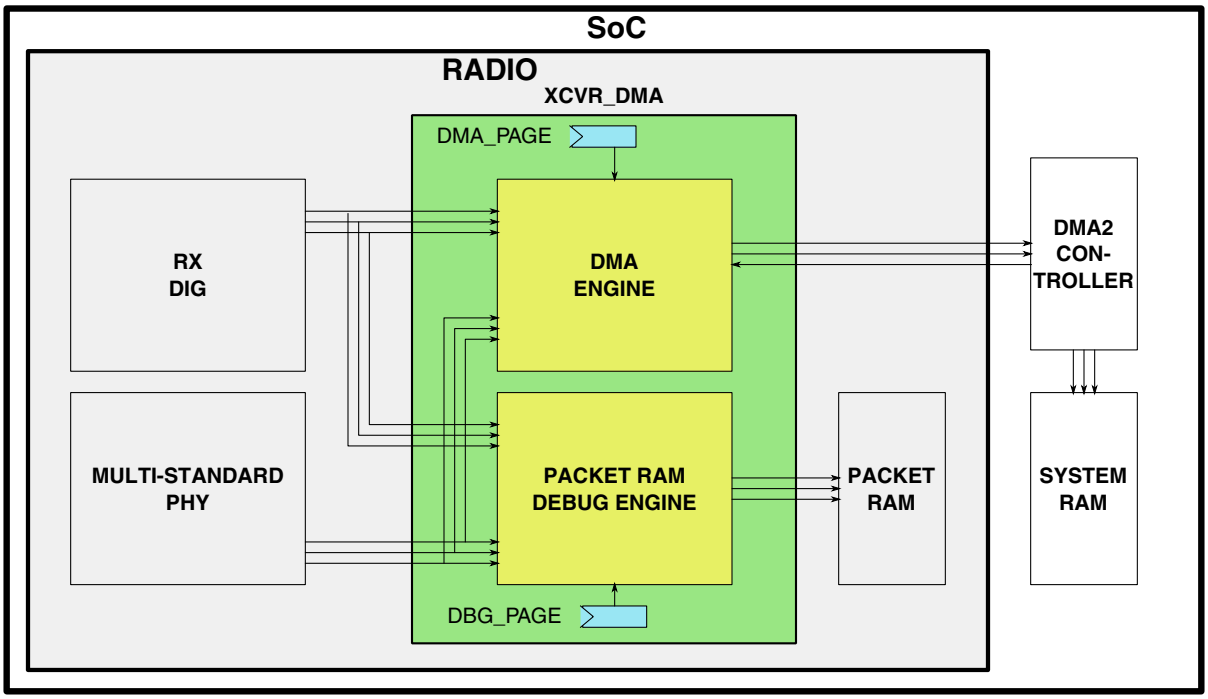



**Figure 64-86. Transceiver DMA and Packet RAM Debug**

### 64.3.6.2 Memory Map and Register Definition

#### 64.3.6.2.1 Register Summary

Below is an overview of the DMA and Packet RAM Debug control fields. More detail on these registers can be found in the XCVR_MISC register section.

#### 64.3.6.2.2 Register Descriptions

| Field | R/W | Description |
|---|---|---|
| DMA_PAGE[3:0] | rw | Transceiver DMA Page Selector. Selects the page of receiver data for storage to system memory when using Transceiver DMA Debug Mode. Setting this register to a non-zero value enables the DMA interface logic. Setting this register to zero disables the interface. The available DMA pages are listed in the following Table 64-39. |
| DMA_EN | rw | Transceiver DMA Enable. Setting DMA_EN=1 enables clock gating to the Transciever DMA engine. This bit should be set prior to the |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | start of a DMA session, and remain set for the duration of the session. |
| SINGLE_REQ_MODE | rw | DMA Single Request Mode. Configures the transceiver to transfer the entire block of DMA data with only a single initial request (ipd_req_radio_rx). The DMA2 controller must be configured accordingly. In this mode, the transceiver will use **ips_xfr_wait** to pace the individual transactions. Single Request Mode should not be used with DMA Pages 11, 12, or 13, because the data rate is too low and therefore ips_xfr_wait would remain assserted for excessively long periods. |
| BYPASS_DMA_SYNC | rw | Bypass External DMA Synchronization. When using transceiver DMA with SINGLE_REQ_MODE=0, synchronization external to the transceiver must be bypassed in order to minimize bus access latency. Using DMA in this mode also requires that the MCU and transceiver both are configured to operate in the RF Oscillator clock domain. |
| DMA_AA_TRIGGERED | r | DMA AA_TRIGGERED. This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of DMA transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DMA_PAGE to initiate DMA transfers. This bit is intended for use with DMA_PAGE=13, but is available on all pages. Its usage is optional. |
| DMA_TIMED_OUT | r | DMA Transfer Timed Out. Status bit indicates that the transceiver DMA, while operating in Single Request Mode, asserted ips_xfr_wait for a period in excess of the programmed **DMA_TIMEOUT** setting, resulting in a timeout condition where the transceiver has forced ips_xfr_wait low. After a timeout, before resuming DMA operations, set DMA_PAGE=0 to disable DMA, and write 1 to this bit to clear it. |
| DMA_TIMEOUT[3:0] | rw | DMA Timeout. In DMA Single Request Mode, adverse consequences may result if the transceiver's ips_xfr_wait signal is asserted for an excessively long period of time, which could occur due to mis-programming of the DMA2 controller, and/or the transceiver itself. DMA Timeout forces the transceiver's ips_xfr_wait low after *N* microseconds and sets the **DMA_TIMED_OUT** status bit, where *N*=DMA_TIMEOUT. DMA_TIMEOUT is only relavent when SINGLE_REQ_MODE=1, otherwise the transceiver does not assert ips_xfr_wait. |
| DMA_START_TRG[2:0] | rw | The DMA Start Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| DMA_START_EDGE | rw | DMA_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0: Trigger fires on a rising edge of the selected trigger source<br>1: Trigger fires on a falling edge of the selected trigger source |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| DMA_START_TRIGGERED | r | DMA Start Trigger Occurred. This read-only status bit becomes set, when, during a DMA session (DMA_PAGE > 0), the trigger source selected by DMA_START_TRG[2:0] occurs, with the edge sensitivity selected by DMA_START_EDGE. To clear this bit, set DMA_PAGE=0. |
| DBG_PAGE[3:0] | rw | Packet RAM Debug Page Selector. Selects the page of receiver data for storage to Packet RAM using Transceiver Packet RAM Debug Mode. Setting this register to a non-zero value enables the Packet RAM Debug mode interface logic. Setting this register to zero disables the interface. The available RAM Debug pages are listed in the following Table 64-39 |
| DBG_EN | rw | Packet RAM Debug Mode Enable. Setting DBG_EN=1 enables clock gating to the Packet RAM Debug engine. This bit should be set prior to the start of a Debug session, and remain set for the duration of the session. |
| XCVR_RAM_ALLOW | rw | Allow Packet RAM Transceiver Access. This bit must be set before performing accesses to the Packet RAM using transceiver address space. Transceiver space accesses to Packet RAM are intended for debug purposes only; the individual protocol engines, and the associated IPS busses, have exclusive access to Packet RAM in mission modes. Transceiver space accesses to Packet RAM include direct accesses to RAM at transceiver addresses 0x700 - 0xFFF, as well as Packet RAM Debug Mode. When this bit is set, control of RAM clock gating and RAM chip enables are forced on continuously, taking these controls away from the protocol engines.<br><br>**Note:** In Packet RAM Debug mode, this bit should be set to 1 first, prior to setting DBG_PAGE to any non-zero setting |
| XCVR_RAM_PAGE | rw | This bit selects which of the 2 Packet RAM blocks is mapped into XCVR address space starting at XCVR_BASE + 0x700.<br><br>0: RAM0 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF<br>1: RAM1 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF |
| DBG_AA_TRIGGERED | r | DBG_AA_TRIGGERED. This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of Packet RAM debug transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DBG_PAGE to initiate data transfers to Packet RAM. This bit is intended for use with DBG_PAGE=13, but is available on all pages. Its usage is optional. |
| DBG_RAM_FULL[1:0] | r | Packet RAM Debug RAM Full. Status Bits indicating that Packet RAM0 (or RAM1) is full, and the Packet RAM Debug engine has attempted to write another word to that RAM. This, and any subsequent write attempts, will not alter the contents of the RAM, once it has reached capacity. Set DBG_PAGE=0 to clear the DBG_RAM_FULL[1:0] bits.<br><br>**NOTE:** If a Debug Stop Trigger is selected (DBG_STOP_TRG > 0), writes to RAM do not stop when DBG_RAM_FULL bit(s) become set; instead, writes continue after an address wrap-around to 0, and will continue indefinitely until either a stop trigger condition occurs, or the debug session is ended by |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | taking DBG_PAGE=0. The DBG_RAM_FULL[1:0] remain a valid indicator that the RAM(s) has reached full status at least once during the debug session. |
| DBG_SOFT_INFO_SEL | rw | Packet RAM Debug PHY Soft Info Output Selector. When acquiring data in Packet RAM Debug Mode with the DBG_PAGE=DMDSOFT page, this bit selects the signal from the PHY used to capture the soft decision data.<br>0: PHY output bit_valid_int is used to capture soft decision data<br>1: PHY output demod_vout is used to capture soft decision data |
| DBG_START_TRG[2:0] | rw | The Packet RAM Debug Start Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| DBG_START_EDGE | rw | Packet RAM Debug Start Trigger Edge Selector. DBG_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0: Trigger fires on a rising edge of the selected trigger source<br>1: Trigger fires on a falling edge of the selected trigger source |
| DBG_STOP_TRG[3:0] | rw | Packet RAM Debug Stop Trigger Selector. The Packet RAM Debug Stop Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare3_en<br>8: TSM: pll unlock<br>9: BLE: crc_err_count_incr (increment DTM RX bad packet counter)<br>10: CRC FAIL: 802.15.4 or GEN_FSK only, selected by PROTOCOL[3:0]<br>11: HEADER FAIL: GEN_FSK only<br>12-15: Reserved |
| DBG_STOP_EDGE | rw | DBG_STOP_EDGE selects the edge sensitivity (rising or falling) of the selected stop-trigger.<br>0: Trigger fires on a rising edge of the selected trigger source<br>1: Trigger fires on a falling edge of the selected trigger source |
| DBG_START_TRIGGERED | r | Packet RAM Debug Start Triggered. This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | DBG_START_TRG[2:0] occurs, with the edge sensitivity selected by DBG_START_EDGE. To clear this bit, set DBG_PAGE=0. |
| DBG_STOP_TRIGGERED | r | Packet RAM Debug Stop Triggered. This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by DBG_STOP_TRG[3:0] occurs, with the edge sensitivity selected by DBG_STOP_EDGE. To clear this bit, set DBG_PAGE=0. |

## 64.3.6.3 Functional Description

### 64.3.6.3.1 Transceiver DMA Mode

Tranceiver DMA mode takes internal receive data from the RX_DIG or PHY, packs the data into an optimal format for bandwidth or RAM dimensions, and hands off the data to an external (SoC) DMA controller for subsequent transfer to system memory. Control and configuration of transceiver DMA operation is governed by the bit fields in the DMA_CTRL register of XCVR address space. These bit fields are referenced in the description that follows. In a typical transceiver DMA debug session, the transceiver DMA hardware performs the following steps:

1. Selects a source of internal data from RX_DIG or PHY for capture based on DMA_PAGE[3:0] setting
2. Waits for a hardware start-trigger, if so configured
3. Uses a defined capture signal from RX_DIG or PHY to latch the internal data into a holding register
4. Sequentially packs 2 or more captures of internal data into a 32-bit register until the packing requirements for that DMA_PAGE setting are met
5. Uses the radio's SRS DarkBlue Line-compliant DMA interface to notify SoC DMA controller there is data ready for transfer
6. Repeats steps 3-5 until the debug session is complete (DMA_PAGE set to 0).

Each source of RX_DIG or PHY data has a capture signal associated with it. The capture signal is generated in the source block. The capture signal is a one-reference-clock wide pulse that indicates that the next sample (or samples) of source data is ready for capture. For RX_DIG data sources, the frequency of the capture signal is affected by the RX_DECT_FILT_OSR[2:0] (over-sampling rate) setting. Due to system bandwidth constraints not all RX_DECT_FILT_OSR settings are available, and in some cases the ideal mission mode setting can't be used. The maximum RX_DECT_FILT_OSR setting

is dependent on DMA_PAGE, and is listed in the table below. For any DMA_PAGE selecting source data from RX_DIG, set RX_DIG_CTRL[RX_DMA_DTEST_EN]=1 (in mission mode, this bit is clear to reduce power consumption.)

At the assertion of the capture signal, the transceiver takes the data sample(s) presented by RX_DIG or PHY and stores them in a holding register. Subsequent assertions on the capture signal fill different portions of the holding register, until the packing requirements for the selected DMA_PAGE setting are acheived. A description of the packing requirements for each DMA_PAGE setting is included in the table below. Packing is complete when the page-specific packing requirements are met, and the contents of the holding register are transferred into the DMA_DATA[31:0] register for pickup by the SoC DMA controller.

The DMA_DATA[31:0] is a memory-mapped, read-only IPS bus register. It is available for host access, but its primary function is to be read only by the DMA controller upon request from the radio. Once valid data is stored into the DMA_DATA register, the radio notifies the SoC DMA controller that a new sample is ready using one of 2 methods:

## REQUEST-PER-SAMPLE Mode

In this mode, the radio signals the external DMA controller with a new transfer request whenever a new 32-bit sample word is ready in DMA_DATA[31:0]. This is accomplished by setting SINGLE_REQ_MODE=0. The maximum DMA transfer rate in this mode is 2Msamples/sec. This mode is not optimal from the point-of-view of the DMA controller, because first-cycle-latency is incurred for every sample. The debug session block size (total data to be tranferred) need not be known in advance. To reduce IPS bus latency in this mode, in order to meet the 2Msamples/sec data rate, the following programming restrictions are mandatory:

1. Both radio and MCU must be programmed to run off the same clock source (no clock domain crossing)
2. That clock source must be the Reference Oscillator.
3. Synchronization of the DMA request and done signals must be bypassed, so set BYPASS_DMA_SYNC=1.
4. Synchronization of the IPS bus must be bypassed, so set RSIM_CTRL[GASKET_BYPASS_OVRD_EN]=RSIM_CTRL[GASKET_BYPASS_OVRD]=1.
5. MCU code should be executed out of internal RAM, not flash, since flash access requirements might not be met at this bus frequency.
6. No Radio IPS bus activity by the host is allowed during a DMA session
7. Once DMA debug session is complete, set RSIM_CTRL[GASKET_BYPASS_OVRD_EN]=RSIM_CTRL[GASKET_BYPASS_OVRD]=0.
8. Radio IPS accesses are now permitted

**Note:** From the SoC perspective, steps (1) and (2) put the SoC in a configuration that is a special case where full MCU-to-radio IPS bus timing has not been closed. In this mode, only paths associated with, and required for, DMA transfers, have closed timing. Therefore, no other IPS bus or ordinary mission-mode activity should be attempted in this mode. This restriction means host accesses to all radio registers is prohibited.

In REQUEST-PER-SAMPLE mode, whenever a new sample is ready, the radio asserts **ipd_req_radio_rx** to notify the external DMA controller. The controller will respond to the request, by reading the sample from the DMA_DATA register, and then asserts **ipd_done_radio_rx**. In order to ensure that the 2Msample/second transfer rate is met, the controller must respond with **ipd_done_radio_rx** within 15 reference oscillator clock periods after the radio asserts **ipd_req_radio_rx**. This timing is acheivable if the aforementioned configuration instructions are followed. IC-level simulation of this mode demonstrates that request-to-done timing is a maximum of 13 reference clock periods, after proper configuration. As long as the external controller asserts **ipd_done_radio_rx** no later than 14 reference clock periods after the radio asserts **ipd_req_radio_rx**, the radio shall deassert **ipd_req_radio_rx** on the next reference clock cycle, and hold it deasserted until a new sample is ready. If the external controller asserts **ipd_done_radio_rx** exactly 15 clock periods after the radio asserts **ipd_req_radio_rx**, the radio shall deassert **ipd_req_radio_rx** after the 14th consecutive clock cycle of **ipd_req_radio_rx** assertion, for one clock cycle, in order to generate a low-to-high transition on **ipd_req_radio_rx** on the following clock cycle, when a new sample will be ready for transfer. This is the zero-operating-margin case. If the external controller does not assert **ipd_done_radio_rx** within 15 reference clock periods, there is no guarantee that the controller retrieved the sample from DMA_DATA in time, before the radio updated DMA_DATA to the next sample. This is considered a system malfunction, yielding the possibility that samples have been missed by the controller. It is not a certainty that samples were skipped; it depends on the timing of the DMA IPS bus read relative to the request, but this condition should be avoided.

## SINGLE REQUEST Mode

In Single Request mode, the entire block of desired DMA data, is transferred to system memory based on a single request by the transceiver to the external DMA controller. The block size of the debug session must be known in advance, and this size must be programmed into the DMA controller. In this mode, DMA controller does not incur first-cycle latency on every sample, only on the first sample of a debug session. This is because there is only a single, initial request from the radio. In this mode, the MCU can be configured for any clock source, DMA-signal synchronization is not required (BYPASS_DMA_SYNC=0), and RSIM IPS gasket bypass is also not required. Host IPS accesses to radio registers propagate through the synchronizing IPS bus gaskets as in

mission mode; however IPS bus accesses to transceiver registers during DMA activity should be avoided in order to avoid delaying the controller accesses to the DMA_DATA register.

In this mode, once the radio has its first sample packed into DMA_DATA and ready to transfer to the external controller, it will assert **ipd_req_radio_rx**. The radio will hold **ipd_req_radio_rx** asserted until the first read access to the DMA_DATA register by the external controller. At this point, the radio shall deassert **ipd_req_radio_rx** and hold it low for the remainder of the debug session (until DMA_PAGE is set to 0). The external controller shall transfer the entire block of DMA data based on this single radio request. The radio shall ignore **ipd_done_radio_rx** in this mode. The external controller, operating with reduced latency in this mode, lacking the need to process new requests for every sample, will be able to consume samples from the radio faster than the radio can generate them (i.e., greater than 2Msamples/sec). Thus, the radio will assert **ips_xfr_wait_xcvr** to the controller to pace the transfers. When the external controller reads a sample from DMA_DATA, the radio will assert **ips_xfr_wait_xcvr** on the next reference clock cycle, and keep it asserted until a new sample is available on DMA_DATA; it will then deassert **ips_xfr_wait_xcvr** for 1 clock cycle to allow the controller to access DMA_DATA, then reassert it again. This process iterates until the debug session is complete: the pre-programmed block of data is transferred, and DMA_PAGE is set to 0 by program software.

The IPS SkyBlue signal **ips_xfr_wait** allows a peripheral to insert wait states on the IPS bus for cases in which it cannot deliver readback data (ips_rdata) in the same clock cycle that ips_module_en is asserted. The transceiver DMA interface uses **ips_xfr_wait_xcvr** to throttle the DMA transfers as described above. Because of the potential hazards that can arise due to an extended assertion of **ips_xfr_wait** on any IPS bus, which could result from an incorrectly programmed transceiver or DMA controller, a programmable timeout mechanism has been built in to the transceiver DMA hardware. The timeout mechanism measures the amount of time **ips_xfr_wait_xcvr** is asserted, in microseconds, by the transceiver DMA hardware, and compares it to the contents of the DMA_TIMEOUT[3:0] register, which is also calibrated in microseconds. When they match, the transceiver DMA will force **ips_xfr_wait_xcvr** low, and set the DMA_TIMED_OUT status bit. To clear this bit, disable transceiver DMA by writing DMA_PAGE=0, then write 1 to DMA_TIMED_OUT bit to clear it. Since the timeout condition arose from an erroneously configured transceiver or DMA controller, the data transferred during the DMA session, if any, should be considered invalid. Troubleshoot and rectify the conditions which led to the timeout condition, before initiating a new DMA debug session. DMA timeout applies only to Single Request mode, since **ips_xfr_wait_xcvr** is only asserted in this mode.

## PAGE TABLE

The following table describes the available DMA_PAGE settings. The internal source data selected, the capture signal employed, the data packing method, the OSR setting, and the data rate, are provided for each DMA_PAGE setting.

| DMA PAGE | MNEMONIC | PACKING | OSR | CAPTURE SIGNAL | REQUEST RATE | REMARKS |
|---|---|---|---|---|---|---|
| 0 | DMAIDLE | --- | --- | --- | --- | Transceiver DMA Idle. No DMA Activity. |
| 1 | RXDIGIQ | {4'd0, rx_dig_q[11:0], 4'd0, rx_dig_i[11:0]} | 4 | rx_dig_iq_vld (2MHz) | 2MHz | Q and I data captured. Max supported sample rate ref_osc/16 =2MHz for 32MHz clock. |
| 2 | RXDIGI | {4'd0, rx_dig_i[11:0], 4'd0, rx_dig_i[11:0]} | 2 | rx_dig_iq_vld (4MHz) | 2MHz | I data data captured. Max supported sample rate ref_osc/8 = 4MHz for 32MHz clock. Newer sample is in upper bytes. |
| 3 | RXDIGQ | {4'd0, rx_dig_q[11:0], 4'd0, rx_dig_q[11:0]} | 2 | rx_dig_iq_vld (4MHz) | 2MHz | Q data captured. Max supported sample rate ref_osc/8 = 4MHz for 32MHz clock. Newer sample is in upper bytes. |
| 4 | RAWADCIQ | {rx_dig_q[7:0], rx_dig_i[7:0], rx_dig_q[7:0], rx_dig_i[7:0]} | 2 | rx_dig_iq_vld (4MHz) | 2MHz | Raw ADC capture of I and Q channel but every other I and Q ADC sample are discarded The vld strobe will assert at ref_clock/8 = 4MHz rate. (OSR needs to be programmed for 16). Newest samples are in upper bytes |
| 5 | RAWADCI | {rx_dig_i[7:0], rx_dig_i[7:0], rx_dig_i[7:0], rx_dig_i[7:0]} | 1 | rx_dig_iq_vld (8MHz) | 2MHz | Raw ADC capture of I channel. The vld strobe will assert at ref_clock/4 = 8MHz rate. |

*Table continues on the next page...*

| DMA PAGE | MNEMONIC | PACKING | OSR | CAPTURE SIGNAL | REQUEST RATE | REMARKS |
|---|---|---|---|---|---|---|
| | | | | | | (OSR needs to be programmed for 8). Newest sample is in MSB |
| 6 | RAWADCQ | {rx_dig_q[7:0], rx_dig_q[7:0], rx_dig_q[7:0], rx_dig_q[7:0]} | 1 | rx_dig_iq_vld (8MHz) | 2MHz | Raw ADC capture of Q channel. The vld strobe will assert at ref_clock/4 = 8MHz rate. (OSR needs to be programmed for 8). Newest sample is in MSB |
| 7 | DCESTIQ | {4'd0, Qdc_est[11:0], 4'd0, Idc_est[11:0]} | 4 | IQdc_est_vld (2MHz) | 2MHz | Q and I DCOC tracking estimator data captured. Max supported sample rate ref_osc/16 =2MHz for 32MHz clock |
| 8 | DCESTI | {4'd0, Idc_est[11:0], 4'd0, Idc_est[11:0]} | 2 | IQdc_est_vld (4MHz) | 2MHz | I channel DCOC tracking estimator data captured. Max supported sample rate ref_osc/8 = 4MHz for 32MHz clock. Newer sample is in upper bytes |
| 9 | DCESTQ | {4'd0, Qdc_est[11:0], 4'd0, Qdc_est[11:0]} | 2 | IQdc_est_vld (4MHz) | 2MHz | Q channel DCOC tracking estimator data captured. Max supported sample rate ref_osc/8 = 4MHz for 32MHz clock. Newer sample is in upper bytes |
| 10 | RXINPH | {3'd0,rxin_ph[4:0], 3'd0, rxin_ph[4:0], 3'd0, rxin_ph[4:0], 3'd0, rxin_ph[4:0]} | 1 | rx_dig_ph_vld (8MHz) | 2MHz | Phase data captured. Max sample rate is ref_osc/2 = 8MHz for 32MHz clock. |

*Table continues on the next page...*

| DMA PAGE | MNEMONIC | PACKING | OSR | CAPTURE SIGNAL | REQUEST RATE | REMARKS |
|---|---|---|---|---|---|---|
| | | | | | | Newest sample in MSB |
| 11 | --- | --- | --- | --- | --- | --- |
| 12 | DMDSOFT | {6'd0, soft_info[9:0], 6'd0, soft_info[9:0]} | --- | cg_vbr_en (1MHz) | 0.5MHz | Soft decision output, captured at 1Mb/sec. Newest sample in the MS byte (bit) |
| 13 | DMDDATA | 32 x {data_out} | --- | cg_vbr_en (1MHz) | 31.250KHz | Demodulated bit stream, post-FIFO, captured at 1Mb/sec and packed into a 32-bit register with the newest sample in the MS bit position. |
| 14 | CFOPHASE | {3'd0, cfo_corr_ph_out[4:0], 3'd0, cfo_corr_ph_out[4:0], 3'd0, cfo_corr_ph_out[4:0], 3'd0, cfo_corr_ph_out[4:0]} | --- | rx_dig_ph_vld (8MHz) | 2MHz | CFO-corrected phase, captured at rx_dig_ph_vld (8MHz). Newest sample in MS byte |

## EXAMPLE PROCEDURE

The following steps outline the setup and execution of a Transceiver DMA debug session. This example uses BLE protocol, acquires RX DIG I/Q data, and employs REQUEST-PER-SAMPLE DMA mode:

1. Configure MCU to run off the RF OSC clock source, with divider ratio = 1 (no division)
2. Program RX_DIG for BLE protocol, including RX_DIG_CTRL[RX_DECT_FILT_OSR]=4
3. Set RX_DIG_CTRL[RX_DMA_DTEST_EN]=1
4. Enable clocking to the DMA engine by setting DMA_CTRL[DMA_EN]=1
5. Select a DMA start-trigger source, if desired, by setting DMA_CTRL[DMA_START_TRG] to a non-zero value
6. Program DMA_CTRL[DMA_PAGE]=RXDIGIQ, DMA_CTRL[SINGLE_REQ_MODE]=0, and DMA_CTRL[BYPASS_DMA_SYNC]=1.
7. Set TSM_CTRL[FORCE_RX_EN]=1 to launch an RX sequence.

8. Set RSIM_CTRL[GASKET_BYPASS_OVRD_EN]=RSIM_CTRL[GASKET_BYPASS_OVRD]=1 (XCVR IPS gasket bypass)
9. Begin transmitting RF to the DUT
10. Wait for sufficient RX data to be collected to satisfy the requirements of the debug session
11. Clear RSIM_CTRL[GASKET_BYPASS_OVRD_EN]=RSIM_CTRL[GASKET_BYPASS_OVRD]=0 (XCVR IPS gasket bypass off)
12. Program DMA_CTRL[DMA_PAGE]=0 to terminate the acquisition
13. Download the captured data from system memory

**Note:** Steps 5 (establish the start trigger) and 6 (select the DMA_PAGE to enable the DMA engine), should be performed as sequential, discrete writes to DMA_CTRL, and not combined into a single write.

#### 64.3.6.3.2 Packet RAM Debug Mode

Packet RAM Debug mode takes internal receive data from the RX_DIG or PHY, packs the data into a format suitable for storage in the radio's Packet RAM, and then directly fills the 2 RAMs with source data, under program control. Control and configuration of Packet RAM Debug operation is governed by the bit fields in the PACKET_RAM_CTRL register of XCVR address space. These bit fields are referenced in the description that follows. In a typical Packet RAM debug session, the debug hardware performs the following steps:

1. Selects a source of internal data from RX_DIG or PHY for capture based on DBG_PAGE[3:0] setting
2. Waits for a hardware start-trigger, if so configured
3. Uses a defined capture signal from RX_DIG or PHY to latch the internal data into a holding register
4. Selects a RAM fill method: sequential or simultaneous, based on DBG_PAGE setting
5. Sequentially packs 2 or more captures of internal data into a 16- or 32-bit register until the packing requirements for that DBG_PAGE setting are met
6. Directly accesses the 2 Packet RAM instances to store the receive data (addressing, write enabling, chip enabling, etc.)
7. Repeats steps 3-6 until the debug session is complete (DBG_PAGE set to 0), or a stop-on-trigger event occurs

Each source of RX_DIG or PHY data has a capture signal associated with it. The capture signal is generated in the source block. The capture signal is a one-reference-clock wide pulse that indicates that the next sample (or samples) of source data is ready for capture. For RX_DIG data sources, the frequency of the capture signal is affected by the RX_DECT_FILT_OSR[2:0] setting. In general, there are no restrictions on the OSR setting for Packet RAM debug mode; the RAM fill rate should be able to track with the mission mode OSR settings used by the supported protocols. For any DMA_PAGE selecting source data from RX_DIG, set RX_DIG_CTRL[RX_DMA_DTEST_EN]=1 (in mission mode, this bit is clear to reduce power consumption.)

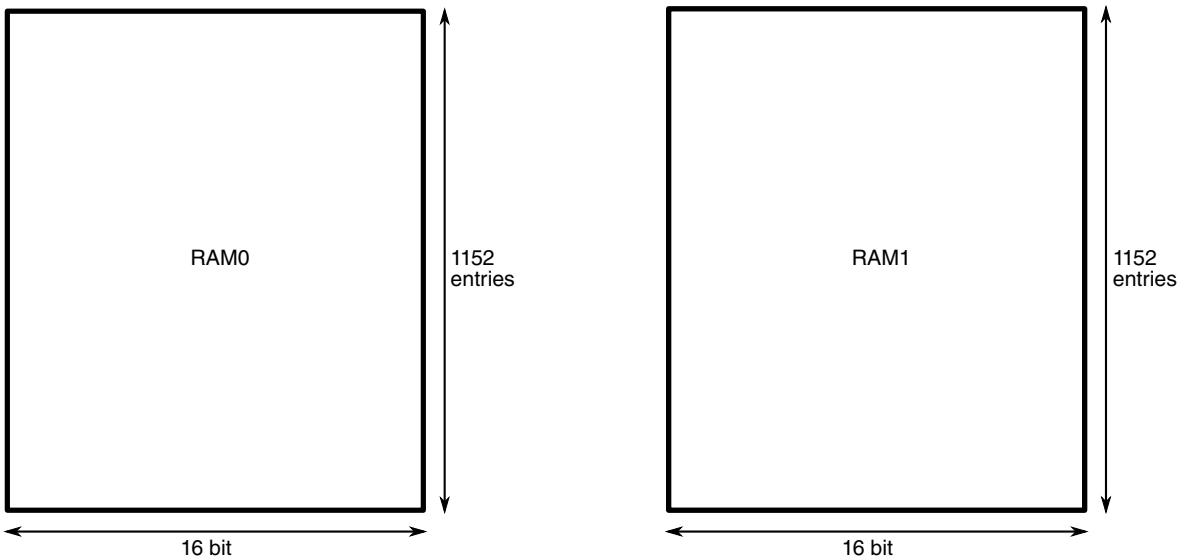
At the assertion of the capture signal, the transceiver takes the data sample(s) presented by RX_DIG or PHY and stores them in a holding register. Subsequent assertions on the capture signal fill different portions of the holding register, until the packing requirements for the selected DBG_PAGE setting are acheived. Packing of the receive data takes into account the RAM fill method employed, based on the DBG_PAGE setting. RAM fill can be sequential (RAM0 fills first completely, followed by RAM1) or simultaneous (both RAMs fill concurrently). A description of the packing requirements and the RAM fill method for each DBG_PAGE setting is included in the table below. Packing is complete when the page-specific packing requirements are met, and the contents of the holding register are written directly to the RAM instance(s).

The bandwidth to Packet RAM is much higher than Transceiver DMA, since the overhead of DMA-related synchronization and multi-master arbitration can be avoided. The maximum transfer rate for RAM debug mode, for a sample of width less than 8 bits (e.g., RAW ADC samples), and simultaneous RAM fill, can be as high as 128Msamples/sec (compared to 2Msamples/sec for DMA). The tradeoff is that the total Packet RAM space is smaller than the space available in system memory, restricting the size of the block of data that can be saved.

## PACKET RAM

The Packet RAM consists of 2 identical RAM blocks, with identical dimensions, as shown in the following diagram.

**RAM DIMENSIONS**
**2304B (RAM0) + 2304B (RAM1) = 4608B (total)**




The RAMs are referred to as RAM0 and RAM1. In mission mode, the RAMs are partitioned into segments to suit the needs of the various protocols, and the sectioning is protocol-dependent. However for the purpose of Packet RAM Debug mode, the entire RAM space is made available for receiver data storage, as one contiguous address space. In Packet RAM Debug mode, the debug engine commandeers both RAMs for buffering receive data, and access attempts by the (mission mode) protocol engines to the RAMs are blocked. For a Packet RAM Debug session, set XCVR_RAM_ALLOW=1. This affords the transceiver IPS bus and the RAM debug engine exclusive access to both RAM blocks comprising the Packet RAM. It also continuously asserts the Chip Enable to both RAMs, and enables a free-running clock to both RAMs, overriding these controls from the protocol link layers.

A Packet RAM debug session consists of 2 stages: Acquisition and Downloading.

## ACQUISITION

In a typical debug session, the selection of a receive data source is made by setting DBG_PAGE to a non-zero value (see the table below for a description of the debug pages). Setting DBG_PAGE to a non-zero value activates the Packet RAM Debug Engine. The engine has write-only access to both RAM's while receive data is being collected. (XCVR host IPS access is not blocked, but should be avoided during acquisition).

If a hardware start-trigger is selected (DBG_START_TRG > 0), the debug engine will wait until the selected triggering event occurs before acquiring any samples; otherwise, acquisition begins as soon as DBG_PAGE is set to a non-zero value.

The RAM's will fill with receive data, either sequentially or simultaneously, determined by the DBG_PAGE setting. If the receive data source selection involves simultaneous capture of symmetric I and Q channel data, RAM0 is dedicated to I channel capture and RAM1 to Q channel; in such cases, both RAMs fill simultaneously. In all other cases, the RAMs fill sequentially: RAM0 fills to capacity, followed by RAM1. RAM addressing always starts at address 0 of either (or both) RAMs; the address for both RAMs is always reset to 0 when DBG_PAGE=0.

**If a stop-trigger is not selected (DBG_STOP_TRG = 0):** When a RAM reaches capacity, further write attempts by the debug engine to that RAM are blocked, and the DBG_RAM_FULL[0] or [1] status bits become set, depending on which RAM is at capacity. When the fill method is simultaneous, both RAMs will reach the full state at the same instant. Otherwise, RAM0 will reach full first. When sufficient receive data has been acquired to satisfy the requirements of the debug session, or when both RAMs are full, the acquisition should be terminated by setting DBG_PAGE=0. The DBG_RAM_FULL[0] and [1] status bits are "sticky"; once set, they will remain set even after DBG_PAGE is set to 0 to end the session. Both bits will self-clear when a new acquisition is started by setting DBG_PAGE to a non-zero value. The XCVR_RAM_ALLOW bit should be set for the duration of the debug session.

**If a stop-trigger is selected (DBG_STOP_TRG > 0):** When a RAM (or both RAMs) reaches capacity, further write attempts by the debug engine to that RAM are not blocked, but the DBG_RAM_FULL[0] or [1] status bits do become set. After capacity is reached on both, or either RAM, depending on the fill method, the debug engine will wrap back to address 0 of the RAM (or RAM pair) and continue filling, until the selected stop-trigger event occurs. When the stop-trigger event occurs, the debug engine will stop filling the RAMs immediately, the DBG_STOP_TRIGGERED status bit will become set, and the registers RAM0_STOP_ADDRESS[10:0] and RAM1_STOP_ADDRESS[10:0] will hold the addresses of the last RAM word, for both RAMs, filled with a valid sample before the trigger. At this point, the acquisition should be terminated by setting DBG_PAGE=0. The DBG_STOP_TRIGGERED and RAMx_STOP_ADDRESS registers are "sticky"; they will hold their state even after DBG_PAGE is set to 0 to end the session. DBG_STOP_TRIGGERED will self-clear, and RAMx_STOP_ADDRESS will reset to 0, when a new acquisition is started by setting DBG_PAGE to a non-zero value. The XCVR_RAM_ALLOW bit should be set for the duration of the debug session.

A diagram of the acquisition process is shown below:

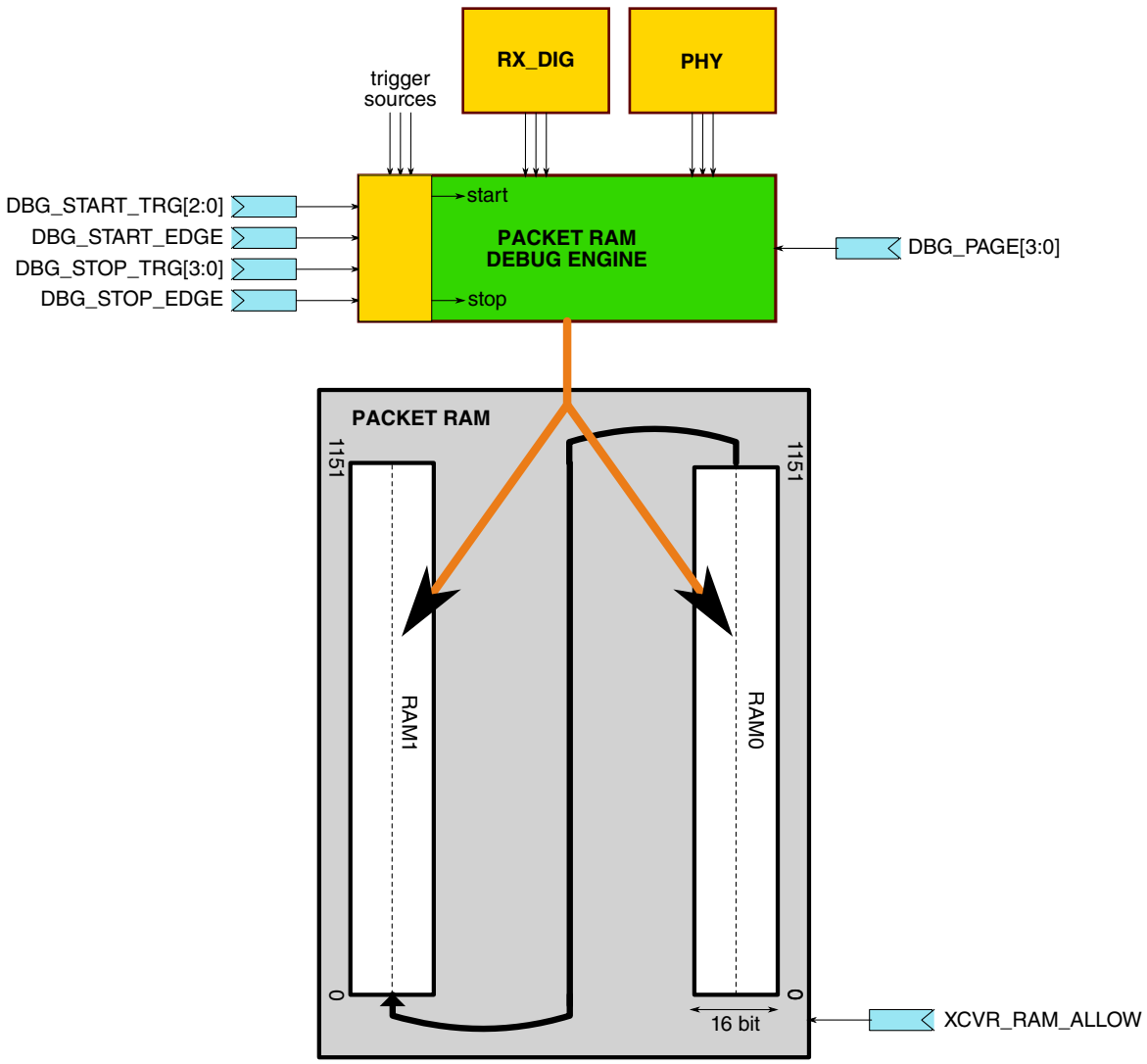



**Figure 64-87. Packet RAM Debug Mode Block Diagram**

## DOWNLOADING

As soon as sufficient data has been collected to satisy the requirements of the debug session, DBG_PAGE should be set to 0 to end the acquisition. The receive data stored in RAM may now be downloaded from the Packet RAM for analysis and post-processing. XCVR_RAM_ALLOW should stay set to allow XCVR IPS access to the RAM. The receive data in the RAM has been packed and distributed between the 2 RAMs in accordance with the DBG_PAGE selection.

Regardless of the DBG_PAGE selection, the host accesses the RAM between addresses XCVR_BASE+0x700 (base of RAM0 or RAM1) and XCVR_BASE+0xFFF (last address of RAM0 or RAM1). Which RAM block gets mapped into this space depends on

the setting of the PACKET_RAM_CTRL[XCVR_RAM_PAGE] bit. Set XCVR_RAM_PAGE=1 to access RAM1 in this space; otherwise RAM0 is accessed in this space.

A diagram depicting how the 2 RAM instances are situated within XCVR address space is shown below.

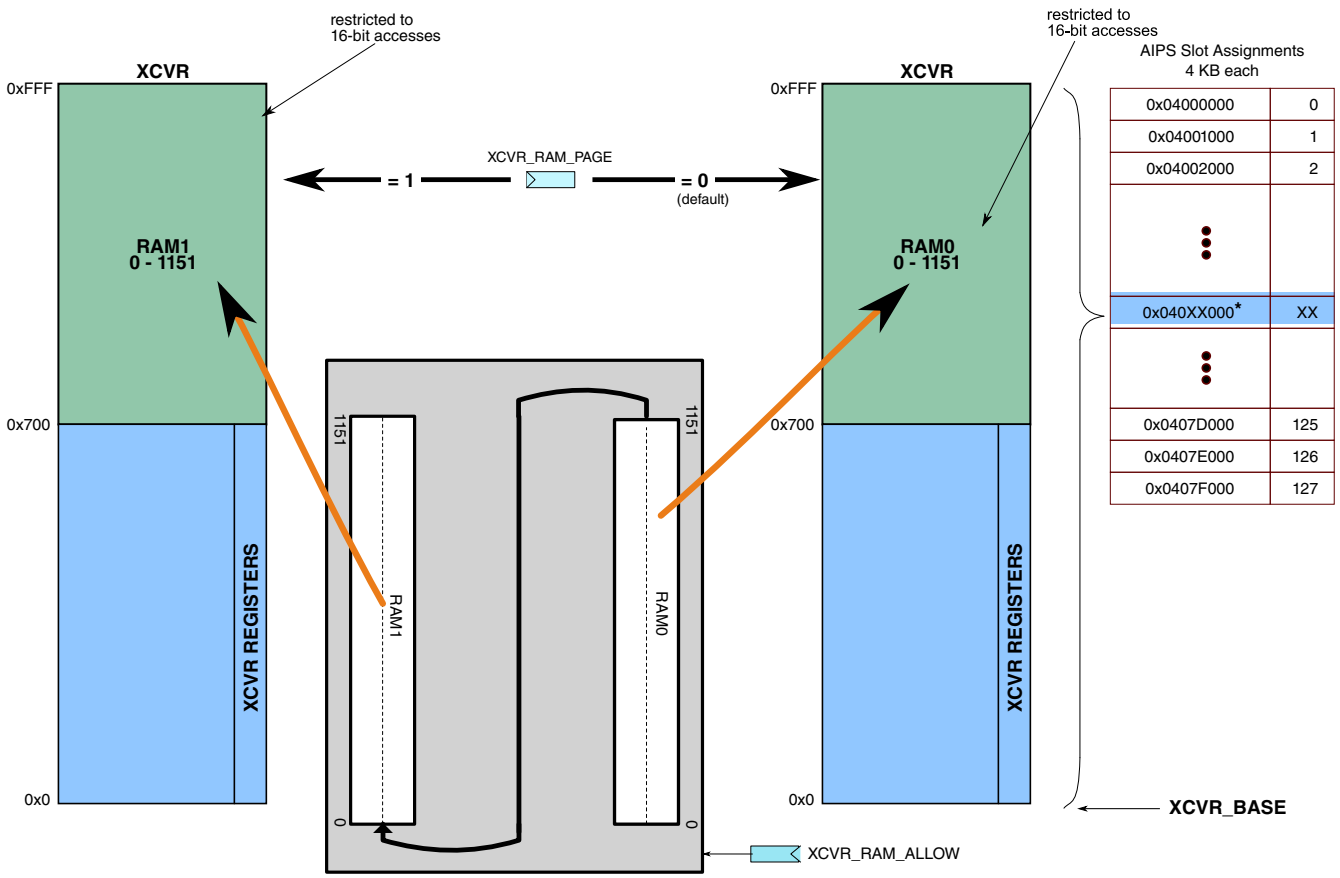



**Figure 64-88. Packet RAM Memory Mapping in XCVR Space**

## PAGE TABLE

The following table describes the available DBG_PAGE settings. In most cases, the DBG_PAGE settings track the Transceiver DMA's DMA_PAGE settings; however DMA_PAGE settings which capture I- and Q-channel-only data are redundant for RAM debug mode, because the bandwidth to packet RAM is high enough to capture I and Q simultaneously with no loss of samples. The internal source data selected, the capture signal employed, the data packing method, and the RAM fill method, and the data rate, are provided for each DBG_PAGE setting.

| DBG PAGE | MNEMONIC | PACKING | CAPTURE SIGNAL | FILL METHOD | REMARKS |
|---|---|---|---|---|---|
| 0 | DBGIDLE | --- | --- | --- | RAM Debug DMA Idle. No RAM Debug Activity. |
| 1 | RXDIGIQ | {4b0, rx_dig_i[11:0]} → RAM0<br>{4b0, rx_dig_q[11:0]} → RAM1 | rx_dig_iq_vld | Simul | Q and I data captured simultaneously. |
| 2 | --- | --- | --- | --- | --- |
| 3 | --- | --- | --- | --- | --- |
| 4 | RAWADCIQ | {rx_dig_i[7:0],rx_dig_i[7:0]} → RAM0<br>{rx_dig_q[7:0],rx_dig_q[7:0]} → RAM1 | rx_dig_iq_vld | Simul | Raw ADC capture of I and Q channel. Newest samples are in upper bytes |
| 5 | --- | --- | --- | --- | --- |
| 6 | --- | --- | --- | --- | --- |
| 7 | DCESTIQ | {4b0, Idc_est[11:0]} → RAM0<br>{4b0, Qdc_est[11:0]} → RAM1 | IQdc_est_vld | Simul | Q and I DCOC tracking estimator data captured. |
| 8 | --- | --- | --- | --- | --- |
| 9 | --- | --- | --- | --- | --- |
| 10 | RXINPH | {3'd0,rxin_ph[4:0], 3'd0,rxin_ph[3:0]} | rx_dig_ph_vld | Sequential | Phase data captured. Newest sample in MSB. Fill RAM0 first with 16 bit words, then fill RAM1 |
| 11 | --- | --- | --- | --- | --- |
| 12 | DMDSOFT | {6'd0, soft_info[9:0], 6'd0, soft_info[9:0]} | cg_vbr_en (1MHz) | Sequential | Soft decision output, captured at 1Mb/sec. Newest sample in the MS byte (bit) |
| 13 | DMDDATA | 16 x {data_out} | cg_vbr_en (1MHz) | Sequential | Demodulated bit stream, post-FIFO, captured at 1Mb/sec and packed into a 16-bit register with the newest sample in the MS bit position. Capturing begins only after AA match *and* the first PHY data_out_valid is asserted. |

*Table continues on the next page...*

| DBG PAGE | MNEMONIC | PACKING | CAPTURE SIGNAL | FILL METHOD | REMARKS |
|---|---|---|---|---|---|
| 14 | CFOPHASE | {3'd0, cfo_corr_ph_out[4:0], 3'd0, cfo_corr_ph_out[4:0]} | rx_dig_ph_vld | Sequential | CFO-corrected phase, captured at rx_dig_ph_vld. Newest sample in MS byte |

## EXAMPLE PROCEDURE

The following steps outline the setup and execution of a Packet RAM debug session. This example uses BLE protocol, and acquires RX DIG I/Q data

1. Program RX_DIG for BLE protocol
2. Set RX_DIG_CTRL[RX_DMA_DTEST_EN]=1
3. Set PACKET_RAM_CTRL[XCVR_RAM_ALLOW]=1 to allow exclusive XCVR access to radio RAMs
4. Enable clocking to the Packet RAM Debug engine by setting PACKET_RAM_CTRL[DBG_EN]=1
5. Select a DBG start-trigger source, if desired, by setting PACKET_RAM_CTRL[DBG_START_TRG] to a non-zero value
6. Select a DBG stop-trigger source, if desired, by setting PACKET_RAM_CTRL[DBG_STOP_TRG] to a non-zero value
7. Set TSM_CTRL[FORCE_RX_EN]=1 to launch an RX sequence.
8. Begin transmitting RF to the DUT
9. Program PACKET_RAM_CTRL[DBG_PAGE]=RXDIGIQ to begin acquisition
10. **If a stop-trigger is selected (DBG_STOP_TRG > 0) --**
11. Wait for DBG_STOP_TRIGGERED=1, indicating a stop-trigger occurred
12. **If a stop-trigger is not selected (DBG_STOP_TRG = 0) --**
13. Wait for sufficient RX data to be collected, or poll DBG_RAM_FULL[0] (and/or DBG_RAM_FULL[1]), to satisfy the requirements of the debug session
14. Program PACKET_RAM_CTRL[DBG_PAGE]=0 to terminate the acquisition
15. Download the captured data from Packet RAM

**Note:** Steps 3, 4, 5, and 6 can be combined into a single write to PACKET_RAM_CTRL.

**Note:** Step 9 (select the DBG_PAGE to enable the DBG engine) should be performed as a sequential, discrete write, after steps 5 and/or 6 (establishing the start/stop triggers) and not combined into a single write to PACKET_RAM_CTRL.

#### 64.3.6.3.3 Simultaneous Transceiver DMA and RAM Debug

Transceiver DMA mode and Packet RAM Debug mode are described in the previous sections. The DMA and Debug engines are separate, independent entities, which theoretically allows both to be employed simultaneously. The most practical case for simultaneous operation would be to select an RX_DIG source for DMA_PAGE and a PHY source for DBG_PAGE. During a such simul debug session, RX_DIG data would go to system memory and PHY data to Packet RAM. After acquisition, data would be downloaded from both memories. Post-processing software could consolidate the data, to take advantage of the synergy that results from combining data captured simultaneously from multiple sources. (DTEST allows a possible third simultaneous source, beyond the scope of this document).

In addition, it is also theoretically possible to combine either or both debug modes with mission-mode packet reception, with the caveat that the Packet RAM would not be available to receive mission-mode packet data in such a scenario. (Packet processing, filtering, CRC checking, and Link Layer interrupts don't rely on packet RAM and so would not be impacted by the debug modes). Such a combination has been demonstrated successfully in radio-level simulation, with the following setup:

1. The GENERIC_FSK Link Layer is engaged in RX mode
2. Transceiver DMA mode is engaged with DMA_PAGE set to RXINPH (RX phase capture)
3. Packet RAM Debug mode is engaged with DBG_PAGE set to DMDSOFT (PHY soft decision data)
4. A GENERIC_FSK packet is received, CRC passes, and GENERIC_FSK RX_IRQ and SEQ_END_IRQ are asserted (GENERIC_FSK packet data not stored)
5. RX phase data is downloaded from system memory and verified
6. Demod soft decision data is downloaded from Packet RAM and verified

#### 64.3.6.3.4 Trigger Sources

**START TRIGGER SOURCES**

Registers DMA_CTRL[DMA_START_TRG] and PACKET_RAM_CTRL[DBG_START_TRG], enable a hardware trigger that must assert before acquisition begins, for the transceiver DMA and Packet RAM debug engines, respectively. The same description applies to both registers (i.e., trigger sources are the same), as indicated in the following table:

| xxx_START_TRG[2:0] | HARDWARE TRIGGER |
|---|---|
| 0 | no trigger |
| 1 | FSK PHY: gfsk_preamble_found |

*Table continues on the next page...*

| xxx_START_TRG[2:0] | HARDWARE TRIGGER |
|---|---|
| 2 | FSK PHY: aa_sfd_matched |
| 3 | ZBDEMOD: rx_preamble_det |
| 4 | ZBDEMOD: rx_sfd_detect |
| 5 | RXDIG: agc_dcoc_gain_chg |
| 6 | TSM: rx_dig_en |
| 7 | TSM: tsm_spare2_en |

## STOP TRIGGER SOURCES

Register PACKET_RAM_CTRL[DBG_STOP_TRG], enables a hardware trigger such that, when it asserts, acquisition terminates immediately, in Packet RAM debug mode, as indicated in the following table:

| DBG_STOP_TRG[3:0] | HARDWARE TRIGGER |
|---|---|
| 0 | no trigger |
| 1 | FSK PHY: gfsk_preamble_found |
| 2 | FSK PHY: aa_sfd_matched |
| 3 | ZBDEMOD: rx_preamble_det |
| 4 | ZBDEMOD: rx_sfd_detect |
| 5 | RXDIG: agc_dcoc_gain_chg |
| 6 | TSM: rx_dig_en |
| 7 | TSM: tsm_spare3_en |
| 8 | TSM: pll unlock |
| 9 | BLE: crc_err_count_incr (increment DTM RX bad packet counter) |
| 10 | CRC FAIL: 802.15.4 or GEN_FSK only, selected by PROTOCOL[3:0] |
| 11 | HEADER FAIL: GEN_FSK only |
| 12-15 | Reserved |

**Note:** For DBG_STOP_TRIG=11 (HEADER FAIL), register GEN_WHITEN_SZ_THR[REC_BAD_PKT] must be set to 0 (default) to cause an RX recycle on a header violation.

### 64.3.6.4 Clocks

The module xcvr_dma includes the Transceiver DMA engine and the Packet RAM Debug engine. Each has its own clock:

1. ipg_clk_dma serves the Transceiver DMA hardware

2. ipg_clk_ram_dbg serves the Packet RAM debug hardware

The **ipg_clk_dma** is a gated 32mhz (or 26mhz) clock from the reference oscillator. This clock is activated when DMA_CTRL[DMA_EN]=1, or when DMA_CTRL[DMA_PAGE] is set to a non-zero value, and deactivated when set to 0. Several other conditions allow this clock to free-run; for example, if the DMA request ipd_req_radio_rx is asserted, or if the DMA_TIMED_OUT status bit is asserted awaiting software clearing, **ipg_clk_dma** will free-run until those conditions are no longer true

The **ipg_clk_ram_dbg** is a gated 32mhz (or 26mhz) clock from the reference oscillator. This clock is activated when PACKET_RAM_CTRL[DBG_EN]=1, or when PACKET_RAM_CTRL[DBG_PAGE] is set to a non-zero value.

These 2 clocks are synchronous to each other. All inputs to xcvr_dma module, including both those destined for the Transceiver DMA engine and the Packet RAM debug engine, are also synchronized to the reference oscillator domain. There are no clock domain crossings in this module.

The DMA DarkBlue line signals can cross clock domain boundaries, when SINGLE_REQ_MODE=1. The clock domain crossings are from the reference oscillator domain to the MCU clock domain for ipd_req_radio_rx, and vice versa for ipd_done_radio_rx. Synchronization, both ways, is accomplished outside the radio, in an SoC level block called radio_dma_sync_glue. These signals cross no clock domains inside the radio itself.

The DMA SkyBlue line signal ips_xfr_wait_xcvr can cross clock domain boundaries, when SINGLE_REQ_MODE=1. The clock domain crossing is from the reference oscillator domain to the MCU clock domain. Synchronization is accomplished in the XCVR IPS bus gasket.

There are no special requirements for these clocks, and no anticipated critical paths associated with them.

### 64.3.6.5 Reset

The xcvr_dma module has a single, active-low, asynchronous reset input: **ipg_hard_async_reset_b** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

### 64.3.6.6 Interrupts

There are no interrupts associated with either Transceiver DMA mode or Packet RAM Debug mode.

## 64.3.7 2.4 GHz PHY

### 64.3.7.1 Introduction

This documents describes the PHY and its operation. More detail and information on the theory of operation can be found in the Architecture Definition Document (ADD).

The Following protocols are supported:

- BLE
- custom FSK modulation schemes
- MSK

The PHY is designed such that it can support a variety of modulation schemes. Several parameters can be adjusted to suit each scheme as described in the configuration section.

#### 64.3.7.1.1 Block diagram

The block diagram below illustrates the basic architecture of the PHY, but several details are not depicted for simplicity.

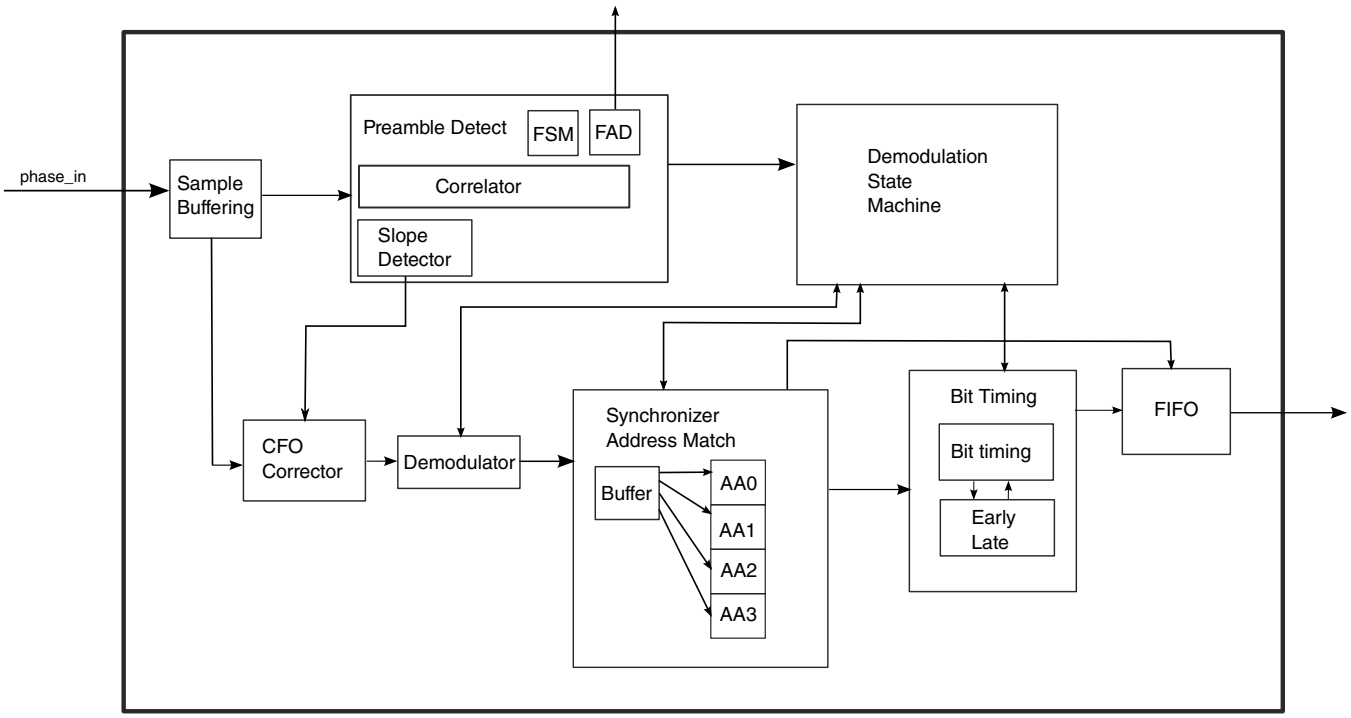



**Figure 64-89. PHY Block diagram**

## 64.3.7.1.2 Overview

The PHY is made up of several sub-modules. These modules can be viewed as members of two classes: datapath (the flow of the signal and the resultant bits) and control (enable/disable logic and state control).

### 64.3.7.1.2.1 PHY Datapath

The signal of interest is passed through the PHY along the data path. The overall objective is to process phase samples that are presented and create bits that are sent out. The significant sub-systems are here presented in generally chronological order as the signal passes through.

Initially, the phase samples are input, sequentially, at the sampling rate, one at a time. The first step is to buffer these samples, so they can be used all at once for certain computations. The buffer size is determined by the largest set of samples that will every be needed by any single component (48 samples).

The buffer of samples is then presented to the preamble detect (PD) logic. The PD module is responsible for detecting the presence of a signal and then indicating the timing and frequency offset when a signal is detected.

In parallel with the PD module, the CFO corrector is presented with the buffered samples. When a preamble is detected, the CFO estimate is presented to the corrector and the correction is applied to the incoming samples. In this fashion, the samples leaving the corrector will be optimally corrected for any frequency offset before beginning demodulation.

Demodulation is performed on frequency corrected samples after a preamble is detected. The produces soft information at the sample rate.

The soft information (soft bits) are then passed to the synchronization block. The primary objective of the synchronizer is to search the soft bits for a match to a sync pattern, typically referred to as an access address (AA). Up to 4 AA's can be searched for. When such a match occurs, the bit timing (the sample offset at which each bit is optimally demodulated) is recorded.

The bit timing block is then able to process the soft bits into hard bits by using the timing information from the synchronizer.

Finally the hard bits are prepared in a FIFO to allow an external block to read as desired. This is realized by a bit timing reference signal that times the output to match.

#### 64.3.7.1.2.2 PHY Control

The control system of the PHY is largely composed of state machines. There are various direct control signals, but the most sophisticated logic is represented in three blocks: demod state machine, preamble detect state machine, and the FAD state machine.

The demodulation state machine has the master control over the entire PHY. When activated, it initiates the preamble search and decides when to activate the demodulator and bit timing engines.

The preamble detect state machine controls the correlator logic and makes the decision when a preamble has been found. Different search mechanisms are used based on the PD_MODE selected.

The fast antenna diversity (FAD) state machine is essentially a replacement of the preamble detect state machine when FAD is enabled. It has the task of detecting the preamble and deciding which antenna to stay on for further processing.

### 64.3.7.2 Interfaces

The PHY interacts with the system in a few distinct ways. The interfaces can be grouped into 4 categories: configuration, signal input, demodulation output, and miscellaneous.

### PHY Configuration Interface

The PHY is configured by static signals that are observed and used throughout the receive process. It is important that these signals (typically originating from IP registers) remain static during a receive operation to avoid unpredictable behavior. Some examples of these types of configurations are PD thresholds, AA to search for, etc.

### Signal Input Interface

The PHY processes the inputs signal from a signed, 5-bit phase inputs. This input is expected to be marked as 'valid' on the phase_in_valid signal. The phase_in_valid signal must be synchronous with input 'clk', but should only assert for 1 clock cycle when a new sample is presented. It is also required that at lease 3 clocks (clk) be present per each phase_in_valid. This is to ensure that all pipe-lined operations can complete.

### Demodulation Output

The demodulator produces bits and, after re-timing, sends them out. The bits are re-timed to the input bit_timing_en which is assumed to be the system bit clock. It is required that the bit_timing_en is synchronous with the input clock and phase_in_valid signals, as synchronization is not performed in the PHY, only re-timing.

### Miscellaneous

There are various other outputs from the PHY that are mostly intended for debug. A few significant signals are described here:

- Soft Info (soft bits): A representation of the demodulator output in fixed-point format. These outputs can be used to see how strong the bits are and the inter-symbol behavior.
- CFO: Frequency offset as determined by the preamble detect engine.
- corr_mag: correlator magnitude, a representation of the correlation strength between the input signal and the reference preamble.
- aa_matched: indicates that an acceptable address synchronization has occurred.
- ant_sw: indicates the requested antenna when FAD is enabled. May be ignored when FAD is not enabled.

## 64.3.7.3 Functional Description

This section describes the functional operation of the PHY.

### 64.3.7.3.1 Protocol Configuration

The PHY has several configurable parameters that can be changed based on the desired protocol behavior. The details of these parameters are described in the following sections and a summary table is provided here for quick reference.

**Table 64-40. FSK schemes supported by the demodulator**

| Scheme | R<br>baud rate (kbps) | *h*<br>modulation index | Frequency pulse | Peak LO error<br>(kHz) |
|---|---|---|---|---|
| **BLE** | 1000 | 0.50 | Gaussian; BT=0.5 | ±300 |
| **Custom scheme 1** | 500 | 0.70 | Gaussian, BT=0.5 | ±200 |
| **Custom scheme 2** | 1000 | 0.50 | Gaussian, BT=0.3 | ±200 |
| **Custom scheme 3** | 500 | 0.50 | Half-sine pulse, L=1 | ±200 |
| **Custom scheme 4** | 250 | 0.50 | Gaussian, BT=0.5 | ±100 |
| **Custom scheme 5** | 250 | 1.00 | Gaussian, BT=0.5 | ±100 |
| **Custom scheme 6** | 1000 | 0.50 | Gaussian, BT=0.7 | ±300 |
| **Custom scheme 7** | 2000 | 0.50 | Gaussian, BT=0.5 | ±300 |
| **Custom scheme 8** | 250 | 0.50 | Half-sine pulse, L=1 | ±100 |
| **Custom scheme 9** | 2000 | 0.30 | Gaussian, BT=0.5 | ±200 |
| **Custom scheme 10** | 500 | 0.50 | Rectangular pulse, L=1 | ±200 |

**Table 64-41. PHY Configurations**

| Config | PD Mode Normal | PD Thresh | PD Mode Comp. | PD Thresh Compressed | AA Corr Gain | PD FREQ THRESH | Preamble Reference {PRE_REF7, PRE_REF6, PRE_REF5, PRE_REF4, PRE_REF3, PRE_REF2, PRE_REF1, PRE_REF0} |
|---|---|---|---|---|---|---|---|
| **BLE** | 2 | 0.8 = 0xCD | 2 | 0.9 = 0xE6 | 0.875 = 0x07 | 0x0A | {0x1e,0x1e,0x00,0x02,0x02,0x02,0x00,0x1e} |
| 3 | 0.9 = 0xE6 | 2 | 0.9375 = 0xF0 | 1.5 = 0x0C | 0x06 | {0x1e, 0x1f, 0x00,0x01,0x02,0x01,0x00,0x1f} | |
| **CS1** | 2 | 0.675 = 0xAD | 2 | 0.9 = 0xE6 | 0.75 = 0x06 | 0x0D | {0x1d,0x1e,0x00,0x02,0x03,0x02,0x00,0x1e} |
| **CS2** | 3 | 0.9 = 0xE6 | 3 | 0.93 = 0xEE | 1.0 = 0x08 | 0x06 | {0x1f,0x1f,0x00,0x01,0x01,0x01,0x00,0x1f} |
| **CS3** | 2 | 0.75 = 0xC0 | 2 | 0.9 = 0xE6 | 0.875 = 0x07 | 0x0D | {0x1d,0x1e,0x00,0x02,0x03,0x02,0x00,0x1e} |
| **CS4** | 2 | 0.8 = 0xCD | 2 | 0.9 = 0xE6 | 0.875 = 0x07 | 0x0D | {0x1e,0x1e,0x00,0x02,0x02,0x02,0x00,0x1e} |
| **CS5** | 2 | 0.6875 = 0xB0 | 2 | 0.8 = 0xCD | 0.875 = 0x07 | 0x0D | {0x1b,0x1d,0x00,0x03,0x05,0x03,0x00,0x1d} |
| **CS6** | 2 | 0.85 = 0xD9 | 2 | 0.9 = 0xE6 | 0.875 = 0x07 | 0x0A | {0x1d,0x1e,0x00,0x02,0x03,0x02,0x00,0x1e} |
| **CS7** | 3 | 0.8 = 0xCD | 2 | 0.93 = 0xEE | 1.25 = 0x0A | 0x05 | {0x1e,0x1e,0x00,0x02,0x02,0x02,0x00,0x1e} |

*Table continues on the next page...*

**Table 64-41. PHY Configurations (continued)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **CS8** | 2 | 0.75 = 0xC0 | 2 | 0.9 = 0xE6 | 1.0 = 0x08 | 0x0D | {0x1d,0x1e,0x00,0x02,0x03,0x02,0x00,0x1e} |
| **CS9** | 3 | 0.9 = 0xE6 | 3 | 0.93 = 0xEE | 2.0 = 0x10 | 0x03 | {0x1f,0x1f,0x00,0x01,0x01,0x01,0x00,0x1f} |
| **CS10** | 2 | 0.82 = 0xD3 | 2 | 0.9 = 0xE6 | 1.0 = 0x08 | 0x0D | {0x1e,0x1e,0x1f,0x01,0x02,0x02,0x01,0x1f} |

#### 64.3.7.3.2 Scan and Preamble Detect

The initial operation of the PHY when it is enabled is in a SCAN state. The purpose of this operation is to scan the input samples for a preamble, so this state is alternately referred to as 'Preamble Detect' or 'PD'. Additionally, when a preamble is detected, the PHY acquires a coarse time reference for the incoming packet. So this process is also sometimes referred to as coarse timing search (CTS).

##### 64.3.7.3.2.1 Correlation

To accomplish preamble detection, a correlator is used to assess the similarity of the incoming phase samples with a reference waveform. The reference waveform is 6 symbols long. The PHY is configured to look for a particular waveform by programming phase values into the PRE_REF*n* IP register fields. However, there are only 8 fields. These represent 2 symbols of 4 samples each. Internally, the PHY replicates this reference to produce a 6 symbol reference. Therefore, it is required that the preamble be repetitive on a 2 symbol basis.

As the phase samples are received by the PHY, they are buffered. The buffer depth is 48 samples so it contains the last 12 potential symbols. From this buffer, samples are selected to compute a frequency offset (CFO) that may be present because of crystal inaccuracy, etc. The number of symbols over which the frequency estimate is computed is determined by the configuration of the PHY. For normal, 6-symbol preamble detect, the frequency estimate is calculated based on the frequency difference over 6 symbols (24 samples). For compressed, 4-symbol mode, the estimate is over 4 symbols (16 samples). For FAD, the estimate is computed over 2 or 4 symbols, depending on the current state of the search. This estimate is continually updated during a preamble search. It is latched when a preamble is declared found.

For clarity, normal, 6-symbol correlation is described, but the other modes operate similarly. As the CFO is calculated, it is applied immediately over the last 24 samples. This is done by generating a phase ramp that would be caused by the frequency offset and then subtracting it from the last 24 samples. This produces a frequency corrected buffer of 24 samples.

The reference waveform is then subtracted from the frequency corrected samples. These differences are converted to the I/Q domain using a look-up table so that a magnitude can be computed. The correlation magnitude is the result of calculating the I/Q mean over the previous 6 symbols and then squaring. The resultant value is referred to as the correlation magnitude or correlation value. It serves as an indicator of how well the previous 6 symbols correlate with the desired reference preamble and updates on every sample.

As previously mentioned, there are 2 distinct modes for the correlator, compressed and normal. The difference is that compressed mode only correlates based on 4 symbols.

#### 64.3.7.3.2.2 Detection

The correlation magnitude provides a measure of how well correlated the incoming signal is to the reference waveform. So, the higher the magnitude, the more likely there is a real preamble present. The magnitude is compared to a threshold that depends on the given protocol and configuration of compressed mode. For every sample that the magnitude is greater than the threshold, a 'pass' event is created. The pass events are processed differently depending on which PD mode is configured.

PD mode 3 is a detection mode that simply looks for the correlation magnitude to exceed the threshold. If it does, then another check is performed 2 symbols (8 samples) later to confirm the finding. If it is confirmed, a preamble is declared 'found' and the CFO estimate is locked for the demodulator.

PD mode 2 also searches for magnitudes that exceed the programmed threshold. However, is also looks for signature peaks. This means that a preamble is only declared 'found' if 3 peaks, above the threshold, and 2 associated 'dips' between the peaks, are observed. This peak and dip behavior is a result of the preamble being negatively correlated when shifted by one symbol.

In either case, when a preamble is declared 'found', a signal is sent to the rest of the PHY that the CFO is to be locked and demodulation can begin for address search and synchronization.

It is important to not that until an address is found, preamble search continues. This can allow for a refinement in the CFO value after initial declaration of a found preamble.

#### 64.3.7.3.2.3 FAD

A simple fast antenna diversity (FAD) mode is provided to allow for 2 antenna scanning. If FAD mode is enabled, the incoming stream of samples is scanned for a preamble just as in the normal mode, but on a significantly reduced scale. The initial operation of FAD checks for correlation over 3 symbols, using a CFO estimate calculated over 2 symbols.

The FAD search switching is configured by setting the FAD_PROC_DUR and FAD_PROC_DLY parameters. The delay parameter allows for antenna switch time and processing delay of the samples. The duration parameter determines the time at which the antenna should switch.

The antenna switch output will continue to toggle until a viable preamble detection occurs. When it does, the antenna will stay in the same state to allow for demodulation. Additionally, regular preamble search is resumed so a refined CFO estimate is possible.

#### 64.3.7.3.3 Demodulation

As the PD logic detects a preamble, the CFO estimate is locked and this estimate is used to correct all phase samples before entering the demodulator. The overall purpose of the demodulator is to process the frequency corrected phase samples and produce soft-bit values at each sample time. This results in a soft-bit stream at 4x the bit/symbol rate.

The corrected phase samples are passed into the demodulator and then buffered. The buffer size is 8, so 2 symbols will fit. The purpose of having 2 symbols is so that a probabilistic estimate can be used to predict the desired bit level.

The output of the demodulator is a signed sfix10n8 soft representation of the demodulated bit. The soft bits are output at the sample rate, so there are 4 soft values per bit. This allows for fractional timing and early/late adjustment elsewhere in the PHY.

#### 64.3.7.3.4 Address Search and Synchronization

When a preamble has been detected, the synchronization block is activated. The purpose of this block is to analyze the demodulated soft bits and look for an Access Address (AA). The PHY supports up to 4 AA's to be searched for and they can vary in size from 1 to 4 octets. When an address is detected, the system is notified, a fine timing estimate of the symbol boundary is produced, and hard bit decisions begin. For each AA to be searched for, an independent engine is enabled so that all 4 can potentially be searched in parallel.

As the soft bits enter the synchronization engine, they are buffered after being multiplied by AA_CORR_GAIN. This gain is protocol dependent to properly scale the soft bits for ideal detection. The buffer size is 128 to allow for the maximum size address to fit (32 bits x 4 samples/bit). The representation of the soft bits is reduced to sfix4n2 in the buffer.

The buffered soft bits are then compared in two ways to the desired AA: hard and soft.

For hard AA detection, the sign bit of the soft bit is used and a hamming distance is calculated using the programmed AA. If the calculated hamming distance is less than or equal to the programmed allowable hamming distance for the particular address, a hard AA match is indicated.

For soft detection, the soft bits are multiplied by the associated desired bit (as a sign) and then all the products are summed. The result is a correlated strength indication.

For a successful AA match to occur, the hamming distance must be met and there must be a peak in the soft correlation. The location of the peak determine the exact position of the AA and the fractional timing.

At this point the system is notified of an AA match and the synchronization machine is shut off.

**AA confirmation**

The synchronization block also contains a special engine that is used to turn off the preamble detector under certain conditions. If enabled, the AA confirm block will look at the soft bits for a sequence of bits that indicates that the AA has begun and will signal the PD engine to shut off. The purpose of this is to reduce the risk of further false preamble detections that may cause an erroneous frequency estimate to be stored. The 6-bit pattern that is searched for depends on where in the AA the first doublet occurs. For example, if the expected bit stream (including the preamble) is [0 1 0 1 0 1 0 1 0 0 1 0 1 1 ....], then the pattern to be confirmed is [0 1 0 1 0 0]. This feature is only available for AA0.

### 64.3.7.3.5 Bit Timing and Production

After an address is synchronized, the PHY begins to produce hard bits, using the demodulated soft bits and the timing information obtained during synchronization.

#### 64.3.7.3.5.1 Bit Production

Bits are produced nominally based on the sign of the soft bit, at the proper sample time. However, for enhanced performance, the PHY performs fractional adjustment to optimally decode the bit. The fractional timing initially provided by the synchronization engine is used to combine soft bits before and after the desired sample time. A weighted summation is performed based on this fraction and then the sign of the result is inverted to produce the bit. Along with the bit, an internal bit valid strobe is generated at one-fourth the sample rate to indicate a valid bit.

#### 64.3.7.3.5.2 Early/Late

The early/late (EL) engine allows for the bit timing module to adjust the timing of the eye of each bit to account for drift. When the EL engine detects that the eye of the bit has moved in time, an adjustment signal is generated and the bit timing is accordingly adjusted. The EL engine is configured for the window over which an EL measurement is taken (EL_WINDOW) and the duration for which each timing adjustment is considered valid (EL_INTERVAL).

#### 64.3.7.3.5.3 Bit Synchronization

Because the relationship between the sample clock and the bit eye is not knowable *a priori*, it is necessary to buffer the bits and then re-time them to the system bit clock. The PHY's version of the bit clock is guaranteed to be synchronous to the system clock as both are divided down from the same reference clock. However the alignment is not certain. Therefore a 16 entry FIFO holds bits as generated by the PHY and then outputs them at the clock edges indicated by the system.

#### 64.3.7.3.5.4 Special Case for Bit Processing: MSK

When MSK mode is enabled there are a couple special processing steps that take place. The desired AA that is programmed is first transformed to an equivalent bit pattern as it would be observed over the air using the FSK demodulator. This allows the synchronizer to properly search for the incoming address. Additionally, the final bit of the address is stored so that it may be used for the bit stream processing.

When a packet is detected in MSK mode, the bits are demodulated normally, then, starting with the last bit of the AA, they are transformed serially into a true MSK bit stream to account for the transformation that is caused by using the FSK demodulator. The actual bits that are presented out of the PHY are, therefore, the expected bits and no further transform should be required.

## 64.3.8 Wake-On-Radio

### 64.3.8.1 Introduction

The 2.4GHz Wake-On-Radio (WOR) Scheduler, is a new hardware automation block, intended to reduce radio and SoC dynamic power consumption, and facilitate radio frequency hopping.

#### 64.3.8.1.1 Overview

The 2.4GHz Radio Wake-On-Radio Scheduler provides hardware automation for the insertion of the radio digital logic into Deep Sleep Mode (DSM), and subsequent DSM exit, as well as the sequencing of RF operations, both TX and RX, between DSM cycles, without MCU intervention. Wake-On-Radio (WOR) substantially increases the autonomy of the radio within the SoC, taking advantage of the separated power domains for the MCU and the Radio in the SoC, by providing sufficient hardware automation to ensure very long MCU low-power intervals, encompassing multiple RF operations, whenever possible. Two different scheduling methods are supported: Timeslot-based scheduling, and Constant-Interval scheduling. WOR also includes hardware automation for RF Frequency Hopping. WOR supports the radio's 802.15.4, and GENERIC_FSK protocol engines, and interfaces to the radio's RSIM (Radio System Integration Module) for the scheduling of DSM entry/exit. WOR divides future RF operations into "timeslots", which can occur at regularly-, or non-regularly-scheduled intervals. WOR provides a descriptor-based format for software to program timeslot information. WOR interacts with the protocol engines to dynamically manage radio interrupts, allowing link layer interrupts to propagate only when it is necessary to notify the MCU of required action.

#### 64.3.8.1.2 Block Diagram

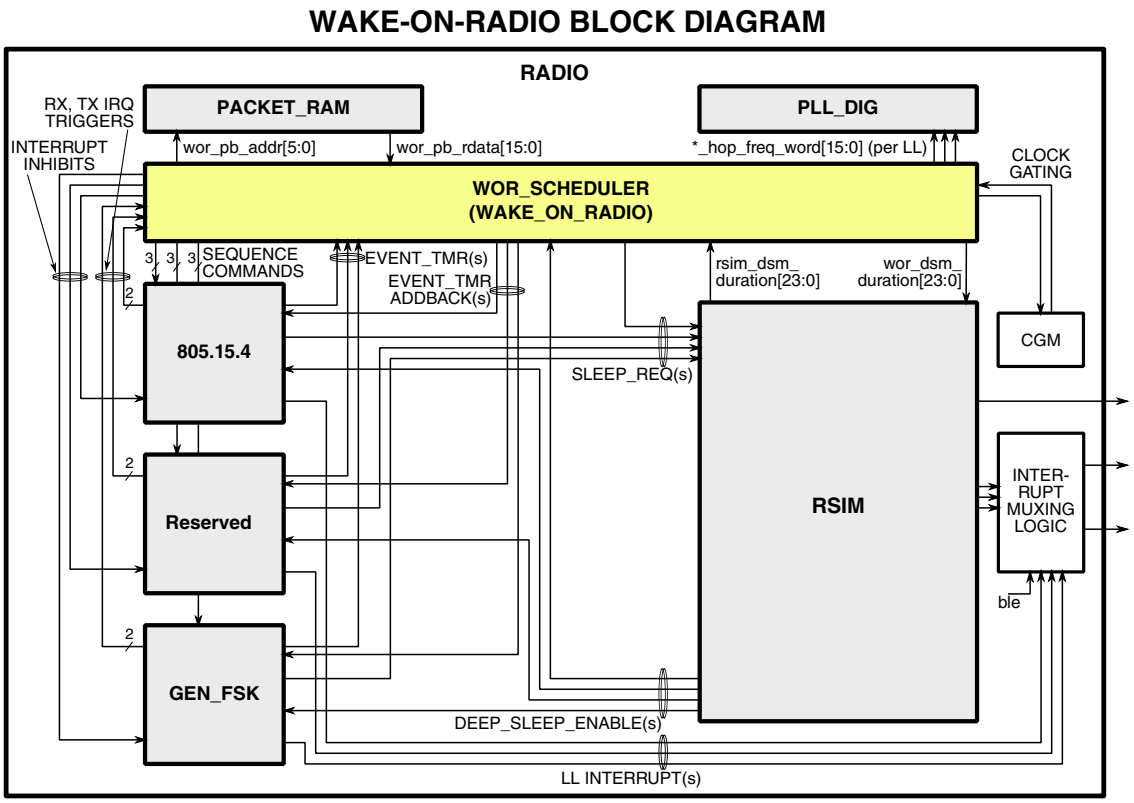



**Figure 64-90. WAKE-ON-RADIO BLOCK DIAGRAM**

### 64.3.8.2 Memory Map and Register Definition

Programmable registers to provide configuration and status for the Wake-On-Radio Scheduler are mapped into Transciever Address Space. WOR registers are described in the table below.

#### 64.3.8.2.1 XCVR_WOR register descriptions

##### 64.3.8.2.1.1 XCVR_WOR_ADDR Memory map

XCVR_WOR base address: 4103_04C0h

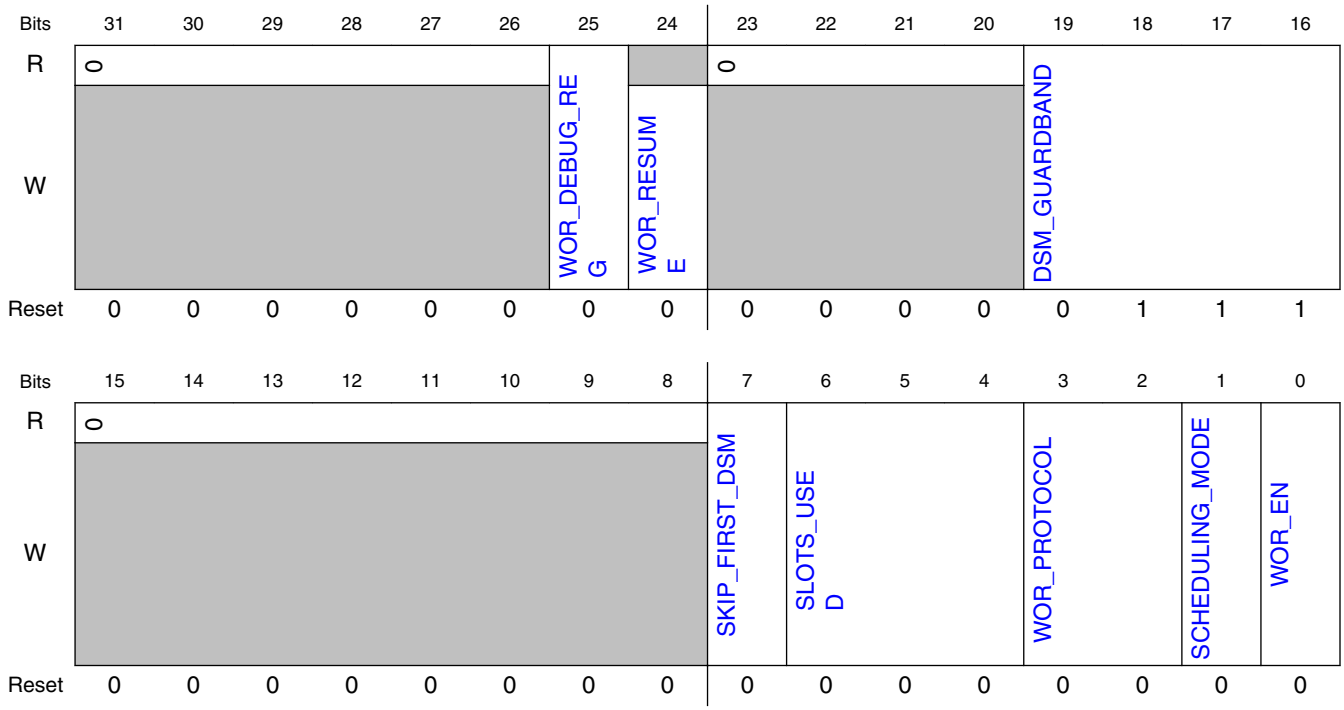
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | WAKE-ON-RADIO CONTROL REGISTER (WOR_CTRL) | 32 | RW | 0007_0000h |
| 4h | WAKE-ON-RADIO TIMEOUT REGISTER (WOR_TIMEOUT) | 32 | RW | 00FF_0000h |
| 8h | WAKE-ON-RADIO TIMESTAMP 1 (TIMESTAMP1) | 32 | RO | 0000_0000h |
| Ch | WAKE-ON-RADIO TIMESTAMP 2 (TIMESTAMP2) | 32 | RO | 0000_0000h |
| 10h | WAKE-ON-RADIO TIMESTAMP 3 (TIMESTAMP3) | 32 | RO | 0000_0000h |
| 14h | WAKE-ON-RADIO STATUS REGISTER (WOR_STATUS) | 32 | RO | 0000_0000h |
| 18h | WINDOW-WIDENING CONTROL REGISTER (WW_CTRL) | 32 | RW | 0000_0002h |
| 1Ch | FREQUENCY HOP CONTROL REGISTER (HOP_CTRL) | 32 | RW | 0030_0000h |
| 20h | SLOT 0 DESCRIPTOR (LSB) (SLOT0_DESC0) | 32 | RW | 0000_0000h |
| 24h | SLOT 0 DESCRIPTOR (MSB) (SLOT0_DESC1) | 32 | RW | 0000_0000h |
| 28h | SLOT 1 DESCRIPTOR (LSB) (SLOT1_DESC0) | 32 | RW | 0000_0000h |
| 2Ch | SLOT 1 DESCRIPTOR (MSB) (SLOT1_DESC1) | 32 | RW | 0000_0000h |
| 30h | SLOT 2 DESCRIPTOR (LSB) (SLOT2_DESC0) | 32 | RW | 0000_0000h |
| 34h | SLOT 2 DESCRIPTOR (MSB) (SLOT2_DESC1) | 32 | RW | 0000_0000h |
| 38h | SLOT 3 DESCRIPTOR (LSB) (SLOT3_DESC0) | 32 | RW | 0000_0000h |
| 3Ch | SLOT 3 DESCRIPTOR (MSB) (SLOT3_DESC1) | 32 | RW | 0000_0000h |

## 64.3.8.2.1.2 WAKE-ON-RADIO CONTROL REGISTER (WOR_CTRL)

### *64.3.8.2.1.2.1 Offset*

| Register | Offset |
|---|---|
| WOR_CTRL | 0h |

### 64.3.8.2.1.2.2 Diagram




### 64.3.8.2.1.2.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>WOR_DEBUG_REG | WAKE-ON-RADIO Debug Register Enable<br>WOR_DEBUG_REG alters the readback values of several read-only registers, for debug purposes. The registers which have a different readback value when WOR_DEBUG_REG=1 are:<br>• TIMESTAMP1<br>• TIMESTAMP2<br>• TIMESTAMP3<br>See the description of those registers, which indicates their nominal, and debug-mode readback values. |
| 24<br>WOR_RESUME | WAKE-ON-RADIO Resume<br>This bit causes the WOR state machine to advance from its WAIT_RESUME state. The state machine will enter WAIT_RESUME on any slot in which an MCU wakeup occurs (based on descriptor WAKE_ON_COMPLETE and/or WAKE_ON_TIMEOUT fields). After waking up (interrupting) the MCU, WOR will hold in WAIT_RESUME, giving software whatever time it needs to complete all required activity. Software then sets WOR_RESUME=1, causing the state machine to break out of WAIT_RESUME, advance to the next slot, and enter DSM. This bit is write-only. Writes of '0' have no effect. Readback value is always 0. Setting WOR_RESUME has no effect if WOR state machine is not in WAIT_RESUME.<br>1: Causes WOR state machine to advance out of WAIT_RESUME state, and begin the next slot<br>0: No effect. |
| 23-20 | Reserved |

*Table continues on the next page...*

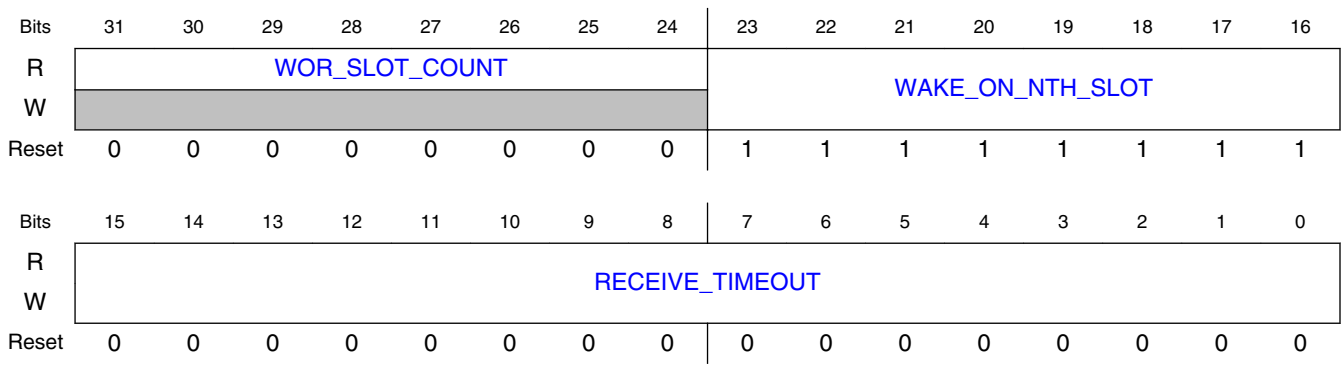
| Field | Function |
|---|---|
| — | |
| 19-16<br>DSM_GUARDBAND | WAKE-ON-RADIO DSM Guardband<br>Determines the number of 32.768KHz clock cycles of guardband used by the WOR state machine in computing duration of the DSM for any slot. Increasing this setting by 1 LSB causes WOR to shorten the DSM duration by 1 32.768KHz clock cycle. This guardband takes into account synchronization, and other, delays within the RSIM, and ensures that DSM will wakeup will never occur "too late" with respect to the slot descriptor's START_TIME field. The current default setting is 7. Values below 7 should not be used. |
| 15-8<br>— | Reserved |
| 7<br>SKIP_FIRST_DSM | WAKE-ON-RADIO Skip DSM On First Slot<br>Setting this bit causes the WOR State Machine to skip the DSM entry/exit cycle for the first SLOT (SLOT 0) encountered after setting SLOTS_USED > 0, regardless of the setting of the NO_DSM field of the slot's descriptor. Subsequent slots, including SLOT 0 when it is re-encountered, will enter/exit DSM based on the descriptor NO_DSM. The SKIP_FIRST_DSM feature is useful, especially in Constant Interval Scheduling mode, when the desire is to begin RF operations immediately (at the start of the first EVENT_INTERVAL), as programmed in the descriptor START_TIME, without waiting out a DSM cycle<br>1: WOR skips DSM entry/exit on the first SLOT 0 after setting SLOTS_USED > 0, regardless of descriptor NO_DSM<br>0: WOR always obeys descriptor NO_DSM<br>**Note:** SKIP_FIRST_DSM should not be used if the SLOT 0 descriptor FUNCTION is set to "NO_RF". |
| 6-4<br>SLOTS_USED | WAKE-ON-RADIO Number Of Slots Used<br>Determines the number of timeslots used to schedule future DSM and RF operations, for either scheduling mode. Up to 4 timeslots can be enabled. Increasing the number of timeslots used decreases average power consumption. Valid values are 0 ≤ SLOTS_USED ≤ 4. The SLOTS_USED register is the mechanism by which the WOR state machine is activated and de-activated. Setting SLOTS_USED to 0 disables the WOR state machine and puts the WOR functionality in an IDLE state. A subsequent write of a non-zero value to SL0TS_USED activates the state machine, initializing to SLOT 0, and starts a new WOR "session". |
| 3-2<br>WOR_PROTOCOL | WAKE-ON-RADIO Protocol Selector<br>Assigns the Wake-On-Radio function to one of the radio's protocol engines.<br>0: Not assigned<br>1: Assigned to 802.15.4<br>2: Assigned to GENERIC_FSK<br>3: Reserved |
| 1<br>SCHEDULING_MODE | WAKE-ON-RADIO Scheduling Mode<br>Determines the scheduling mode used by the WOR HW to organize future DSM and RF operations.<br>1: Constant-Interval Scheduling mode.<br>0: Timeslot Scheduling mode. |
| 0<br>WOR_EN | WAKE-ON-RADIO Enable<br>Wake-On-Radio Master Enable. In addition to enabling the block function, this bit gates the clock to the WOR module. This bit should be set, and remain set, whenever WOR is in use.<br>1: Wake-On-Radio is enabled.<br>0: Wake-On-Radio is disabled, and clocks to the module are gated off. |

# 64.3.8.2.1.3 WAKE-ON-RADIO TIMEOUT REGISTER (WOR_TIMEOUT)

## *64.3.8.2.1.3.1 Offset*

| Register | Offset |
|---|---|
| WOR_TIMEOUT | 4h |

## *64.3.8.2.1.3.2 Diagram*




## *64.3.8.2.1.3.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>WOR_SLOT_COUNT | WAKE-ON-RADIO Absolute Slot Count<br>This read-only register indicates the number of slots that have elapsed since SLOTS_USED was set to a non-zero value to activate the WOR state machine. WOR_SLOT_COUNT can be considered an Absolute Slot Number. WOR_SLOT_COUNT counts slots up to a maximum of 255. If the number of elapsed slots exceeds 255, WOR_SLOT_COUNT will remain at 255. WOR_SLOT_COUNT is "sticky" in the sense that it will hold its value after the WOR session has ended by setting SLOTS_USED=0; WOR_SLOT_COUNT will clear to 0 when a new WOR session is started by taking SLOTS_USED from 0 to any non-zero value. |
| 23-16<br>WAKE_ON_NTH_SLOT | WAKE-ON-RADIO Force Wake On Nth Slot<br>If the WAKE_ON_COMPLETE bit is not set in any descriptor, WOR will not awaken the MCU even after a number of good packets are received. Additionally, if WAKE_ON_TIMEOUT is also not set in any descriptor, WOR will not wake up the MCU under any circumstances, which would not be desirable. This register provides a backup method to awaken the MCU, after *N+1* slots have elapsed, where *N* = WAKE_ON_NTH_SLOT[7:0], regardless of whether or not any good packets have been received, and regardless of the setting of the WAKE_ON_* descriptor bits. Applies to Constant-Interval Mode only. (For Timeslot mode, at least one slot *must* have a WAKE_ON_* bit set in its descriptor). |
| 15-0<br>RECEIVE_TIMEOUT | WAKE-ON-RADIO Receive Timeout<br>For any RX slot, determines how long RX stays on after the descriptor's START_TIME is reached, in microseconds. A timeout is declared if a good packet is not received before the expiration of this time period (START_TIME + RECEIVE_TIMEOUT + WINDOW_WIDENING). Applies to TIMESLOT mode only. |

## 64.3.8.2.1.4 WAKE-ON-RADIO TIMESTAMP 1 (TIMESTAMP1)

### 64.3.8.2.1.4.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP1 | 8h |

### 64.3.8.2.1.4.2 Function

Timestamp captured for a packet received on SLOT 1. During SLOT 1, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP1 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 1.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WOR_ADDBACK[27:0], which is the number of microseconds that the WOR state machine "added back" to the assigned Link Layer's Event Timer, to correct for the amount of time that Link Layer was in DSM (Event Timer frozen). The WOR_ADDBACK readback value applies to the most recently-completed DSM cycle.

### 64.3.8.2.1.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP1 | | | | | | | | TIMESTAMP1_FRAC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.8.2.1.4.4 Fields

| Field | Function |
|---|---|
| 31-8 | WAKE-ON-RADIO TIMESTAMP1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TIMESTAMP1 | Timestamp captured for a packet received on SLOT 1. During SLOT 1, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP1 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 1. |
| 7-0<br>TIMESTAMP1_FRAC | WAKE-ON-RADIO TIMESTAMP 1 Fractional<br>TIMESTAMP1_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP1_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional component has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP1_FRAC[3:0], shall always read back '0000'. |

## 64.3.8.2.1.5 WAKE-ON-RADIO TIMESTAMP 2 (TIMESTAMP2)

### *64.3.8.2.1.5.1 Offset*

| Register | Offset |
|---|---|
| TIMESTAMP2 | Ch |

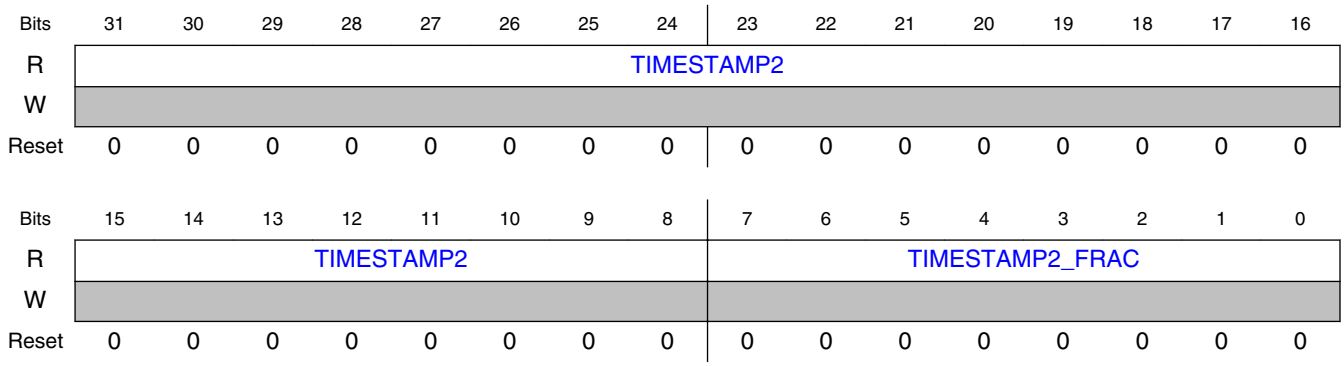
### *64.3.8.2.1.5.2 Function*

Timestamp captured for a packet received on SLOT 2. During SLOT 2, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP2 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 2.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WW_DSM_ADJUST[19:0], which is the accumulated DSM component of the Window Widening adjustment value that is used to offset the start and end of receive operations during the timeslot.

#### 64.3.8.2.1.5.3 Diagram




#### 64.3.8.2.1.5.4 Fields

| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP2 | WAKE-ON-RADIO TIMESTAMP2<br>Timestamp captured for a packet received on SLOT 2. During SLOT 2, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP2 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 2. |
| 7-0<br>TIMESTAMP2_FRAC | WAKE-ON-RADIO TIMESTAMP 2 Fractional<br>TIMESTAMP2_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP2_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP2_FRAC[3:0], shall always read back '0000'. |

### 64.3.8.2.1.6 WAKE-ON-RADIO TIMESTAMP 3 (TIMESTAMP3)

#### 64.3.8.2.1.6.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP3 | 10h |

#### 64.3.8.2.1.6.2 Function

Timestamp captured for a packet received on SLOT 3. During SLOT 3, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into

this register at the instant of detection (there is no offset applied). TIMESTAMP3 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 3.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WW_RUN_ADJUST[19:0], which is the accumulated "run time" component of the Window Widening adjustment value that is used to offset the start and end of receive operations during the timeslot.

#### *64.3.8.2.1.6.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP3 | | | | | | | | TIMESTAMP3_FRAC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.8.2.1.6.4 Fields*

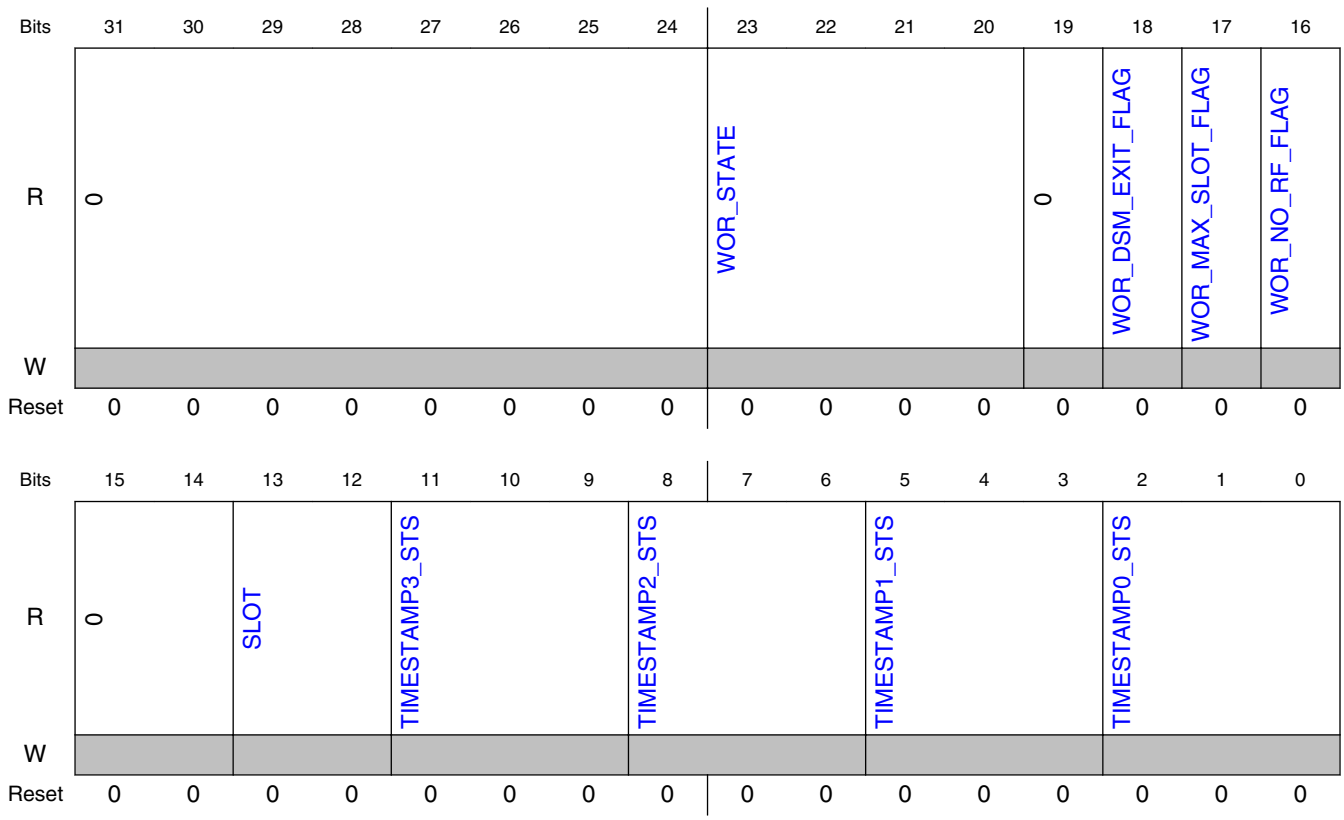
| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP3 | WAKE-ON-RADIO TIMESTAMP3<br>Timestamp captured for a packet received on SLOT 3. During SLOT 3, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP3 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 3. |
| 7-0<br>TIMESTAMP3_FRAC | WAKE-ON-RADIO TIMESTAMP 3 Fractional<br>TIMESTAMP3_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP3_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP3_FRAC[3:0], shall always read back '0000'. |

### 64.3.8.2.1.7 WAKE-ON-RADIO STATUS REGISTER (WOR_STATUS)

### 64.3.8.2.1.7.1 Offset

| Register | Offset |
|---|---|
| WOR_STATUS | 14h |

### 64.3.8.2.1.7.2 Diagram




### 64.3.8.2.1.7.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20<br>WOR_STATE | WAKE-ON-RADIO Current State<br>This read-only register reflects the instantaneous state of the Wake-On-Radio state vector. A listing of the Wake-On-Radio states, along with the associated state vector, is below:<br>• DISABLED (0)<br>• COMP_DURATION1 (1)<br>• COMP_DURATION2 (2)<br>• SLEEP_REQ (3)<br>• IN_DSM (4) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • COMP_ADDBACK (5)<br>• COMP_WW (6)<br>• WAIT_SEQ_START (7)<br>• WAIT_SEQ_END (8)<br>• WAIT_RESUME (9) |
| 19<br>—<br> | Reserved |
| 18<br>WOR_DSM_EXIT_FLAG | WAKE-ON-RADIO Early DSM Exit Flag<br>This read-only flag indicates that, in the current slot, an event external to the radio has forced an early exit from DSM, before the WOR-computed DSM duration has expired. The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to an "DSM Early Exit wakeup", as described by the conditions above. The WOR_IRQ will become set, and the MCU awakened, even if the slot descriptor's WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT bits are both set to 0. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "DSM Early Exit wakeup", software must set WOR_RESUME to advance to the next slot. |
| 17<br>WOR_MAX_SLOT_FLAG | WAKE-ON-RADIO Maximum Slot Count Reached Flag<br>This read-only flag indicates that, in the current slot, the following conditions have all been met:<br>1. SCHEDULING_MODE is set to 1 (Constant Interval)<br>2. WOR_SLOT_COUNT[7:0] (absolute slot number) has matched WAKE_ON_NTH_SLOT[7:0]<br><br>The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to a "Maximum Slot Reached wakeup", as described by the conditions above. The WOR_IRQ will become set, and the MCU awakened, even if the slot descriptor's WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT bits are both set to 0. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "Maximum Slot Reached wakeup", software must set WOR_RESUME to advance to the next slot |
| 16<br>WOR_NO_RF_FLAG | WAKE-ON-RADIO NO_RF Slot Flag<br>This read-only flag indicates that, in the current slot, the following conditions have all been met:<br>1. The slot's descriptor FUNCTION is set to "NO_RF"<br>2. A DSM cycle has just completed (exited) during the slot<br>3. The slot's descriptor WAKE_ON_COMPLETE bit was set (or the WOR_SLOT_COUNT matched WAKE_ON_NTH_SLOT), requiring an MCU wakeup.<br><br>The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to a "NO_RF wakeup", as described by the conditions above. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "NO_RF wakeup", software must set WOR_RESUME to advance to the next slot |
| 15-14<br>—<br> | Reserved |
| 13-12<br>SLOT | WAKE-ON-RADIO Current Slot<br>This read-only register reflects the instantaneous state of the WOR's SLOT counter, indicating which SLOT is currently in service by the WOR state machine. Software can determine which of the 4 descriptors is currently in effect based on this register. |
| 11-9<br>TIMESTAMP3_STS | WAKE-ON-RADIO Timestamp 3 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP3 register contents: |

*Table continues on the next page...*

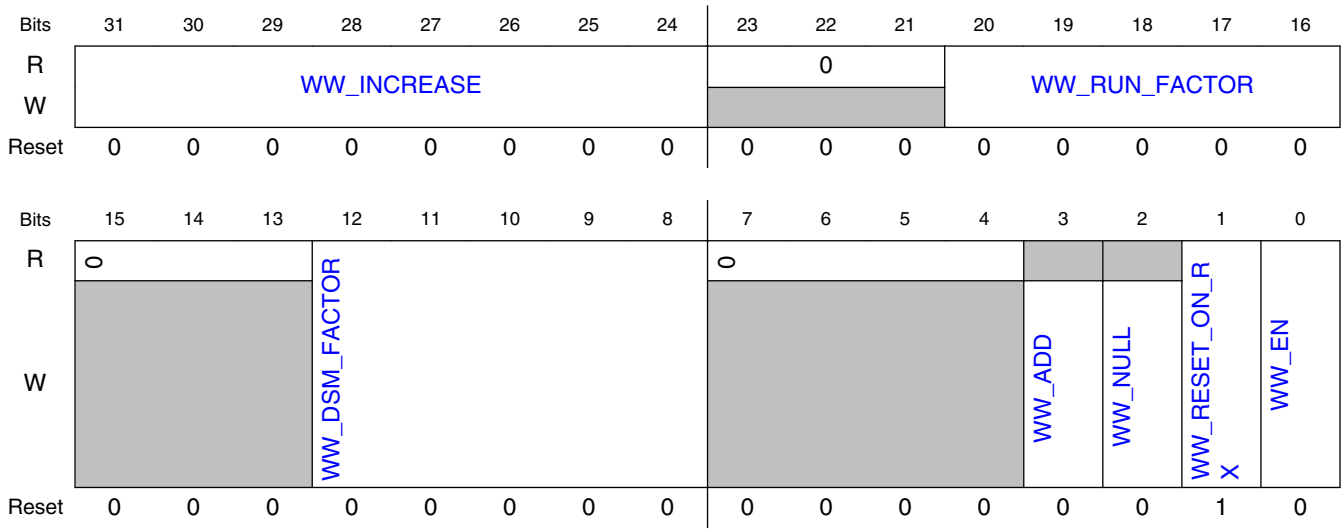
| Field | Function |
|---|---|
| | **TIMESTAMP3_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 3 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP3 register.<br>**TIMESTAMP3_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 3 RX operation<br>**TIMESTAMP3_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 3 RX operation |
| 8-6<br>TIMESTAMP2_STS | WAKE-ON-RADIO Timestamp 2 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP2 register contents:<br>**TIMESTAMP2_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 2 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP2 register.<br>**TIMESTAMP2_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 2 RX operation<br>**TIMESTAMP2_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 2 RX operation |
| 5-3<br>TIMESTAMP1_STS | WAKE-ON-RADIO Timestamp 1 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP1 register contents:<br>**TIMESTAMP1_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 1 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP1 register.<br>**TIMESTAMP1_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 1 RX operation<br>**TIMESTAMP1_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 1 RX operation |
| 2-0<br>TIMESTAMP0_STS | WAKE-ON-RADIO Timestamp 0 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP0 register contents:<br>**TIMESTAMP0_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 0 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP0 register.<br>**TIMESTAMP0_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 0 RX operation<br>**TIMESTAMP0_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 0 RX operation<br>**Note:** TIMESTAMP0 register (timestamp for SLOT 0) is captured into the assigned Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space. However TIMESTAMP0_STS[2:0], which applies to that register, resides in WOR (XCVR) address space. |

## 64.3.8.2.1.8 WINDOW-WIDENING CONTROL REGISTER (WW_CTRL)

### *64.3.8.2.1.8.1 Offset*

| Register | Offset |
|---|---|
| WW_CTRL | 18h |

### 64.3.8.2.1.8.2 Diagram




### 64.3.8.2.1.8.3 Fields

| Field | Function |
|---|---|
| 31-24<br>WW_INCREASE | Window-widening Manual Increase Amount<br>This register allows software to increase (add to) the HW-computed window-widening adjustment. This incremental amount is added whenever a 1 is written to the WW_ADD register bit. The units of WW_INCREASE[7:0] are microseconds. It is an unsigned (positive only) addition. This is a one-time only addition (whever WW_ADD is set), and the window-widening adjustment value continues to accumulate after the increment. The capability to add to the HW-computed adjustment is intended for contingency or debug purposes only, as the HW-computed value should be accurate. |
| 23-21<br>— | Reserved |
| 20-16<br>WW_RUN_FACTOR | Window-widening Runtime Factor<br>Determines the rate of Window-widening that applies during RUNTIME (Reference Oscillator On). This setting should be based on the combined Sleep Clock Accuracy (SCA) of the Master + Slave. Each LSB is worth 64ppm of SCA drift. Valid values are 0 -> 16. The lower this factor, the greater the power savings. |
| 15-13<br>— | Reserved |
| 12-8<br>WW_DSM_FACTOR | Window-widening DSM Factor<br>WW_DSM_FACTOR[4:0]: Determines the rate of Window-widening that applies during DSM (Reference Oscillator Off). This setting should be based on the combined Sleep Clock Accuracy (SCA) of the Master + Slave. Each LSB is worth 64ppm of combined SCA drift. Valid values are 0 -> 16. The lower this factor, the greater the power savings. |
| 7-4<br>— | Reserved |
| 3 | Window-widening Add Command |

*Table continues on the next page...*

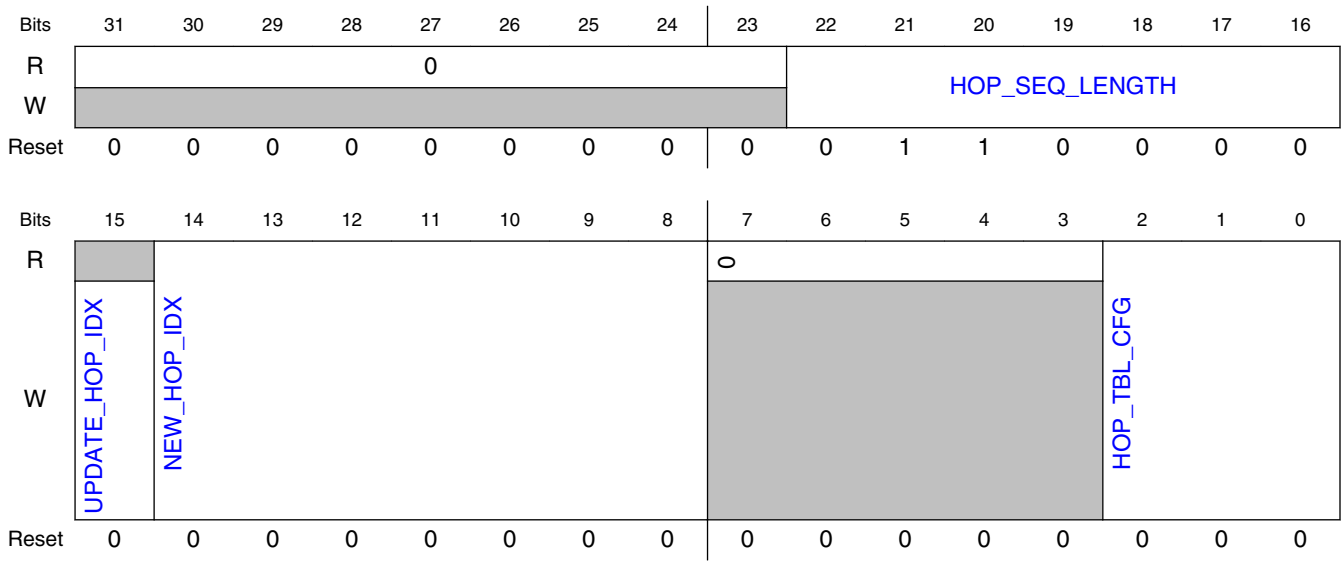
| Field | Function |
|---|---|
| WW_ADD | This bit allows software to increase (add to) the HW-computed window-widening adjustment. The incremental amount is determined by the WW_INCREASE[7:0] register. The capability to add to the HW-computed adjustment is intended for contingency or debug purposes only, as the HW-computed value should be accurate. This bit is write-only. Readback value is always 0.<br>1: Increase the HW-computed window-widening adjustment by amount WW_INCREASE[7:0].<br>0: No effect |
| 2<br>WW_NULL | Window-widening Null Command<br>This bit allows software to reset the accumulated window-widening adjustment to 0. This bit is write-only. Readback value is always 0.<br>1: Reset the accumulated window-widening adjustment to 0.<br>0: No effect. |
| 1<br>WW_RESET_ON_RX | Window-widening Reset on Received Good Packet<br>Enable a good packet reception to reset the accumulated window-widening adjustment to 0. A good packet is one that passes CRC verification, and passes packet filtering rules. Filtering rules vary by protocol.<br>1: Allow good packet reception to reset window-widening adjustment. This is the default.<br>0: Block good packet reception from resetting window-widening adjustment. |
| 0<br>WW_EN | Window-widening Enable<br>Master Enable for Window Widening. Typically, only slave devices employ window widening.<br>1: Window Widening is enabled.<br>0: Window Widening is disabled. |

## 64.3.8.2.1.9 FREQUENCY HOP CONTROL REGISTER (HOP_CTRL)

### *64.3.8.2.1.9.1 Offset*

| Register | Offset |
|---|---|
| HOP_CTRL | 1Ch |

### 64.3.8.2.1.9.2 Diagram




### 64.3.8.2.1.9.3 Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22-16<br>HOP_SEQ_LENGTH | New Hop Table Index<br>The HOP_SEQ_LENGTH[6:0] register determines the number of entries in the WOR Hop Table, regardless of how the table is configured by HOP_TBL_CFG[2:0]. During WOR operation, once the Hop Table Index reaches HOP_SEQ_LENGTH - 1, the next Hop Table Index will wrap to 0. HOP_SEQ_LENGTH must not exceed the table size specified by HOP_TBL_CFG (48 or 96 entries). |
| 15<br>UPDATE_HOP_IDX | Update Hop Table Index<br>Software can jump to a new index (read pointer address) of the WOR Hop Table at any time by writing the desired index to the NEW_HOP_IDX[6:0] register, and setting the UPDATE_HOP_IDX bit to 1. This bit is write-only. Readback value is always 0.<br>1: Jump to new Hop Table Index pointed to by NEW_HOP_IDX[6:0]<br>0: No effect |
| 14-8<br>NEW_HOP_IDX | New Hop Table Index<br>Software can jump to a new index (read pointer address) of the WOR Hop Table at any time by writing the desired index to this register, and setting the UPDATE_HOP_IDX bit. |
| 7-3<br>— | Reserved |
| 2-0<br>HOP_TBL_CFG | Hop Table Configuration<br>For maximum flexibility in Frequency Hop scheduling, 4 hop table configurations are defined. The configurations affect:<br>1. The source of the RF channel information (Link Layer register or WOR Hop Table)<br>2. How the RF channel information is organized in the WOR Hop Table<br>3. How the hop frequency word is interpreted by the PLL Digital Block |

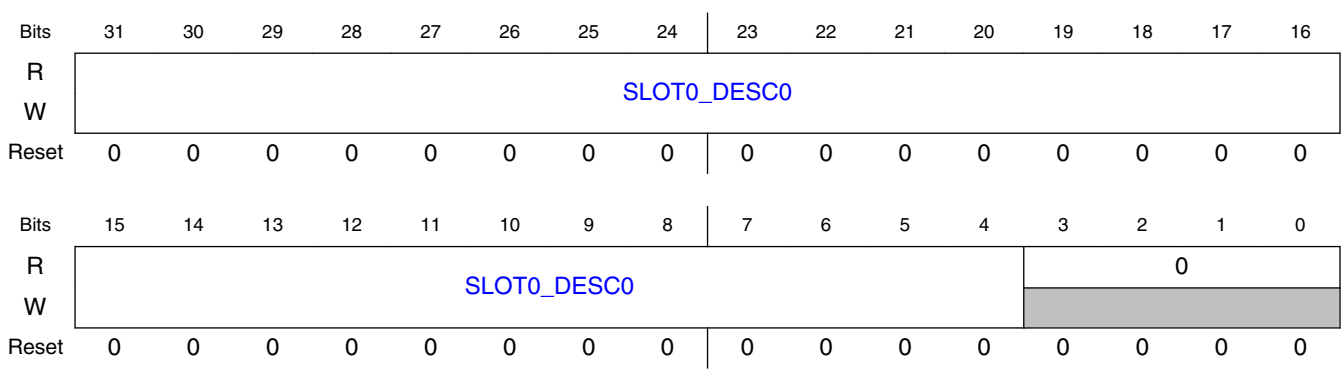
| Field | Function |
|---|---|
| | Source of RF channel information:<br>0: Link Layer CHANNEL_NUM register (legacy configuration). The Link Layer is the one assigned by WOR_PROTOCOL register.<br>1: WOR Hop Table<br>2: WOR Hop Table<br>3: WOR Hop Table<br>4-7: Reserved<br><br>Hop Table Organization:<br>0: N/A<br>1: Table size is 96 entries. Two 7-bit CHANNEL_NUM fields are packed into a 16-bit hop frequency word for RAM storage<br>2-3: Table size is 48 entries. Each 16-bit hop frequency word in RAM contains a specification for a single RF channel<br>4-7: Reserved<br><br>PLL Digital Interpretation of Hop Frequency Word:<br>0: Lower 7 bits of hop frequency word contain a 7-bit mapped channel (legacy format).<br>1: Lower 7 bits of hop frequency word contain a 7-bit mapped channel (legacy format).<br>2-3: See Chapter *PLL Digital*<br>4-7: Reserved |

## 64.3.8.2.1.10 SLOT 0 DESCRIPTOR (LSB) (SLOT0_DESC0)

### *64.3.8.2.1.10.1 Offset*

| Register | Offset |
|---|---|
| SLOT0_DESC0 | 20h |

### *64.3.8.2.1.10.2 Diagram*



#### 64.3.8.2.1.10.3 Fields

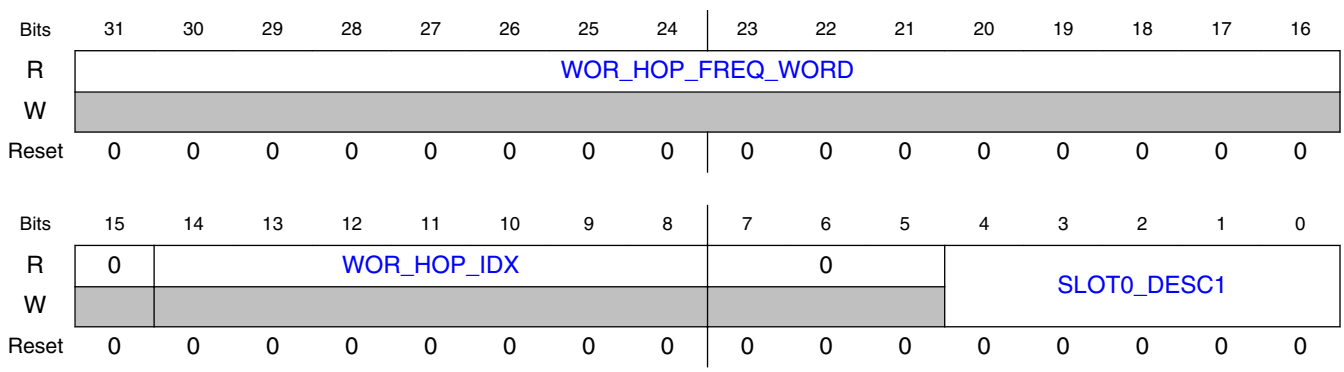
| Field | Function |
|---|---|
| 31-4<br>SLOT0_DESC0 | Slot 0 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 0 |
| 3-0<br>— | Reserved |

## 64.3.8.2.1.11 SLOT 0 DESCRIPTOR (MSB) (SLOT0_DESC1)

#### 64.3.8.2.1.11.1 Offset

| Register | Offset |
|---|---|
| SLOT0_DESC1 | 24h |

#### 64.3.8.2.1.11.2 Diagram




#### 64.3.8.2.1.11.3 Fields

| Field | Function |
|---|---|
| 31-16<br>WOR_HOP_FREQ_WORD | Current Hop Frequency Word<br>This read-only register reflects the current Hop Frequency Word, formulated by WOR and transmitted to the PLL Digital Block |
| 15<br>— | Reserved |
| 14-8<br>WOR_HOP_IDX | Current Hop Table Index<br>This read-only register reflects the current Hop Table Index (read address pointer) into the WOR Hop Table |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4-0<br>SLOT0_DESC1 | Slot 0 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 0 |

## 64.3.8.2.1.12 SLOT 1 DESCRIPTOR (LSB) (SLOT1_DESC0)

### *64.3.8.2.1.12.1 Offset*

| Register | Offset |
|---|---|
| SLOT1_DESC0 | 28h |

### *64.3.8.2.1.12.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLOT1_DESC0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLOT1_DESC0 | | | | | | | | | | | | 0 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.3.8.2.1.12.3 Fields*

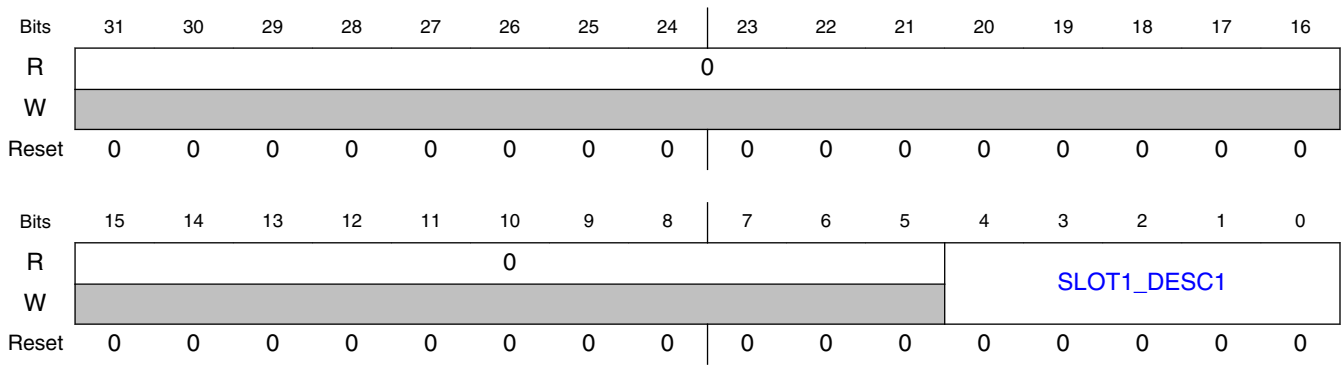
| Field | Function |
|---|---|
| 31-4<br>SLOT1_DESC0 | Slot 1 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 1 |
| 3-0<br>— | Reserved |

## 64.3.8.2.1.13 SLOT 1 DESCRIPTOR (MSB) (SLOT1_DESC1)

#### *64.3.8.2.1.13.1 Offset*

| Register | Offset |
|---|---|
| SLOT1_DESC1 | 2Ch |

#### *64.3.8.2.1.13.2 Diagram*




#### *64.3.8.2.1.13.3 Fields*

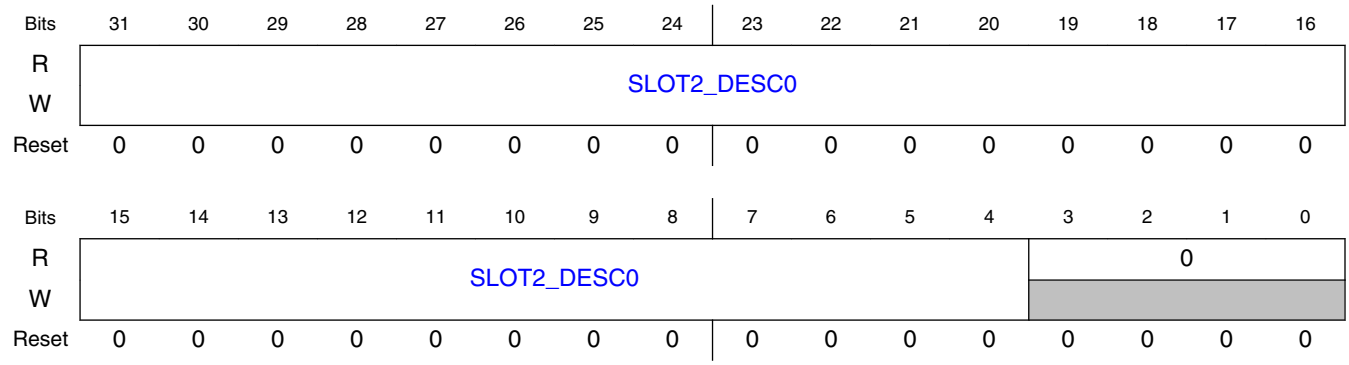
| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT1_DESC1 | Slot 1 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 1 |

### 64.3.8.2.1.14 SLOT 2 DESCRIPTOR (LSB) (SLOT2_DESC0)

#### *64.3.8.2.1.14.1 Offset*

| Register | Offset |
|---|---|
| SLOT2_DESC0 | 30h |

#### 64.3.8.2.1.14.2 Diagram




#### 64.3.8.2.1.14.3 Fields

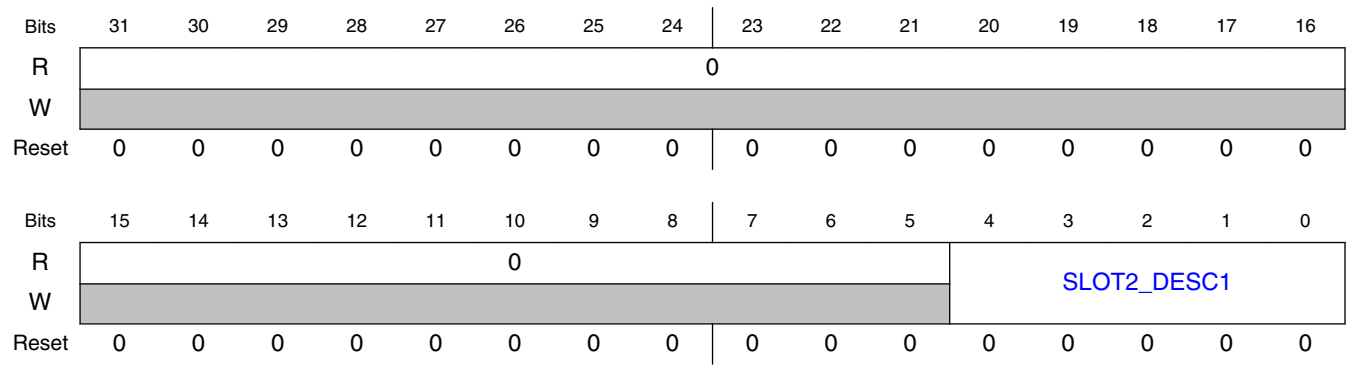
| Field | Function |
|---|---|
| 31-4<br>SLOT2_DESC0 | Slot 2 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 2 |
| 3-0<br>— | Reserved |

### 64.3.8.2.1.15 SLOT 2 DESCRIPTOR (MSB) (SLOT2_DESC1)

#### 64.3.8.2.1.15.1 Offset

| Register | Offset |
|---|---|
| SLOT2_DESC1 | 34h |

#### 64.3.8.2.1.15.2 Diagram



#### *64.3.8.2.1.15.3 Fields*

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT2_DESC1 | Slot 2 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 2 |

### 64.3.8.2.1.16 SLOT 3 DESCRIPTOR (LSB) (SLOT3_DESC0)

#### *64.3.8.2.1.16.1 Offset*

| Register | Offset |
|---|---|
| SLOT3_DESC0 | 38h |

#### *64.3.8.2.1.16.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLOT3_DESC0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLOT3_DESC0 | | | | | | | | | | | | 0 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.3.8.2.1.16.3 Fields*

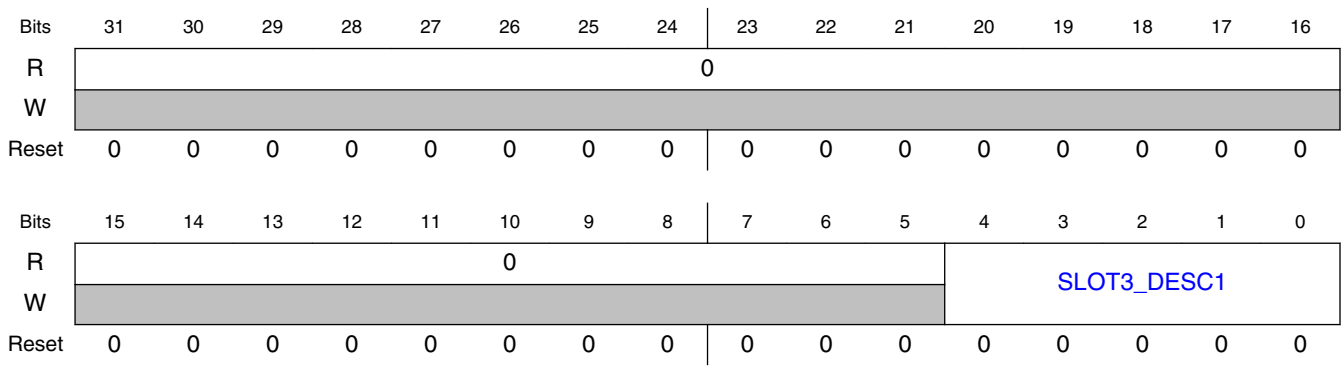
| Field | Function |
|---|---|
| 31-4<br>SLOT3_DESC0 | Slot 3 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 3 |
| 3-0<br>— | Reserved |

#### 64.3.8.2.1.17 SLOT 3 DESCRIPTOR (MSB) (SLOT3_DESC1)

##### *64.3.8.2.1.17.1 Offset*

| Register | Offset |
|---|---|
| SLOT3_DESC1 | 3Ch |

##### *64.3.8.2.1.17.2 Diagram*




##### *64.3.8.2.1.17.3 Fields*

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT3_DESC1 | Slot 3 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 3 |

## 64.3.8.3 Functional Description

### 64.3.8.3.1 Assign To Protocol

The 2.4GHz radio supports multiple protocols, each with its own protocol engine. All protocols can benefit from scheduling in advance several RF operations and DSM cycles, with the radio conducting the successive operations autonomously, allowing the MCU to remain in low power state for the duration. This is the concept behind Wake-On-Radio (WOR).

The WOR function can be assigned to any one of the following protocols: 802.15.4, GENERIC_FSK. (BLE has its own, internal DSM and wake-on-radio control scheme, as well as hardware-controlled frequency hopping, built into the IP, so it does not interact with the WOR Scheduler).

Assigning WOR to a protocol is done by way of the WOR_PROTOCOL register, as shown in the following table:

| WOR_PROTOCOL[2:0] | ASSIGNED PROTOCOL ENGINE |
|---|---|
| 0 | Not Assigned |
| 1 | 802.15.4 |
| 2 | GENERIC_FSK |
| 3 | Reserved |

Assigning to protocol means that the assigned Link Layer's WOR interface signals are routed, exclusively, to and from the WOR state machine. WOR interface signals include the event timers, sequence start- and stop-command signals, interrupt inhibits, and event timer addback commands. These interface signals are covered in more details in later sections.

The following diagram illustrates how 1 of the 3 WOR-capable Link Layers is assigned to the WOR function:

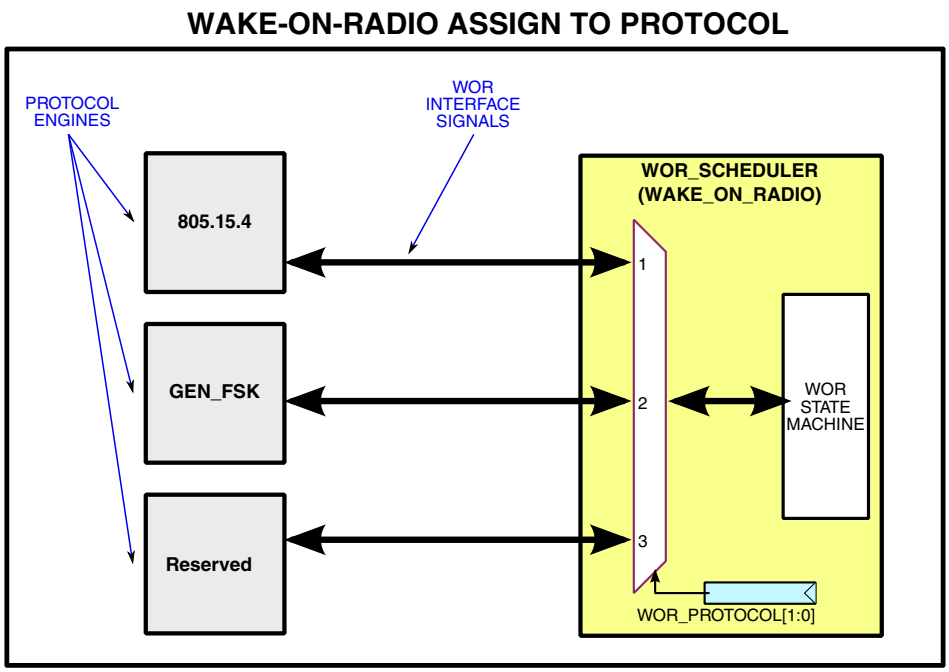



**Figure 64-91. WAKE-ON-RADIO ASSIGN TO PROTOCOL**

In a multi-protocol setting, assigning a protocol to Wake-On-Radio does not mean that unassigned protocols are incapable of DSM. There is a back-up method, Manual DSM, which uses software-driven DSM entry and exit. In a multi-protocol setting, any 2 of the radio's protocols can be assigned to different DSM control scheme, with the control schemes being Wake-On-Radio, Manual DSM, and BLE DSM.

### 64.3.8.3.2 Scheduling Modes

For maximum flexibility in scheduling future RF operations and Deep Sleep cycles, Wake-On-Radio (WOR) defines 2 scheduling modes: Timeslot Mode, and Constant Interval Mode. These modes are described in the sections below.

**Timeslot Mode**

Timeslot mode is desirable when the schedule for RF transactions is sporadic, varies over time, or is otherwise non-periodic. Timeslot mode is based loosely on the TSCH (Timeslotted Channel Hopping) concept, first defined in IEEE Std 802.15.4e-2012. Key principles of TSCH are adopted by WOR, but far short of full adherence.

In TSCH, time is divided into slots of fixed duration. Slots are grouped into slotframes, containing fixed number of slots. Once a TSCH PAN (network) is established, slotframes will repeat forever, back-to-back. Devices can be assigned to 1 or more slots within a slotframe. For any given device on any given slot, the assigned slot can be TX or RX. One device can be assigned to transmit in a given slot; another device can be assigned to receive in the same slot. Assigning Device A to Transmit, and Device B to Receive, in the same slot, creates a link. Links can be dedicated (one TX and one RX device assigned). Or, links can be shared (multiple TX devices, using CSMA, and multiple RX devices). Typically, each slot uses a different hop frequency.

Slotframes are TDMA structures, with time on one axis and hop frequency on the other. The diagram below shows one "row" of a typical slotframe, with links established for Devices A, B, and C, all on the same hop frequency. This slotframe consists of 3 timeslots: TS0, TS1, and TS2.

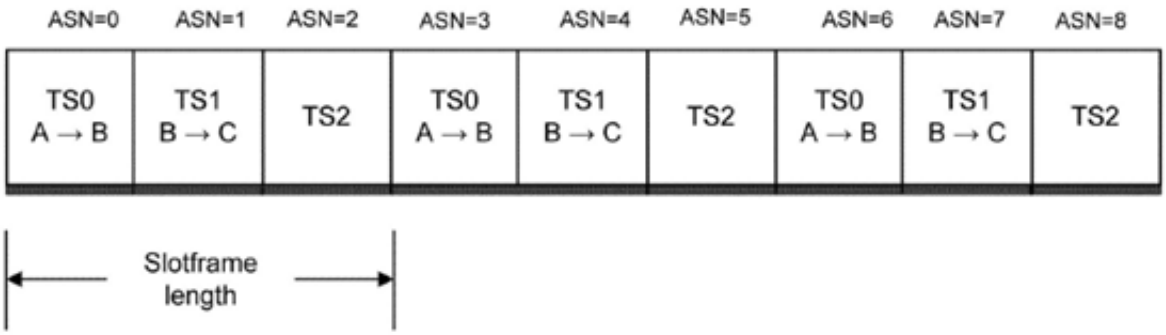



**Figure 64-92. 802.15.4 TSCH 1-D SLOTFRAME**

Another diagram below, shows a slotframe of 2 dimensions. The time axis has 100 slots, and the frequency axis has 4 indices into a table of available hop frequencies. Devices A, B, C, and D are assigned to various slots, and links are created amongst them.

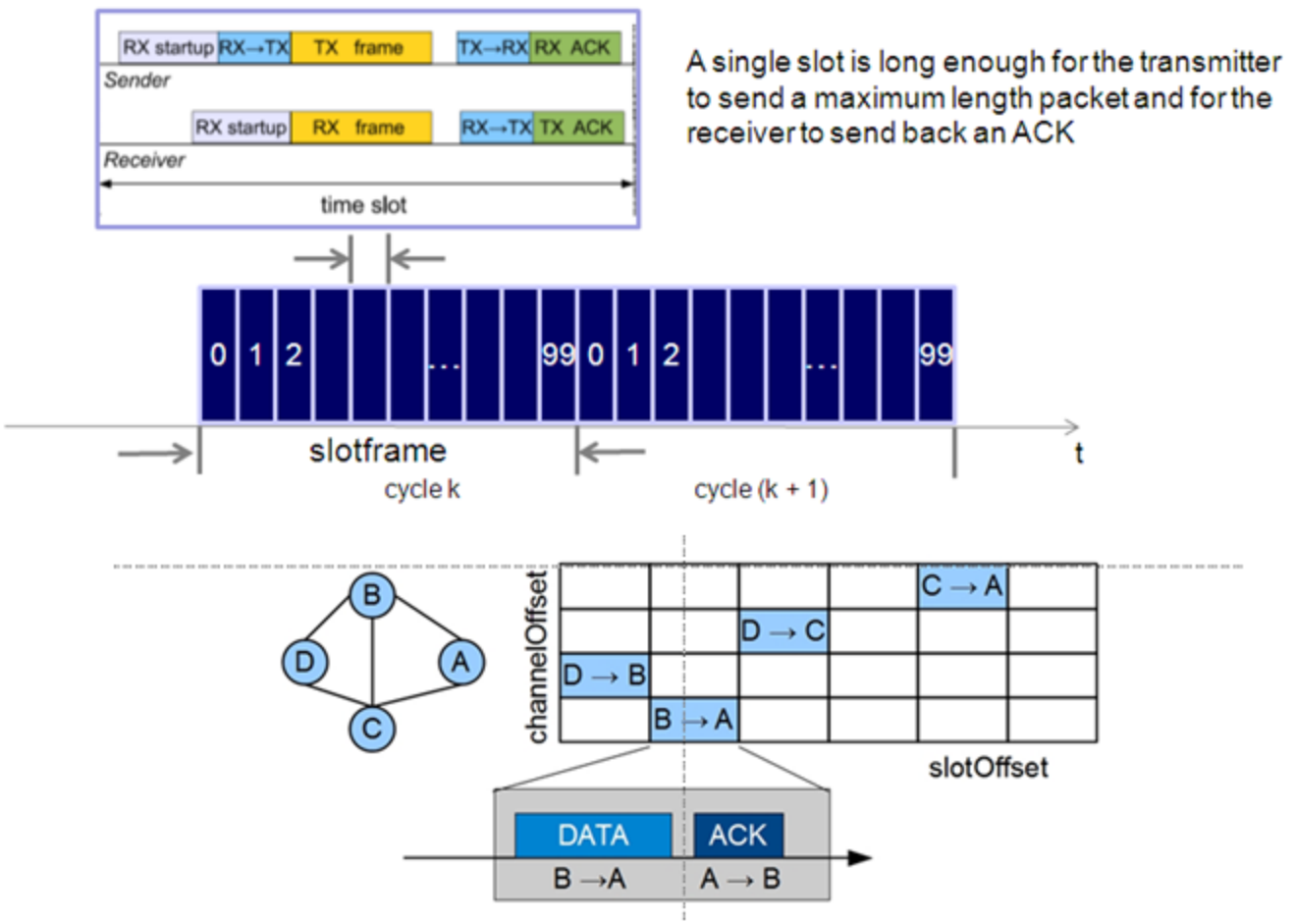



**Figure 64-93. 802.15.4 TSCH 2-D SLOTFRAME**

When RF operations are to be scheduled using Wake-On-Radio in a TIMESLOT context, the start times for the next *N* operations are programmed into the descriptor table to assign them to slots. One field of each slot descriptor, START_TIME, holds this information for WOR. See Section Descriptor Format for more information on the descriptor fields, and how they are packed into the descriptor. *N* is the number of slots used, and can be any integer from 2 to 4. The register SLOTS_USED is programmed with this value. The start times programmed into the descriptors, are relative to the Event Timer of the assigned Link Layer. WOR software will read the Event Timer, add to this value the desired time-to-next-event, and write the sum into the descriptor START_TIME. WOR hardware will monitor the assigned Link Layer's Event Timer; when it matches the START_TIME field of the current slot's descriptor, WOR hardware will initiate an RF sequence. The type of sequence is determined by the descriptor's FUNCTION field, and can be TX or RX. The RF operation will be allowed to continue, until it either self-terminates, or a timeout is reached. The timeout period for TIMESLOT scheduling mode is determined by the RECEIVE_TIMEOUT[15:0] register. The units of this register are microseconds. If a timeout is reached, i.e.,

EVENT_TMR = START_TIME + RECEIVE_TIMEOUT,

WOR hardware will terminate the sequence. Timeout applies to RX sequences only. RX sequences may result in no packet being received, a packet being received which is not addressed to the device, or a packet which is received corruptly. Any of these scenarios will result in a timeout and a WOR shutdown of the sequence. A correctly received packet addressed to the device will self-terminate the RX sequence before the timeout is reached. Timeout is not relavent to TX sequences, since these always self-terminate. The following diagram depicts how the WOR hardware processes the descriptor table, the assigned Event Timer, and the RECEIVE_TIMEOUT register, to start and terminate an RF operation.

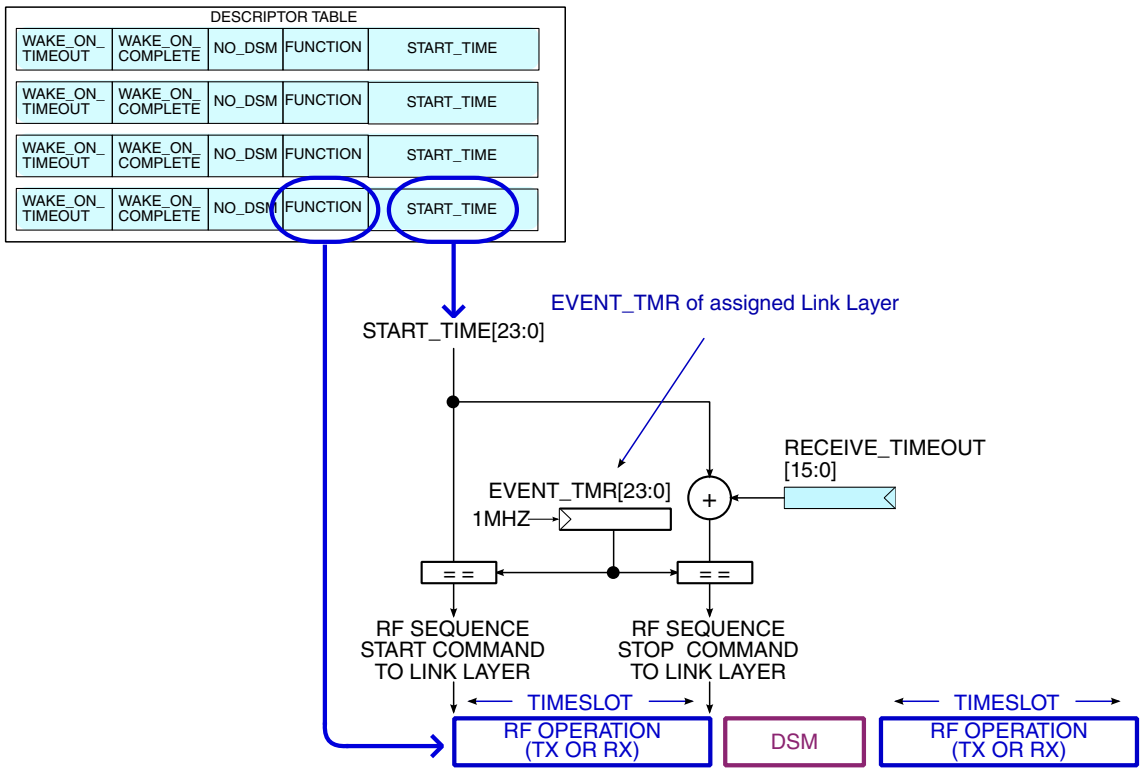



**Figure 64-94. EVENT SCHEDULING IN TIMESTLOT SCHEDULING MODE**

In any timeslot, the assigned FUNCTION is a simple TX or RX operation. There is no automatic acknowledgement of packets. If acknowledgement is required during a timeslot, the MCU must be awakened (by setting the WAKE_ON_COMPLETE bit of the slot's descriptor), and the software must handle the acknowledgement, using standard Link Layer (not WOR) resources, within the turnaround time requirement (if any). WOR hardware will hold its state and wait for software to complete the acknowledgement, and other required tasks, before advancing to the next slot on software command.

If the assigned protocol engine is 802.15.4, an autosequence is available which is a TX sequence preceded by a CCA (Clear Channel Assessment). If the CCA results in a busy channel, the autosequence terminates without transmission; otherwise the transmission proceeds. This autosequence can be used in WOR Timeslot mode by appropriately configuring the 802.15.4 Link Layer.

In any timeslot, if the assigned FUNCTION is RX, a TIMESTAMP can be captured to record the instant the packet's Access Address or SFD arrives at the Link Layer packet processor. There are 4 TIMESTAMP registers, once for each slot; hence, timestamps for the last 4 slots during which packets are received, can be recorded. See Section Timestamps for more details.

Typically, for RX timeslots, it is desirable to wake the MCU on good packets only. Thus, the slot descriptor's WAKE_ON_COMPLETE field should be set to 1. If it is desired to wake the MCU on no-packet, or on incorrecty-received packets, the WAKE_ON_TIMEOUT descriptor field can be set to 1. If WAKE_ON_TIMEOUT is set to 1 on any slot's descriptor, the WAKE_ON_COMPLETE field is *implied* to be set to 1 as well, regardless of its actual programmed value.

Typically, for TX timeslots, it is desirable to wake the MCU after each transmission, to service the Packet Buffer. In such scenarios, WAKE_ON_COMPLETE should be set to 1. If it is required to transmit the same packet repeatedly, across multiple slots, such that the Packet Buffer does not require regular service, WAKE_ON_COMPLETE can be set to 0, to skip the MCU wakeup, on any slot.

In Timeslot Scheduling mode, the MCU *must* be awakened at least once every *N* slots, to refresh the descriptor table (or at least, to refresh the START_TIME's), with *N* being the value of SLOTS_USED. This periodic wakeup requirement must be enforced by appropriate setting of the WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT fields of the 4 slot descriptors. This is a requirement imposed on WOR software. (The same period wakeup requirement does not necessarily apply to Constant Interval scheduling mode. See next section.)

## Constant-Interval Mode

Constant-Interval mode is appropriate when the schedule for RF transactions is perfectly periodic. A good example of such a scheduling regime, is the Bluetooth Low Energy Scanning State. In this state, a SCAN_INTERVAL is defined, which sets the interval between receive operations, and a SCAN_WINDOW is defined, which sets the duration of receiver "ON" time within the SCAN_INTERVAL. Wake-On-Radio adopts the principles of BLE SCAN_INTERVAL and SCAN_WINDOW, broadening their scope to apply to any type of RF operation (TX or RX), and modifying the nomenclature to better represent the nature of the operations, to WOR_EVENT_INTERVAL and WOR_EVENT_WINDOW.

Instead of programming timeslot START_TIME's into the slot descriptors, Contant-Interval requires that WOR_EVENT_INTERVAL and WOR_EVENT_WINDOW be programmed instead. An additional parameter, WOR_EVENT_START, determines the instant of the first RF operation of the Constant-Interval WOR session, in case the desired start time is not immediate. WOR_EVENT_START is also programmed into the descriptor table. In Constant-Interval scheduling mode, WOR hardware processes the descriptors, and uses WOR_EVENT_INTERVAL, WOR_EVENT_WINDOW, and WOR_EVENT_START, to calculate the start time (Event Timer match), for the first RF operation, as well as all subsequent RF operations, on a rolling basis. WOR software is not required to refresh the descriptor table on a periodic basis, and would only intervene

in the descriptors if a change was required to any parameter (e.g., a BLE "Anchor Point Move" during a connection would require descriptor updates). In Constant-Interval mode, the start of WOR_EVENT_WINDOW always aligns to the start of WOR_EVENT_INTERVAL.

Aside from replacing the Timeslot descriptor START_TIME parameters with WOR_EVENT_INTERVAL, WOR_EVENT_WINDOW, and WOR_EVENT_START, the descriptor format is identical across the 2 scheduling modes, and the descriptor storage space (registers) is completely shared, since only 1 scheduling mode can be in effect at any given time. See Section Descriptor Format for more information on the descriptor fields, and how they are packed into the descriptor, and how the descriptor table is shared between the 2 scheduling modes.

A diagram depicting an example of Constant-Interval scheduling mode is shown below. In this diagram, the scanning device is in Constant-Interval mode, with the WOR_EVENT_INTERVAL and WOR_EVENT_WINDOW indicated.

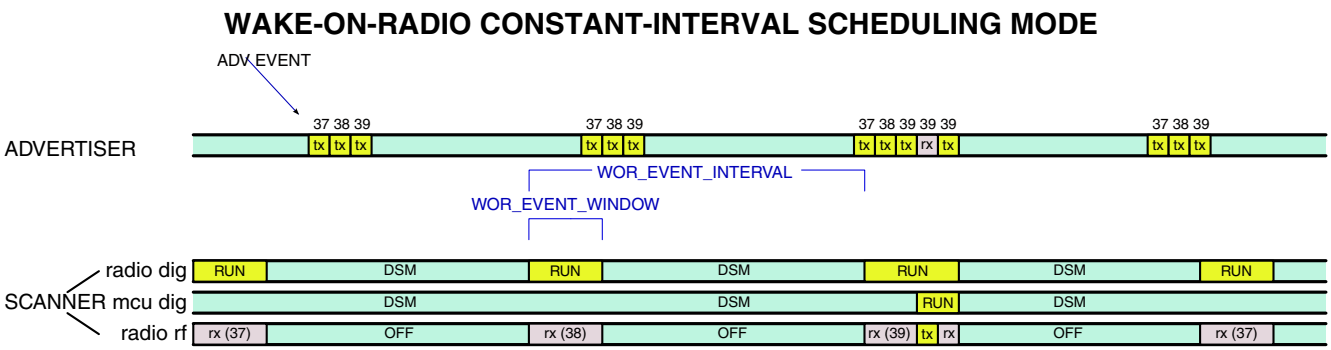



**Figure 64-95. CONSTANT-INTERVAL SCHEDULING MODE EXAMPLE**

When RF operations are to be scheduled using Wake-On-Radio in a Constant-Interval context, the WOR_EVENT_START, WOR_EVENT_INTERVAL, and WOR_EVENT_WINDOW parameters for the Constant-Interval operation are programmed into the descriptor table. These need be programmed only once, unless a change in network timing require an on-the-fly change to one of these parameters. The units for all 3 WOR_EVENT parameters are microseconds. See Section Descriptor Format for more information on the descriptor fields, and how they are packed into the descriptor. WOR hardware will process these 3 parameters to calculate the initial start time, and all subsequent start times, on a rolling basis. The start times computed by WOR hardware, are relative to the Event Timer of the assigned Link Layer. WOR hardware will monitor the assigned Link Layer's Event Timer; when it matches the next computed start time, WOR hardware will initiate an RF sequence. The type of sequence is determined by the current slot descriptor's FUNCTION field, and can be TX or RX. The

RF operation will be allowed to continue, until it either self-terminates, or a timeout is reached. The timeout period for Constant-Interval scheduling mode is determined by the WOR_EVENT_WINDOW descriptor field. If a timeout is reached, i.e.,

EVENT_TMR = START_TIME (computed) + WOR_EVENT_WINDOW (descriptor),

WOR hardware will terminate the sequence. Timeout applies to RX sequences only. RX sequences may result in no packet being received, a packet being received which is not addressed to the device, or a packet which is received corruptly. Any of these scenarios will result in a timeout and a WOR shutdown of the sequence. A correctly received packet addressed to the device will self-terminate the RX sequence before the timeout is reached. Timeout is not relavent to TX sequences, since these always self-terminate. The concept of slots still applies to Constant-Interval Mode. The number of slots used can be any integer from 3 to 4 for Constant-Interval mode. The register SLOTS_USED is programmed with this value. Consecutive RF operations are assigned to consecutive slots, with the slot descriptor's parameters (e.g. FUNCTION, or WAKE_ON_COMPLETE) applying to the current slot. As in Timeslot mode, in Constant-Interval mode, WOR hardware will increment the slot number for each new RF operation, modulo SLOTS_USED.

In any timeslot, the assigned FUNCTION is a simple TX or RX operation. There is no automatic acknowledgement of packets. If acknowledgement is required during a timeslot, the MCU must be awakened (by setting the WAKE_ON_COMPLETE bit of the slot's descriptor), and the software must handle the acknowledgement, using standard Link Layer (not WOR) resources, within the turnaround time requirement (if any). WOR hardware will hold its state and wait for software to complete the acknowledgement, and other required tasks, before advancing to the next slot on software command.

If the assigned protocol engine is 802.15.4, an autosequence is available which is a TX sequence preceded by a CCA (Clear Channel Assessment). If the CCA results in a busy channel, the autosequence terminates without transmission; otherwise the transmission proceeds. This autosequence can be used in WOR Timeslot mode by appropriately configuring the 802.15.4 Link Layer.

In any timeslot, if the assigned FUNCTION is RX, a TIMESTAMP can be captured to record the instant the packet's Access Address or SFD arrives at the Link Layer packet processor. There are 4 TIMESTAMP registers, once for each slot; hence, timestamps for the last 4 slots during which packets are received, can be recorded. See Section Timestamps for more details.

Typically, for RX timeslots, it is desirable to wake the MCU on good packets only. Thus, the slot descriptor's WAKE_ON_COMPLETE field should be set to 1. If it is desired to wake the MCU on no-packet, or on incorrecty-received packets, the

WAKE_ON_TIMEOUT descriptor field can be set to 1. If WAKE_ON_TIMEOUT is set to 1 on any slot's descriptor, the WAKE_ON_COMPLETE field is *implied* to be set to 1 as well, regardless of its actual programmed value.

Typically, for TX timeslots, it is desirable to wake the MCU after each transmission, to service the Packet Buffer. In such scenarios, WAKE_ON_COMPLETE should be set to 1. If it is required to transmit the same packet repeatedly, across multiple slots, such that the Packet Buffer does not require regular service, WAKE_ON_COMPLETE can be set to 0, to skip the MCU wakeup, on any slot.

Unlike Timeslot mode, in Constant-Interval mode, it is not required to wake up the MCU at least every *Nth* slot to refresh the descriptor start times, because WOR hardware is able to compute all future start times on a rolling basis. If all slots are of the same type (same FUNCTION, TX or RX, and same descriptor information), there is no hard limit on the number of consecutive slots that can be executed between MCU wakeups. To set such a upper limit, the WAKE_ON_NTH_SLOT[7:0] register can be programmed with a value. When the number of slots elapsed after setting SLOTS_USED to a non-zero value to begin a WOR session, reaches the value of WAKE_ON_NTH_SLOT, the MCU will automatically be awakened at the end of the current slot, regardless of the setting of the WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT bits in the descriptor. The WAKE_ON_NTH_SLOT register is only relavent to Constant-Interval scheduling mode, and has no effect in Timeslot mode. In Constant-Interval mode, if automatic wake on *Nth* slot is not desired, set WAKE_ON_NTH_SLOT=0 to disable this wakeup upper-limit. However, if a zero setting is used, software *must* program at least one of the slot descriptors to schedule an MCU wakeup by setting the either WAKE_ON_COMPLETE or WAKE_ON_TIMEOUT fields of the descriptor to 1; otherwise a situation could arise where the MCU is never awakened by WOR.

### 64.3.8.3.3 Descriptor Format

As mentioned in previous sections, Wake-On-Radio allows the next several RF operations to be scheduled in advance. For scheduling purposes, each of the next, several RF operations is assigned to a slot. Each slot is associated with a descriptor, which is a hardware-based data structure containing information about the characteristics of the slot, and instructions for how WOR hardware should execute it. Characterisics include start time of the RF operation, and type (function) or operation to be conducted by WOR. Software schedules the next *N* RF operations (slots) by programming the next *N* descriptors with the slots' characteristics. Software then sets the SLOTS_USED register to *N* to begin a WOR session. WOR hardware includes a 2-bit slot counter that cycles through the slots, in order, starting with SLOT 0; the slot counter is used to select the "current descriptor". The current descriptor's slot characteristics are forwarded to the WOR state machine, which executes the RF operation per the instructions in the

descriptor. After the RF operation is complete, the slot counter is incremented by 1 (modulo SLOTS_USED), and the next slot's descriptor becomes the "current" one. The process repeats indefinitely until the WOR session is ended by setting SLOTS_USED=0. WOR hardware automaticlly inserts (unless instructed otherwise in the descriptor) a DSM cycle, after the current slot's RF operation completes, with a DSM duration that lasts until the start time of the *next* slot, so that time between RF operations is always filled with DSM, for maximum power savings. After setting SLOTS_USED to a non-zero value to begin a WOR session, the MCU can be put into a low power state, and remain there, until the last (*Nth*) slot completes, assuming Timeslot scheduling mode. (For Constant Interval mode, MCU wake up on the *Nth* slot is not required. See Section Scheduling Modes for information on when and how the MCU is to be awakened for Constant-Interval mode.)

The descriptor format for Timeslot scheduling mode is shown in the diagram below, indicating the various fields in the descriptor, and the bits assigned to each field:

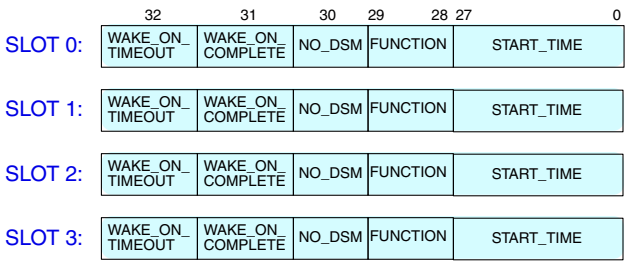



**Figure 64-96. DESCRIPTOR FORMAT (TIMESLOT MODE)**

The descriptor format for Constant-Interval scheduling mode is shown in the diagram below, indicating the various fields in the descriptor, and the bits assigned to each field:

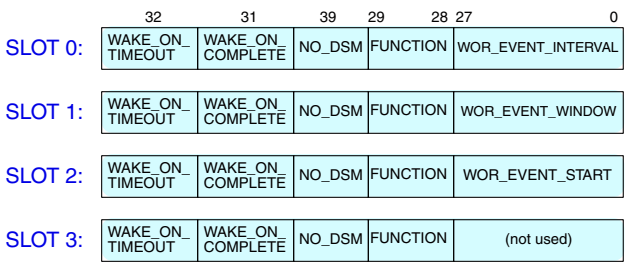



**Figure 64-97. DESCRIPTOR FORMAT (CONSTANT-INTERVAL MODE)**

The descriptor formats for Timeslot and Constant-Interval scheduling modes are almost identical, and so the desciptor hardware is almost completely shared by the 2 scheduling modes, as shown in the diagram below:

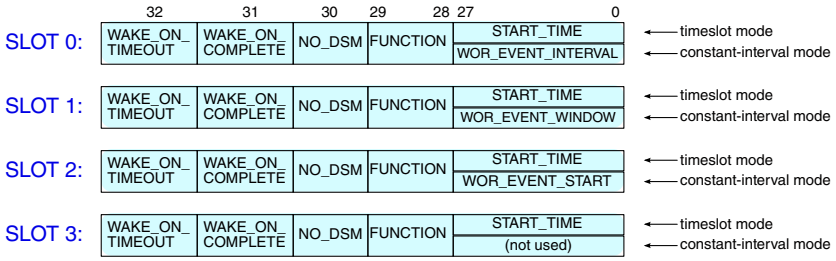



**Figure 64-98. DESCRIPTOR FORMAT (SHARED)**

Descriptor fields are described in the table below:

| FIELD | DESCRIPTION |
|---|---|
| START_TIME | The instant at which to begin the current slot's RF operation. START_TIME is relative to the assigned Link Layer's Event Timer. Units are microseconds. Because the Event Timers' bit widths vary with protocol, see START_TIME programming information later in this section. Applies to Timeslot mode only. WOR hardware will automatically adjust this value with window-widening, if enabled. |
| WOR_EVENT_INTERVAL | Sets the interval between the start of RF operations for consecutive slots. Units are microseconds. Applies to Constant-Interval mode only. WOR hardware will automatically adjust this value with window-widening, if enabled. |
| WOR_EVENT_WINDOW | Maximum duration of the transceiver "ON" time for the current slot. "ON" time will be less than this if the RF operation self-terminates. Units are microseconds. Applies to Constant-Interval mode only. The start of the WOR_EVENT_WINDOW aligns to the start of the WOR_EVENT_INTERVAL. WOR hardware will automatically adjust this value with window-widening, if enabled. |
| WOR_EVENT_START | The instant at which to begin the RF operation for the first SLOT 0 of the WOR session. WOR_EVENT_START is relative to the assigned Link Layer's Event Timer. Units are microseconds. Because the Event Timers' bit widths vary with protocol, see WOR_EVENT_START programming information later in this section. Applies to Constant-Interval mode only. WOR hardware will automatically adjust this value with window-widening, if enabled. |
| FUNCTION | This 2-bit field defines the RF function of the slot:<br>0: No RF<br>1: TX<br>2: RX<br>3: Reserved |
| NO_DSM | By default, a DSM cycle is automatically entered at the end of the current slot's RF operation, and the DSM cycle is timed to expire just before the start time of the next consecutive slot. If this field is set to 1, DSM is not entered, and the WOR state machine holds the radio in RUN state until the start time of the next slot is reached. NO_DSM can be used to execute back-to-back, or closely spaced RF operations, such as a BLE 3-channel advertising event. |

*Table continues on the next page...*

| FIELD | DESCRIPTION |
|---|---|
| WAKE_ON_COMPLETE | Instructs the WOR state machine to permit an interrupt to wake up the MCU at the end of a succesfully completed RF operation. RX operations are successfully completed if a packet is received before the timeout, which passes CRC and packet filtering rules. TX operations are always successfully completed. A successfully completed operation with WAKE_ON_COMPLETE=1 will cause the WOR state machine to temporarily remove its Link Layer interrupt inhibits, allowing the assigned Link Layer's RF interrupts (RXIRQ, TXIRQ, and/or SEQ_END_IRQ) to propagate, resulting in an MCU wakeup. If WAKE_ON_TIMEOUT=1, WAKE_ON_COMPLETE is *implied* to be 1 as well, regardless of the setting of this field. |
| WAKE_ON_TIMEOUT | Instructs the WOR state machine to permit an interrupt to wake up the MCU at the point of timeout of an RX operation. For Timeslot mode, timeout is determined by the RECEIVE_TIMEOUT[15:0] register; for Constant-Interval mode, timeout is determined by WOR_EVENT_WINDOW descriptor field. In either case, window-widening will automatically adjust the timeout instant, if enabled. A timeout which occurs with WAKE_ON_TIMEOUT=1 will cause the WOR state machine to temporarily remove its Link Layer interrupt inhibits, allowing the assigned Link Layer's SEQ_END_IRQ to propagate, resulting in an MCU wakeup, with a notification of timeout. |

Event Timer bit width various across protocols. For 802.15.4, Event Timer is 28 bits; for other protocols, Event Timer is 24 bits. In all cases, the Event Timer clock rate is 1MHz. For programming the non-802.15.4 START_TIME descriptor fields, 4 0's should prepend the 24-bit Event Timer match value. The same rule applies to programming the WOR_EVENT_START field.

When FUNCTION is set to "NO_RF", no RF operation is performed during the slot. In this scenario, the WOR state machine will wait until the slot's programmed (or computed) START_TIME, and then automatically re-enter DSM until the next consecutive slot's START_TIME is reached. This allows multiple (up to 4) DSM-only slots which effectively extends the maximum DSM duration fourfold, to 1073 seconds (about 18 minutes), for the 802.15.4 protocol, or 67 seconds (about 1 minute) for all other supported protocols. However the RF Oscillator is activated breifly at each slot's START_TIME so that the WOR can process the current slot descriptor, and update the assigned Link Layer's Event Timer. Extending the DSM duration using the "NO_RF" method is referred to as "DSM cascading".

During an RX operation on any slot, if an AA or SFD has been detected at the point of timeout, the timeout (and the automatic shudown of the sequence by the WOR state machine) is *deferred* until either:

1: The packet is fully received and determined to be good. In this case the WAKE_ON_COMPLETE bit determines if the MCU is to be awakened.
2: The packet is fully received and determined not to be good. In this case the WAKE_ON_TIMEOUT bit determines if the MCU is to be awakened.

There are no time restrictions on programming the descriptor table. All descriptor fields can be modified at any time. Needless to say, caution should be exercised with modifying descriptor START_TIME, and other parameters, for the *current* slot; START_TIME (or WOR_EVENT_START) must be in the future, and FUNCTION should not be modified once the START_TIME has been reached.

### 64.3.8.3.4 State Machine

In Wake-On-Radio, future RF operations are scheduled by programming instructions into the WOR Descriptor table. Based on descriptor specifications, WOR hardware organizes the scheduled RF operations into slots, with one RF operation per slot. The slot includes more than just the RF operation. Idle time between the current time, and the start of the RF operation, is filled with DSM. WOR hardware must compute the required DSM duration, and request DSM from RSIM (Radio System Integration Module). A diagram showing how WOR hardware processes the descriptor table, and organizes the required operations into consecutive slots, encompassing both DSM and RF, is shown below.

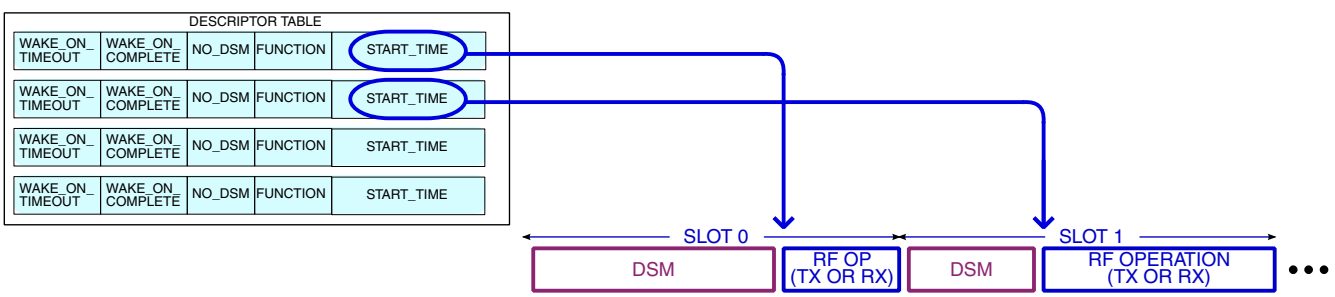



**Figure 64-99. TIMELINE OF CONSECUTIVE SLOTS**

Once a WOR session has been initiated by setting SLOTS_USED to a non-zero value, the timeline of consecutive slots shown, will repeat indefinitely, with the slot number increasing by 1 after each slot, modulo SLOTS_USED.

During each slot, WOR hardware must perform a series of tasks, e.g., computing the DSM duration, commanding the the assigned Link Layer to start and stop the RF sequence, etc. Some of the required WOR hardware tasks are superimposed on the slot timeline, in the diagram below.

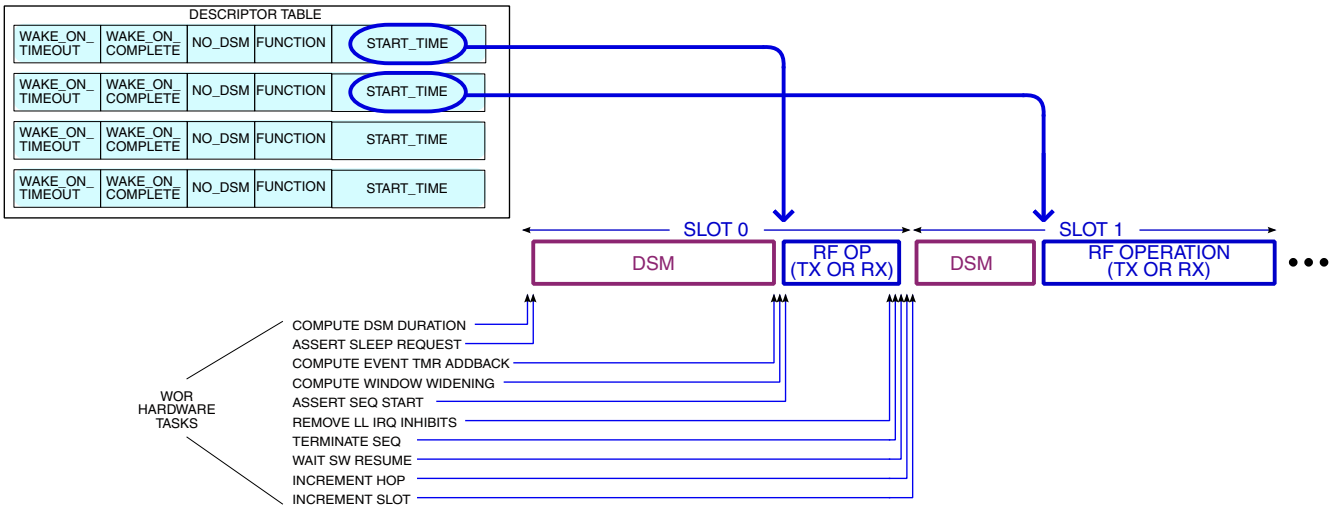



**Figure 64-100. SLOT TIMELINE WITH HARDWARE TASKS**

The WOR state machine is responsible for processing the descriptor table, and performing all the required tasks during each slot. The 10-state machine interfaces with the assigned Link Layer for sequence operations, and with RSIM for DSM operations. The state machine also controls the incrementing of the the slot counter, and the frequency hop table index. A state diagram of the WOR state machine is shown below.

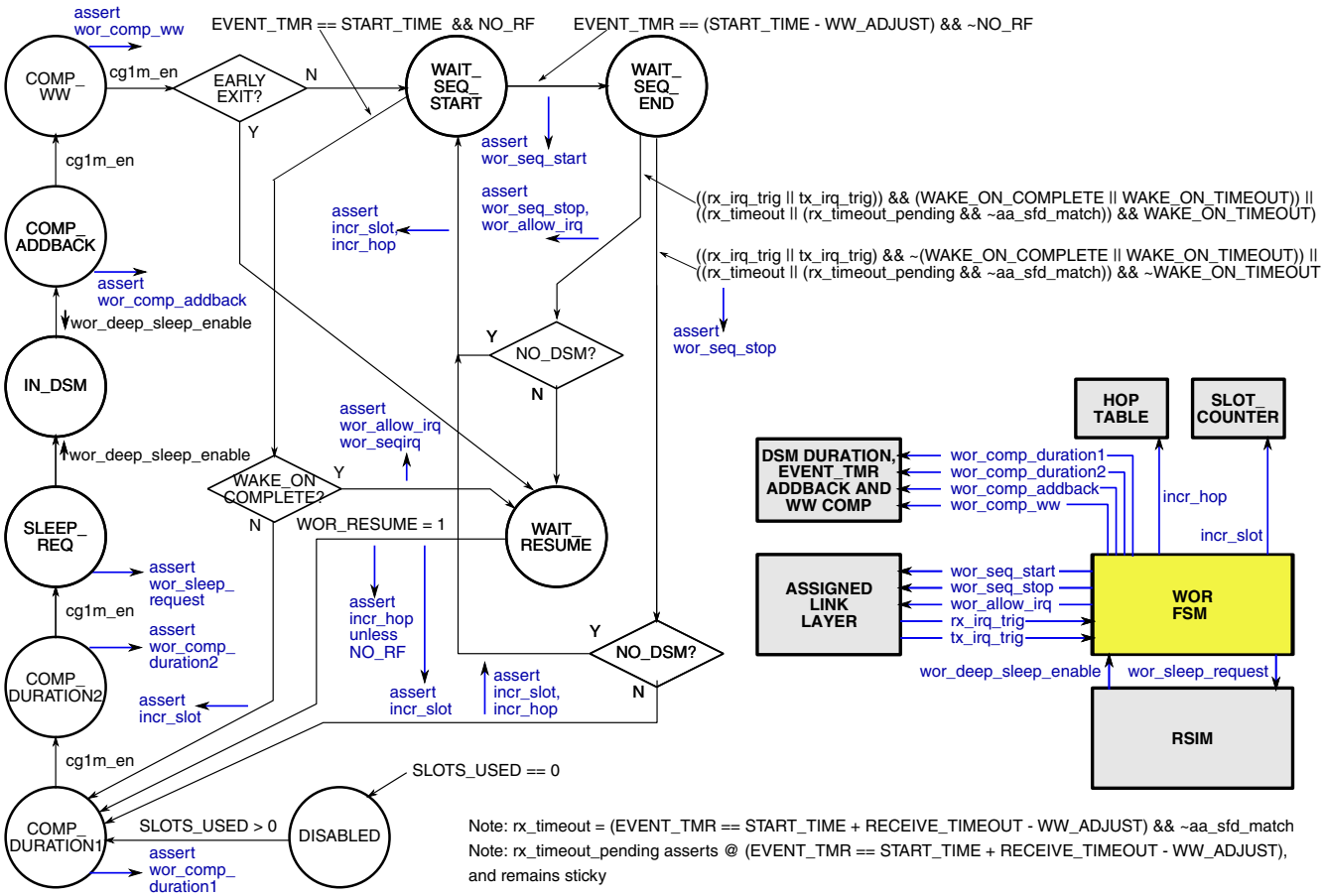



**Figure 64-101. WAKE-ON-RADIO STATE DIAGRAM**

A description of the WOR states follows.

## DISABLED

WOR is in its disabled state whenever SLOTS_USED=0. WOR can be taken to "DISABLED" from any state by setting SLOTS_USED=0. WOR can be enabled by setting SLOTS_USED to a non-zero value, which initiates a new WOR "session".

## COMP_DURATION1

WOR must compute the duration of the upcoming DSM cycle. This is done in 2 stages (states). COMP_DURATION1 is the first stage. Whenever the slot counter is incremented to begin a new slot, the slot's (descriptor's) START_TIME is in the future. In most cases, the START_TIME is distant enough that DSM is justified. During COMP_DURATION1, WOR computes the time to the next START_TIME, by subtracting both the current time (the assigned Link Layer's Event Timer) and the time-accumulated window-widening (if enabled) from the slot descriptor's START_TIME. (See Section Window Widening for more details on this feature.) The result of this subtraction is MICROSECONDS_TO_START_TIME[27:0]. This value must be converted to 32.768KHz cycles to be of use to RSIM. The multiplier is 0.032714844

(536 / 16384), which is an approximation to the ideal conversion factor, with a precision of 0.16%. The product is truncated to integer, the result represents the number of 32.768KHz cycles to remain in DSM prior to the next START_TIME. The hardware computations which take place during COMP_DURATION1 are shown in the following diagram.

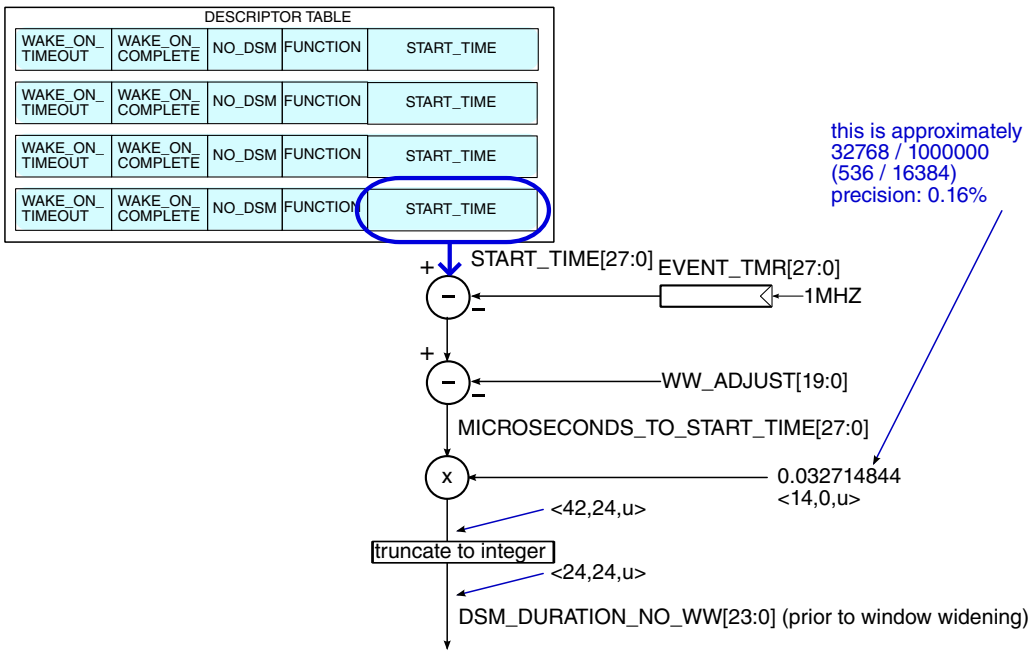



**Figure 64-102. COMP_DURATION1**

However this intermediate result does not account for the fact that window-widening for the upcoming DSM cycle itself, must be subtracted from this product. This is separate from the "accumulated" window-widening which has built up during the WOR session so far, which has already been accounted for. The estimate for the next DSM cycle's window widening is handled in the next state, COMP_DURATION2

The WOR output wor_clr_dsm_duration is asserted in this state, which is sent to the RSIM to proactively clear the DSM Duration Counter, which becomes active in the IN_DSM state (see below). The wor_clr_dsm_duration is an active-high, registered (glitch-proof) output signal.

## COMP_DURATION2

If window-widening is enabled, the DSM duration computed during COMP_DURATION1 may be long enough that it will need to be shortened slightly to account for window-widening which will accumulate during the DSM cycle itself. The computation for this window-widening is performed during this state. (See Section Window Widening for more details on this computation, and the usage of WW_DSM_FACTOR.) The (intermediate) duration from COMP_DURATION1 is

multiplied by the DSM window-widening factor, and the product is subtracted from the duration estimate. The difference is a more accurate duration that accounts for window-widening during the DSM cycle. In a final step, a guardband is subtracted from the DSM duration. The guardband, with units of 32.768KHz clock cycles, guarantees that the DSM will complete on time before the next START_TIME is reached, accounting for RSIM synchronization and other delays in processing the WOR's sleep request. The guardband, which applies to all DSM cycles initiated by WOR, is determined by the register DSM_GUARDBAND[3:0]. The result of this step, is the final DSM duration (wor_dsm_duration[23:0]) which is then sent to the RSIM, which is responsible for radio DSM entry and exit. The hardware computations which take place during COMP_DURATION2 (alongside those covered by COMP_DURATION1) are shown in the following diagram.

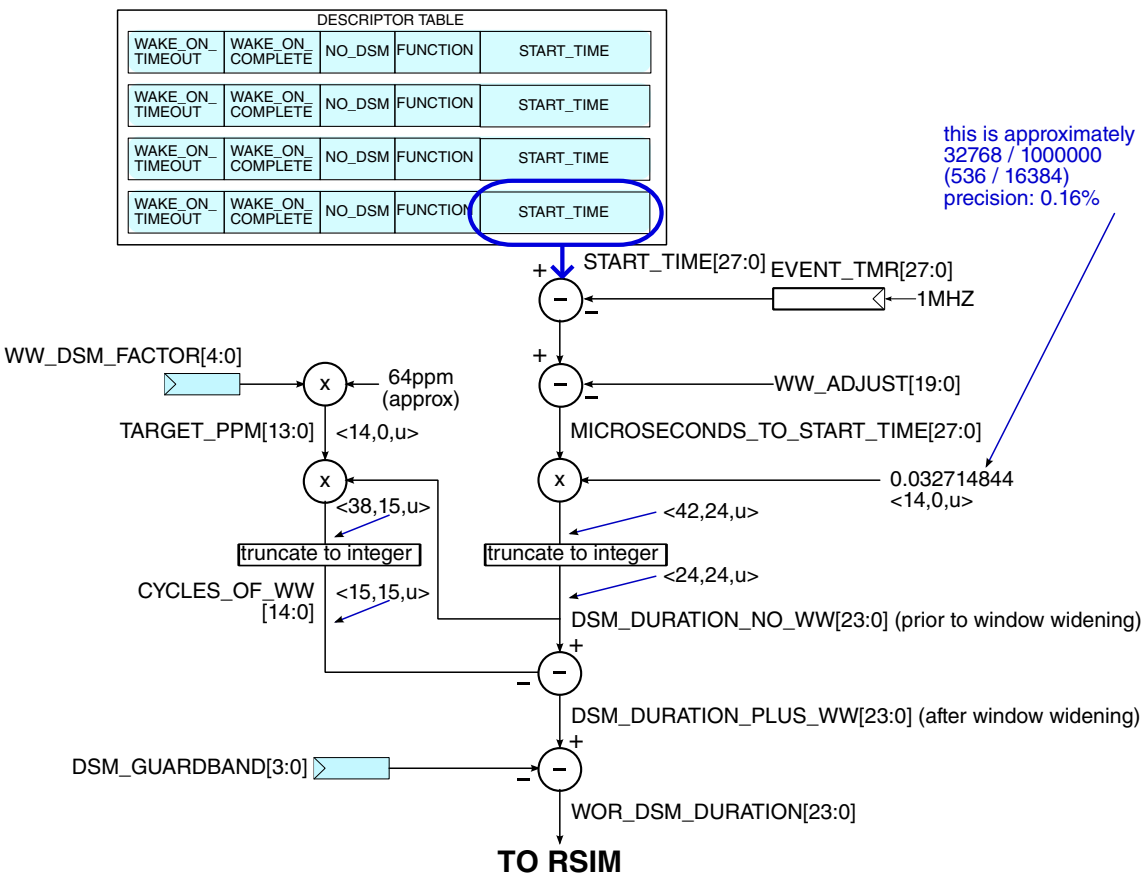



**Figure 64-103. COMP_DURATION2**

## SLEEP_REQ

In this state, WOR asserts its sleep request (wor_sleep_request) to the RSIM. This is a level, active-high, and registered (glitch-proof) signal to RSIM to indicate a DSM cycle is being requested of precise duration wor_dsm_duration[23:0], which was computed during the prior 2 WOR states. The units of wor_dsm_duration[23:0] are 32.768KHz

clock cycles. RSIM shall synchronize wor_sleep_request into the 32.768KHz clock domain, and use it to activate the RSIM's DSM state machine. The RSIM DSM state machine shall place the radio in DSM, and simultaneously assert the output: wor_deep_sleep_enable. This signal is the precise indicator of DSM being "in effect", and is used to gate the assigned Link Layer's clocks and freeze its Event Timer.

## IN_DSM

DSM is in effect during this state. Because the RF Oscillator has been disabled, there are no clocks to WOR, including its state machine, or the Link Layer, in this state. This is merely a holding state that prevails until DSM exits and the clocks to WOR and Link Layer are restarted. During this state, RSIM will use wor_deep_sleep_enable to enable a 24-bit counter which counts out the precise number of cycles in wor_dsm_duration[23:0]. When this counter reaches wor_dsm_duration, RSIM shall exit DSM and de-assert wor_deep_sleep_enable. RSIM shall provide the final count of the 24-bit counter which was used to time the DSM, using output signal rsim_dsm_duration[23:0]. In most cases, rsim_dsm_duration will be an exact match to wor_dsm_duration; the exceptional case is an unscheduled, early exit of DSM (see Section DSM Early Exit.) The interface signals between WOR and RSIM, and a high-level description of the hardware tasks performed by RSIM during IN_DSM state, is shown in the following diagram.

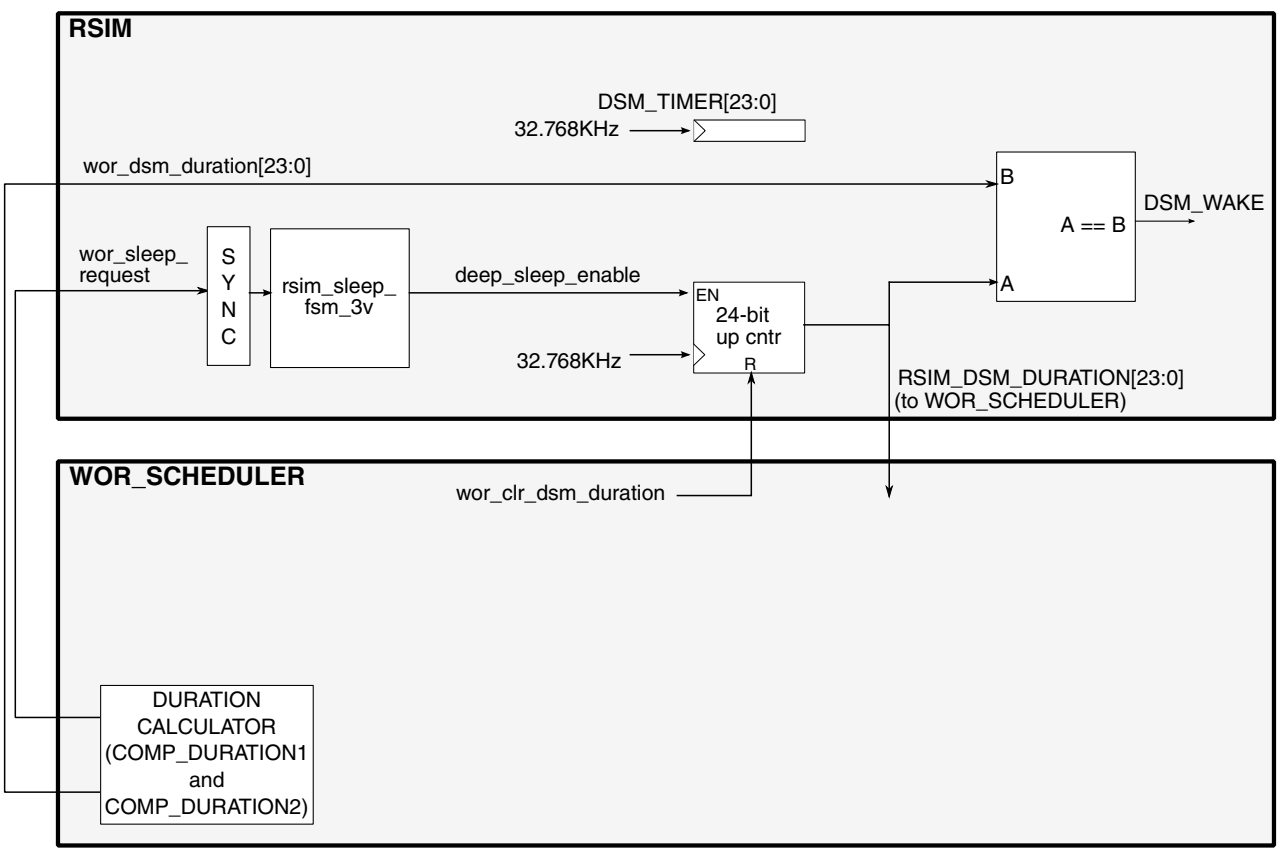



**Figure 64-104. IN_DSM**

## COMP_ADDBACK

During the previous state, IN_DSM, the Reference Oscillator was disabled, and so the Event Timer of the assigned Link Layer was frozen. After DSM completes, clocks are restored and Event Timer resumes counting, but it is "behind" by the amount of time that the radio was in DSM. WOR hardware must compute the amount of DSM time, convert it to microseconds, and "add it back" to the Event Timer, so that after the addback, the Event Timer timebase is restored to where it would have been, had there been no DSM. (Note: there is a maximum of 1us of "loss of precision" upon every DSM exit; this timebase error is random and will not accumulate, it will average out over many DSM cycles). In this state, WOR computes the addback value by multiplying the rsim_dsm_duration[23:0] (the precise number of 32.768KHz clock cycles during which RSIM held the radio in DSM) by 30.517578125 (the number of microseconds per 32.768KHz clock cycle). The 30.517578125 is an unsigned 14-bit value with 5 integer bits. The other multipicand, rsim_dsm_duration is an unsigned integer. The product is the DSM elapsed time in microseconds, a 38 unsigned bit value with 29 integer bits. The product is rounded to the nearest integer and the 5 MSB's are discarded, resulting in the final wor_addback[23:0] value. At the end of this state, this result is sent to the assigned Link Layer's Event Timer, along with the wor_addback command (pulse), indicating to the Link Layer that it must add this amount to the current state of its Event Timer. After this point the protocol's timebase has been restored and operations can now be performed referencing the Event Timer. The following diagram depicts the hardware tasks performed in the COMP_ADDBACK state.

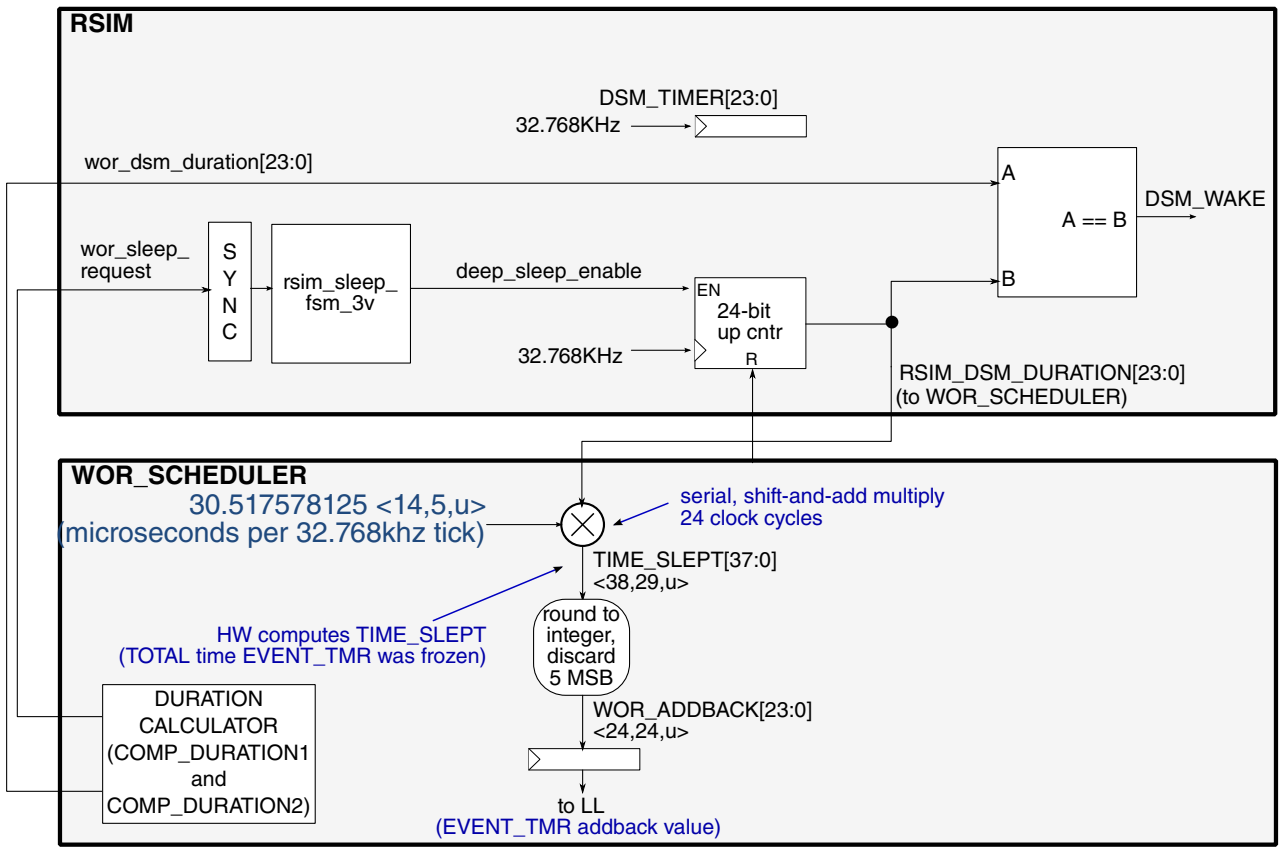



**Figure 64-105. COMP_ADDBACK**

## COMP_WW

If window-widening is enabled, WOR must compute how much to increase the receive window by, based on the duration of the just-completed DSM cycle, so that this early-start offset to the receive window is ready in advance of the slot descriptor's START_TIME. The window-widening increment is computed during this state. Details of this hardware task are described in Section Window Widening.

## WAIT_SEQ_START

After COMP_WW state, the WOR is ready to command an RF operation, and is only waiting for the slot descriptor's START_TIME to arrive. This start time will be in the near future since the DSM duration was timed to wake up "just in time" to start the RF operation, with sufficient guardband to guarantee that a DSM wake up can never occur too late. The WAIT_SEQ_START state merely waits for the assigned Event Timer to match the current slot descriptor's START_TIME. If the slot's FUNCTION is RX, and window widening is enabled, START_TIME is modified by the hardware to be:

START_TIME - ww_adjust[19:0],

where ww_adjust[19:0] is the time-accumulated window-widening "early start" amount for the RX operation (see Section Window Widening.) If the FUNCTION is not RX or if window-widening is disabled, START_TIME is unmodified. Once a START_TIME match to the Event Timer occurs, WOR asserts the Link Layer interface signal wor_tx_start_ext (if FUNCTION is TX), or wor_rx_start_ext (if FUNCTION is RX). Neither interface signal is asserted if FUNCTION is NO_RF. The assigned Link Layer will commence an RF operation, TX or RX, based on this WOR command. At this point, the WOR state machine arrives at a fork:

- If FUNCTION = NO_RF and the descriptor WAKE_ON_COMPLETE=1, the next state is WAIT_RESUME
- Else if FUNCTION = NO_RF and WAKE_ON_COMPLETE=0, the next state returns to COMP_DURATION1 to begin the next slot. The slot counter and hop table index are incremented.
- Else, this is an RF operation that must be waited for. The state machine advances to the WAIT_SEQ_END state.

### WAIT_SEQ_END

This state has been reached because an RF operation was launched during the previous state, and WOR is waiting for that operation to end. There are 3 ways for an RF operation to end:

- A TX operation self-terminates, indicated when the Link Layer asserts tx_irq_trig to WOR.
- A RX operation self-terminates due to a good packet reception, indicated when the Link Layer asserts rx_irq_trig to WOR.
- A RX operation times out. The rx_irq_trig is not asserted before the timeout. See Section Scheduling Modes for a description on the 2 methods WOR uses to compute the point-of-timeout, based on Scheduling Mode.

If either tx_irq_trig or rx_irq_trig is asserted, WOR advances to the next state (see below). If a point-of-timeout is reached without an rx_irq_trig, WOR hardware checks to see if Access Address Match has occurred (for protocols other than 802.15.4), or if SFD has been detected (for 802.15.4). If either is the case, timeout is deferred until either:

- The rx_irq_trig is ultimately asserted to indicate a good packet was received. This is a delayed good packet reception.
- The packet reception underway at point-of-timeout ultimately results in a CRC- or filtering-violation. This will automatically trigger a Link Layer RX recycle which will negate the AA-match or SFD-match indicator. WOR will detect this negation and declare a delayed timeout at this point.

If a timeout occurs, either nominal or delayed, WOR will assert wor_seq_stop_ext to the assigned Link Layer, causing an immediate abort of the RX sequence. This abort shall be handled by the protocol engine like a software abort. The wor_seq_stop_ext will not assert for a deferred timeout which results in a delayed good packet reception.

Nominally, WOR blocks the assigned Link Layer's RF-related interrupts from propagating to the MCU, to prevent unintended MCU wakeups during a WOR session. The RF-related interrupts for the supported protocol engines, are nearly identical in function across a 3 protocols, and are called RXIRQ, TXIRQ, and SEQ_END_IRQ. At the conclusion of an RF operation on a slot, WOR must decide whether to temporarily remove the Link Layer interrupt inhibits, to allow the Link Layer interrupt to be asserted to wake the MCU, based on which of the 5 outcomes to the RF operation occurred, and the state of the WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT descriptor fields. The following table specifies whether WOR blocks, or allows, the RF interrupts to propagate, at the conclusion of the RF operation.

| RF OPERATION OUTCOME | WAKE_ON_TIMEOUT=0, WAKE_ON_COMPLETE=0 | WAKE_ON_TIMEOUT=0, WAKE_ON_COMPLETE=1 | WAKE_ON_TIMEOUT=1, WAKE_ON_COMPLETE=x |
|---|---|---|---|
| tx_irq_trig | BLOCK | ALLOW | ALLOW |
| rx_irq_trig (nominal) | BLOCK | ALLOW | ALLOW |
| rx_irq_trig (delayed) | BLOCK | ALLOW | ALLOW |
| no rx_irq_trig @ nominal timeout | BLOCK | BLOCK | ALLOW |
| no rx_irq_trig @ delayed timeout | BLOCK | BLOCK | ALLOW |

At the conclusion of the RF operation, the WOR next state is a function of whether the Link Layer RF interrupts were BLOCKED or ALLOWED (see previous table), as well as the NO_DSM descriptor field as illustrated in the following table.

| RF OPERATION OUTCOME | INTERRUPTS BLOCKED, NO_DSM=0 | INTERRUPTS ALLOWED, NO_DSM=0 | NO_DSM=1 |
|---|---|---|---|
| tx_irq_trig | COMP_DURATION1 (increment slot, hop idx) | WAIT_RESUME | WAIT_SEQ_START (increment slot, hop idx) |
| rx_irq_trig (nominal) | COMP_DURATION1 (increment slot, hop idx) | WAIT_RESUME | WAIT_SEQ_START (increment slot, hop idx) |
| rx_irq_trig (delayed) | COMP_DURATION1 (increment slot, hop idx) | WAIT_RESUME | WAIT_SEQ_START (increment slot, hop idx) |
| no rx_irq_trig @ nominal timeout | COMP_DURATION1 (increment slot, hop idx) | WAIT_RESUME | WAIT_SEQ_START (increment slot, hop idx) |
| no rx_irq_trig @ delayed timeout | COMP_DURATION1 (increment slot, hop idx) | WAIT_RESUME | WAIT_SEQ_START (increment slot, hop idx) |

## WAIT_RESUME

If WOR has allowed the MCU to be awakened (interrupted) by temporarily negating the RF interrupt inhibits, WOR must enter a state which permits the MCU executing WOR or Link Layer software, to complete all its assigned tasks on the slot. These tasks may include processing and decrypting a received packet, servicing the packet buffer, and formulating and transmitting an acknowledge packet. The WAIT_RESUME state serves that purpose. The WAIT_RESUME state is reached whenever all of the following conditions are met:

- Current state is WAIT_SEQ_END
- FUNCTION is TX or RX
- RF operation completes with any outcome (rx_irq_trig, tx_irq_trig, nominal, or delayed timeout)
- NO_DSM=0
- RF Interrupts are ALLOWED (see table, WAIT_SEQ_END state)

WAIT_RESUME is also reached whenever all of these following conditions are met:

- Current state is WAIT_SEQ_START
- FUNCTION is NO_RF
- START_TIME match to the assigned Link Layer's Event Timer
- WAKE_ON_COMPLETE=1

Once WAIT_RESUME is reached, WOR software must authorize WOR hardware to resume the WOR session and advance to the next slot, by setting the WOR_RESUME register bit to 1. When WOR_RESUME is set to 1 in WAIT_RESUME state, WOR hardware takes the following steps:

1. Advances to the next slot (slot counter++ % SLOTS_USED)
2. Increments the frequency hop table index *unless* FUNCTION = NO_RF
3. Enters COMP_DURATION1 state on the new slot
4. Reimposes Link Layer interrupt inhibits

### 64.3.8.3.5 Window Widening

**OVERVIEW**

Because a device in a connected state is not receiving all the time, its timebase will invariably drift with respect to the timebase of other device(s) in the network, due to finite crystal oscillator tolerances amongst the networked devices. Reducing the duty cycle of the receiver, albeit necessary to achieve power savings, exacerbates the drift problem.

To compensate for drift, Receive Window Widening can be employed. As time elapses in a network connection between successful packet receptions, the window-widening algorithm gradually moves future receiver turn-on times to the left (advance) and the turn-off time to the right (retard) to allow the transmitted packet to fall within the

receiver's window in the presence of a drifting timebase. The effect of window-widening on successive receive cycles, none of which result in a good packet reception, is illustrated in the following diagram.

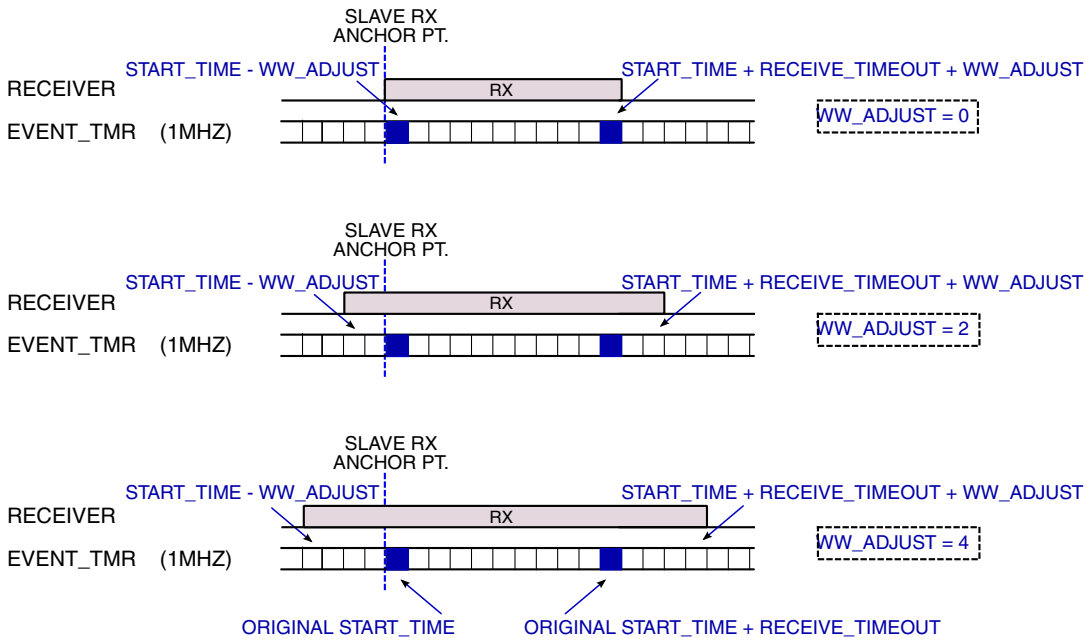



**Figure 64-106. EFFECT OF WINDOW WIDENING ON RECEIVE WINDOW**

Window widening is typically applicable only to the slave device(s) in a network, as the master (coordinator) of the network is normally the master of network timing. Thus, in Wake-On-Radio, window widening is optional, enabled by the by setting the WW_EN register bit to 1.

The Window widening algorithm needs to take into account the "Sleep Clock Accuracy" of the devices in the network. Sleep Clock Accuracy (SCA) is a *Bluetooth Low Energy* concept, and the BLE nomenclature is adopted by Wake-On-Radio, and extrapolated to all supported protocols.

In BLE, drift encountered by a slave device must be compensated by accounting for the SCA of both the master and the slave (self). The master transmits its SCA to the slave in the connection request (CONNECT_REQ), an advertising channel packet which informs an advertising slave of the master's desire to form a connection with it; the parameters on the CONNECT_REQ include the master's SCA. The slave device knows its own SCA, so the composite SCA is known at the point of connection establishment. Wake-On-Radio adopts this concept, although the precise method of informing the slave of Master SCA is protocol-dependent and takes place at a higher level, beyond the scope of this document.

BLE subdivides the allowed Master SCA into coarse ranges, to reduce the number of over-the-air bits required to inform the slave. The ranges are shown in the following diagram.

| SCA | *masterSCA* |
|---|---|
| 0 | 251 ppm to 500 ppm |
| 1 | 151 ppm to 250 ppm |
| 2 | 101 ppm to 150 ppm |
| 3 | 76 ppm to 100 ppm |
| 4 | 51 ppm to 75 ppm |
| 5 | 31 ppm to 50 ppm |
| 6 | 21 ppm to 30 ppm |
| 7 | 0 ppm to 20 ppm |

*Table 2.2: SCA field encoding*

**Figure 64-107. SCA RANGES**

The slave SCA must be accounted for as well, so the maximum composite SCA is 1000ppm. For Wake-On-Radio, this is the maximum allowed drift as well, and window-widening programmability will allow for drift amounts within these ranges to be compensated.

## FACTS

In the WOR hardware, the window-widening adjustment, is a value, ww_adjust[19:0], that is subtracted from the receiver START_TIME, and added to the receiver TIMEOUT, when the slot's FUNCTION = RX, as shown in the diagram below.

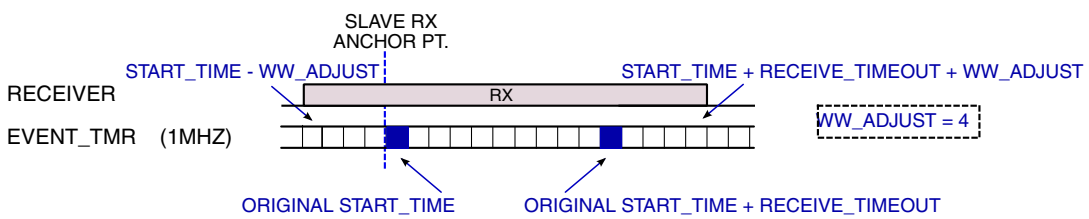



**Figure 64-108. WINDOW WIDENING EFFECT ON RX SEQUENCE**

The ww_adjust[19:0] adjustment, has no effect on START_TIME for slots using FUNCTION TX or NO_RF.

After system reset, if window-widening is enabled using WW_EN, ww_adjust[19:0] will increase linearly over time.

The adjustment, ww_adjust[19:0], will be reset to 0 under any of the following conditions:

1. system reset
2. good packet received (CRC verified and packet filtering passes)
3. Software writes a 1 to WW_NULL register bit

The reset-on-good-packet (item #2 above), can be inhibited by setting register bit WW_RESET_ON_RX=0. By default this bit is set.

The adjustment, ww_adjust[19:0], consists of 2 components, which are summed together:

1. **DSM adjustment:** computed at the conclusion of every DSM cycle based on rsim_dsm_duration[23:0]
2. **RUNTIME adjustment:** computed by periodically incrementing a binary counter by 1 every *N* EVENT_TMR ticks whenever NOT in DSM

The rates at which window-widening accumulates during DSM and RUNTIME, are controlled by 2 programmable registers. The programmed values of these registers should be informed by the combined (master + slave) SCA.

The 2 components of the window-widening adjustment, and the reset conditionality, are illustrated in the following diagram:

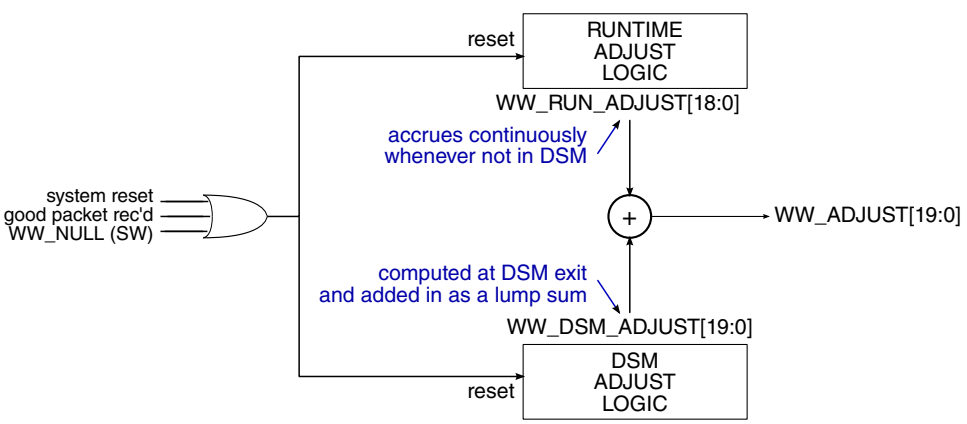



**Figure 64-109. WINDOW WIDENING COMPONENTS AND RESET CONDITIONALITY**

## WINDOW WIDENING DSM ADJUSTMENT

In Wake-On-Radio, every slot includes a DSM cycle (unless NO_DSM=1 in the descriptor). WOR hardware computes the window-widening DSM adjustment after the DSM cycle completes, during the WOR COMP_WW state. This state immediately follows the COMP_ADDBACK state which updates the Event Timer for time-slept. The WW DSM adjustment, ww_dsm_adjust[19:0], is computed, and then added to the composite, accumulated adjustment, ww_adjust[19:0], as a lump sum. During COMP_WW state, WOR hardware multiplies rsim_dsm_duration[23:0], which is the precise number of 32.768KHz cycles for which the RSIM held the radio in DSM, by a programmable factor which represents the number of microseconds per 32.768KHz clock cycle, scaled by the combined SCA of the master and slave (drift), i.e.,

**DSM_DURATION (from RSIM) * (masterSCA + slaveSCA), converted to µs**

Programmable register WW_DSM_FACTOR[4:0] maps the composite SCA (masterSCA + slaveSCA), in parts per million, to a multiplicand which represents

***N* * 64ppm * 30.5178125µs,**

where *N* = WW_DSM_FACTOR, and

30.5178125 is the *precise* number of microseconds per 32.76KHz tick, not an approximation.

Register WW_DSM_ADJUST allows SCA to be applied in coarse ranges, as described earlier in this section. Each LSB of WW_DSM_ADJUST is worth 64ppm of 30.5178125µs. Valid values are 0 ≤ WW_DSM_FACTOR ≤ 16. Consider the table below as a guide for selecting WW_DSM_FACTOR based on composite SCA:

| *masterSCA* + slaveSCA | Use this WW_DSM_FACTOR |
|---|---|
| 251 ppm to 500 ppm | 5 (320ppm) - 8 (512ppm) |
| 151 ppm to 250 ppm | 4 (256ppm) |
| 101 ppm to 150 ppm | 3 (192ppm) |
| 76 ppm to 100 ppm | 2 (128ppm) |
| 51 ppm to 75 ppm | |
| 31 ppm to 50 ppm | 1 (64ppm) |
| 21 ppm to 30 ppm | |
| 0 ppm to 20 ppm | |

**Figure 64-110. WW_DSM_FACTOR BASED ON COMPOSITE SCA**

A summary of all the hardware tasks performed to compute ww_dsm_adjust[19:0] during the COMP_WW state, is shown in the diagram below.

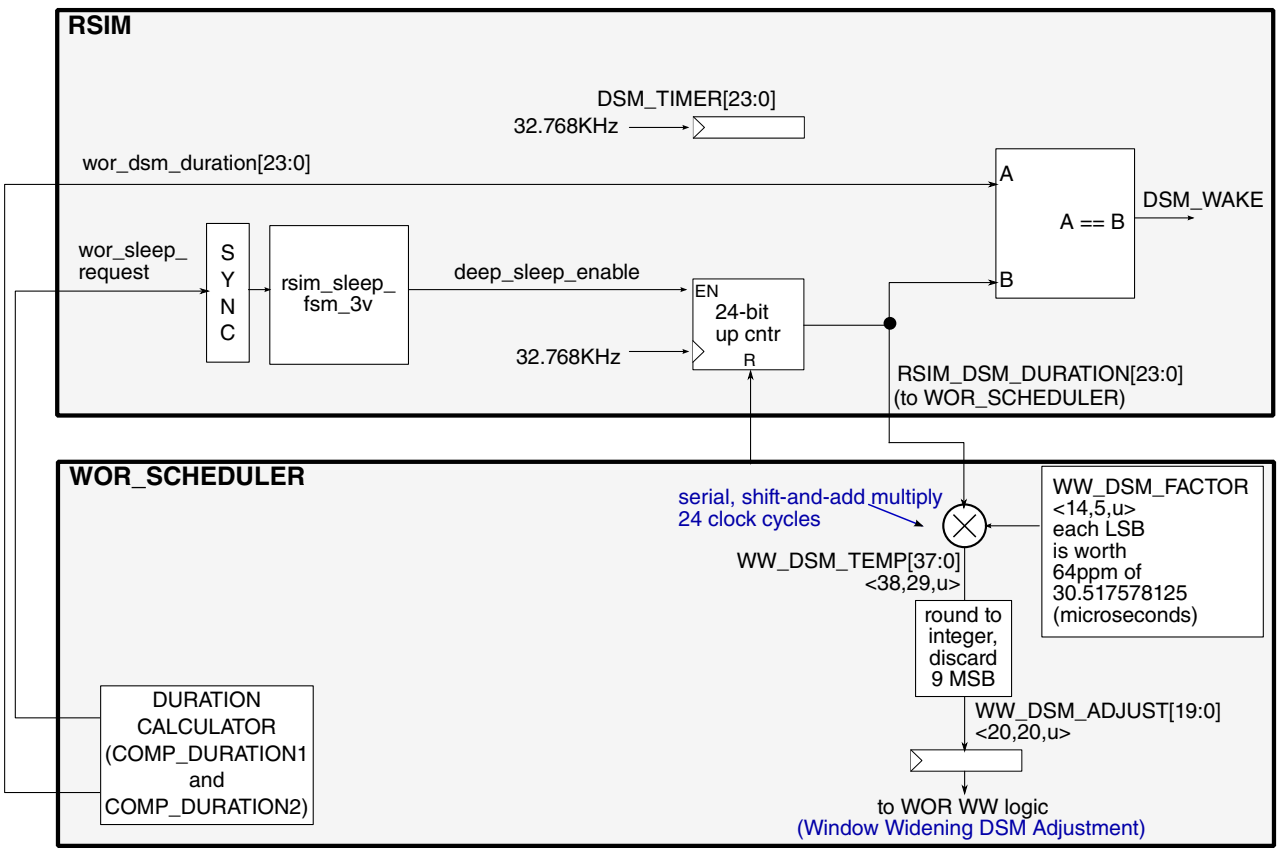



**Figure 64-111. HARDWARE TASKS DURING COMP_WW STATE**

## WINDOW WIDENING RUNTIME ADJUSTMENT

The window-widening RUNTIME adjustment is computed by incrementing a binary counter every *N* EVENT_TMR ticks whenever NOT in DSM (i.e., EVENT_TMR is running). *N* is a variable based on programmable register WW_RUN_ADJUST[4:0]. The programmable resolution, or granularity, of the DSM adjustment (64ppm), will be matched by the RUNTIME adjustment. Unlike the DSM adjustment, the RUNTIME adjustment is an approximation to the ideal value of *N*, since *N* is the result of a division operation whose quotient is not always an integer number of microseconds. However, the objective in Wake-On-Radio is to spend as little time in RUN between events as possible, so the DSM adjustment will dominate the sum. Therefore, additional resolution in the RUNTIME adjustment, although acheivable, would have little benefit in overall power savings.

*N* is a modulo for the binary counter which tracks the runtime adjustment, and represents the ideal number of microseconds between each increment to the adjustment, which is simply the reciprocal of the composite SCA, i.e.,

***N* = 1 / (master SCA + slave SCA)**

Using an example of composite SCA = 192ppm,

***N* = 1 / 192e-6 = 5208.333**

For implementation, each SCA range (the same table of ranges defined for the DSM adjustment) becomes an input to a look-up table (LUT) which approximates this division, rounding the quotient to the nearest integer. SCA range is represented by the WW_RUN_FACTOR. If WW_RUN_FACTOR=0, the RUNTIME adjustment never increments (special case). Valid values are 0 ≤ WW_RUN_FACTOR ≤ 16.

Every time the modulo *N* is reached, the binary counter resets to 0, and the RUNTIME adjustment is incremented by 1, meaning that the start of the next receive operation, has been advanced by 1µs. The hardware which implements the window widening RUNTIME adjustment is shown in the following diagram.

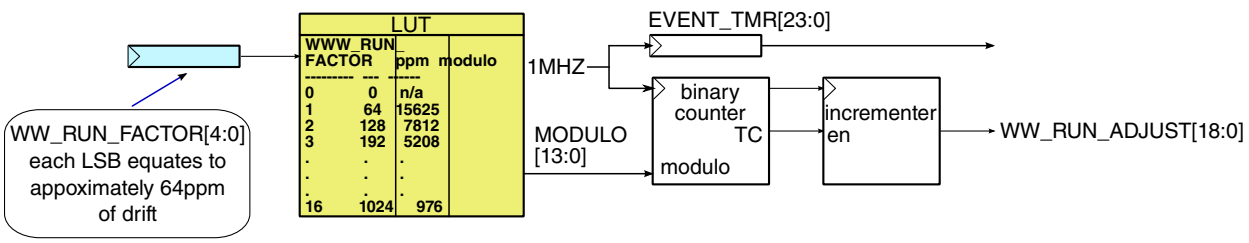



**Figure 64-112. WINDOW WIDENING RUNTIME ADJUSTMENT**

## COMPENSATING DSM DURATION FOR WINDOW WIDENING

In executing each slot as specified by the WOR descriptor, there is one other instance where window widening must be accounted for. During the COMP_DURATION1 state, WOR computes the DSM duration based on descriptor START_TIME and current (Event Timer) time. This duration is timed to complete the DSM cycle as close as possible to the desired START_TIME, but with sufficient guardband to ensure wakeup can never occur too late. But the duration computed during this state may be sufficiently long that the desired START_TIME needs to be advanced, due to drift within the DSM cycle itself. Thus, window widening needs to be estimated and subtracted from this computation. This window widening estimation is used only for the purpose of the duration estimation, and does not accumulate from one DSM cycle to the next (i.e., from one slot to the next). This window widening compensation to DSM duration occurs during the COMP_DURATION2 state. See Section State Machine for a description of WOR states.

The difference between this window widening computation, and the computation during COMP_WW, is that the result must be in units 32.768KHz clock cycles, because it is targetted for RSIM, instead of microseconds for an Event Timer offset. The COMP_DURATION2 computation multiplies the DSM duration (in 32K cycles) computed in COMP_DURATION1 by an estimate of the SCA (drift), to arrive at a

product which represents the number of 32.768KHz cycles by which to foreshorten the upcoming DSM cycle due to the widening requirement. So there is no conversion of units. In this case, 64ppm (0.000064) can be approximated by $1 * 2^{-14}$ and multiples of 64ppm can be approximated by $N * 2^{-14}$, with $2^{-14}$ representing a fixed bit-shift. So, for implementation, for all valid values, 0 ≤ WW_DSM_FACTOR ≤ 16, the value of *N* which most closely approximates the ideal multiple of 64ppm can be found in the following table:

| WW_RUN_FACTOR | *N* |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 10 |
| 11 | 12 |
| 12 | 13 |
| 13 | 14 |
| 14 | 15 |
| 15 | 16 |
| 16 | 17 |

The second multiplicand then becomes $N * 2^{-14}$, leading to the equation for number of 32K cycles of window widening which needs to be subtracted from DSM duration:

**CYCLES_OF_WW[14:0] = DSM_DURATION_NO_WW[23:0] (result of COMP_DURATION1, neglecting widening) * *N* * $2^{-14}$**

This product is subtracted from the DSM duration estimate, along with the DSM_GUARDBAND[3:0], to arrive at the final wor_dsm_duration[23:0] sent to RSIM for execution:

**WOR_DSM_DURATION[23:0] = DSM_DURATION_NO_WW[23:0] - CYCLES_OF_WW[14:0] - DSM_GUARDBAND[3:0].**

The following diagram illustrates all of the hardware tasks performed during COMP_DURATION2 state:

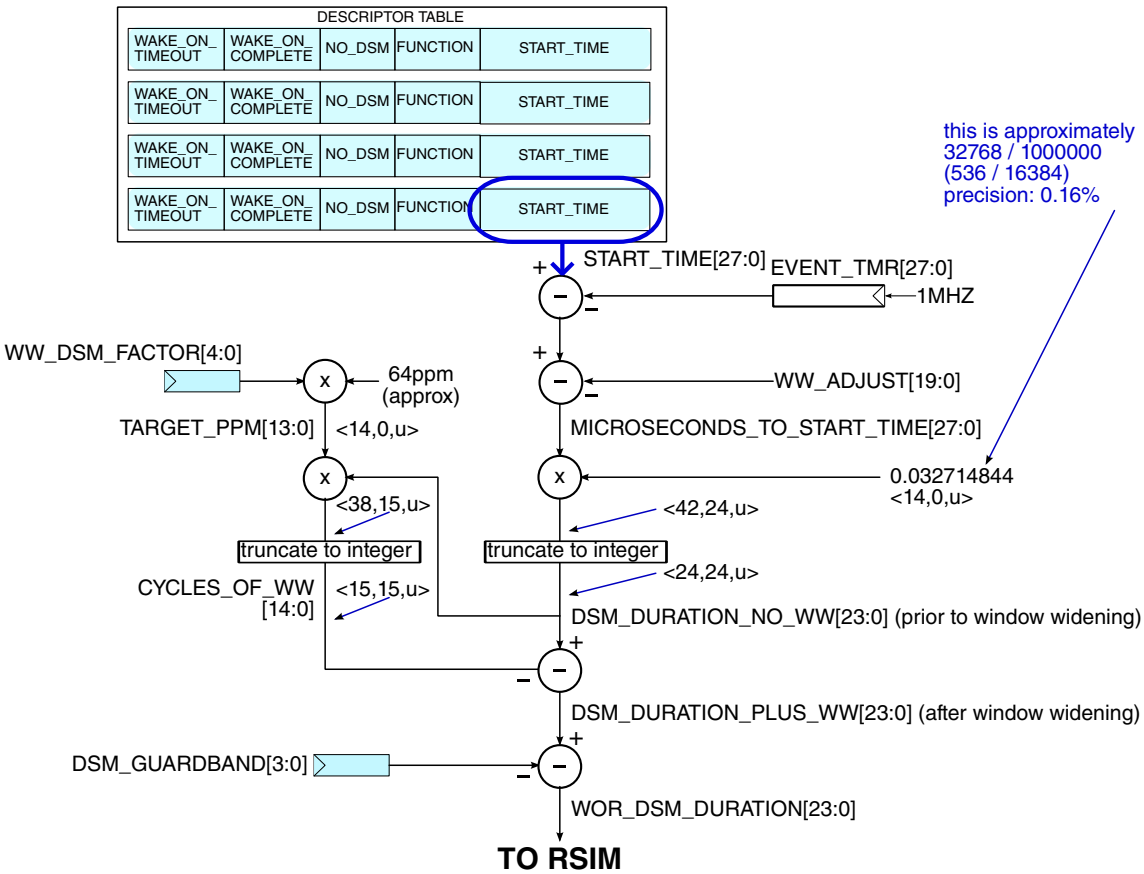



**Figure 64-113. COMP_DURATION2**

## APPLYING THE WINDOW WIDENING

Now that the window-widening adjustment has been computed, it must be put into effect, by modifying the START_TIME of the RF operation to advance it by the number of microseconds in ww_adjust[19:0], which is the combined DSM and RUNTIME window widening adjustment. This occurs during the WAIT_SEQ_START state. The same adjustment is applied again, by modifying the TIMEOUT of the RF operation (assuming timeout is reached), by delaying it by the same number of microseconds that START_TIME was advanced by. This occurs during the WAIT_SEQ_END state. The following diagram illustrates how window-widening is applied in Timeslot Scheduling mode.

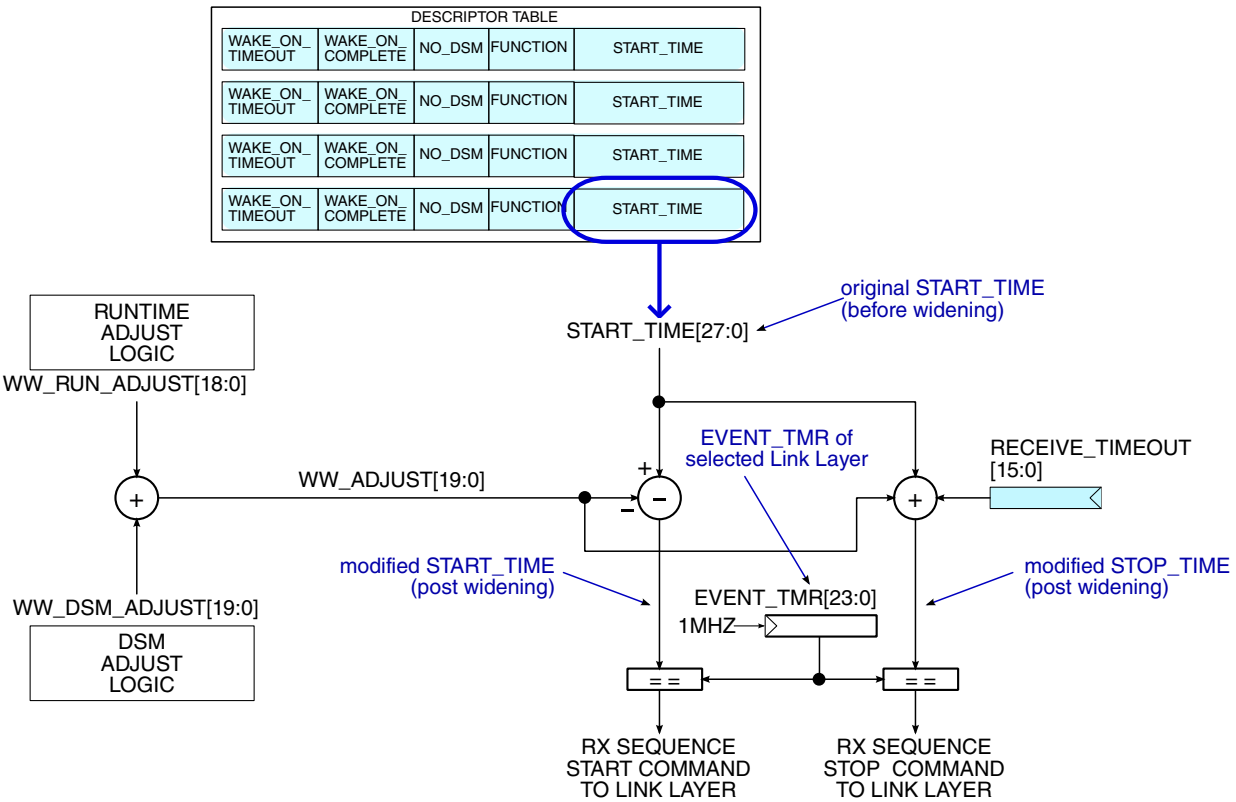



**Figure 64-114. APPLYING WINDOW WIDENING (TIMESLOT MODE)**

The application of window-widening in Constant-Interval scheduling mode, is largely the same as Timeslot mode; however the way the WOR hardware determines START_TIME and TIMEOUT is different, these need to be computed by WOR on a slot-by-slot basis. The following diagram illustrates how window-widening is applied in Constant-Interval Scheduling mode.

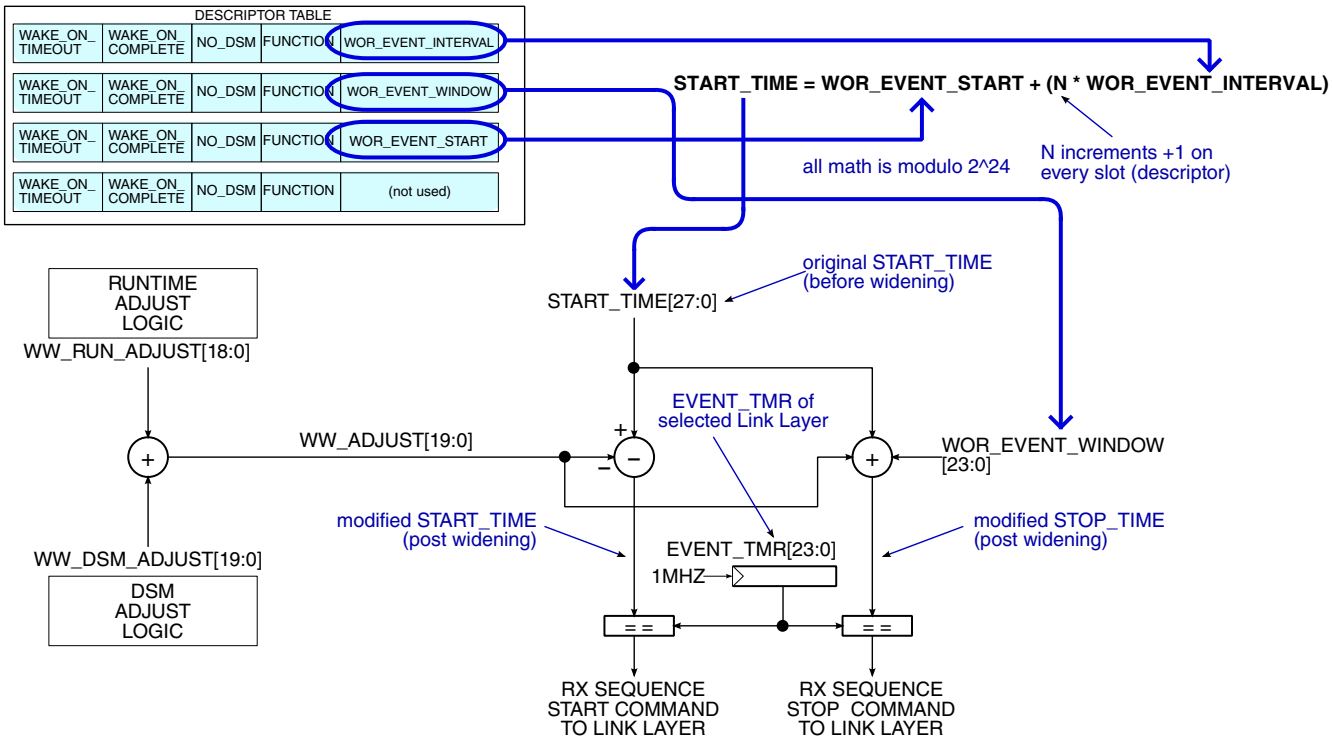



**Figure 64-115. APPLYING WINDOW WIDENING (CONSTANT-INTERVAL MODE)**

## NOTE

When dealing with the combination of high SCA (Sleep Clock Accuracy) as well as long requested DSM intervals, it is the responsibility of software to guarantee that the accumulation of window widening does not become excessively large. Window widening should not be allowed to accumulate to such an extent that the adjusted START_TIME is advanced so much that it moves to the left of the *previous* slot's START_TIME, or that the TIMEOUT is retarded so much that is moves to the right of the *next* slot's START_TIME, especially in scenarios where no good packets are received for an extended period of time. Software should periodically monitor window widening accumulation using mandatory MCU wakeups after every third or fourth slot (timeslot mode), or WAKE_ON_NTH_SLOT (constant interval mode), and use caution when scheduling long DSM intervals (distance between START_TIME on consecutive slots) with high WW_DSM_FACTOR values.

### 64.3.8.3.6 Timestamps

In Wake-On-Radio, every slot with FUNCTION=RX can capture a timeslot associated with any packet that is received in the slot. Any receive slot which results in an Access Address (for GENERIC_FSK protocol) or an SFD (for 802.15.4) detection will capture a timestamp into a register. Timestamps assist Link Layer software in maintaining and fine-tuning the network timebase across 2 or more networked devices.

**Timestamp Capture**

In Wake-On-Radio, each descriptor is associated with a slot. Any slot can be assigned an RX operation. RX operations require a timestamp to be captured. So, the timestamp can be thought of as being associated with the descriptor. The bit width of the timestamp is the same as the bit-width of the assigned Link Layer; i.e., 24 bits for GENERIC_FSK protocol, and 28 bits for 802.15.4. Association of timestamps to descriptors is illustrated in the following diagram.

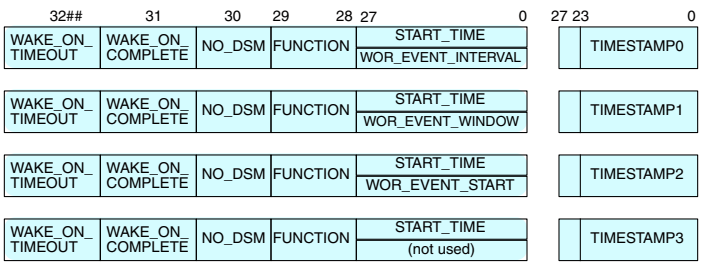



**Figure 64-116. TIMESTAMP CAPTURE**

Capture occurs at the instant AA or SFD is detected, and results in the current state of the assigned Link Layer's Event Timer being transferred to a TIMESTAMP register. WOR hardware does not apply any offset to the captured timestamp. A timestamp is captured for the last *N* RX operations, which result in (at least) one AA or SFD detection, where *N* = SLOTS_USED (typically 4).

Timestamp capture is area-intensive, due to:

- wide bit-width
- multiplicity (per descriptor)

Each of the supported protocol engines already has a TIMESTAMP register of the required bit-width. It is used for non-WOR (legacy) operations. To save area, when WOR is engaged, the TIMESTAMP register indigenous to the Link Layer, is used to capture the timestamp for WOR SLOT 0. Thus, in WOR mode (WOR_EN=1), the assigned Link Layer's TIMESTAMP register will only capture a timestamp when WOR SLOT=0, and will not change otherwise. If WOR is not engaged, the Link Layer TIMESTAMP register will capture all timestamps, the legacy behaviour. When WOR_EN, timestamps associated with SLOTS 1, 2, and 3 are captured in the TIMESTAMP1, TIMESTAMP2,

and TIMESTAMP3 registers, respectively, which are new, and resident in the WOR scheduler. This distribution of timestamp register resources is illustrated in the following diagram.

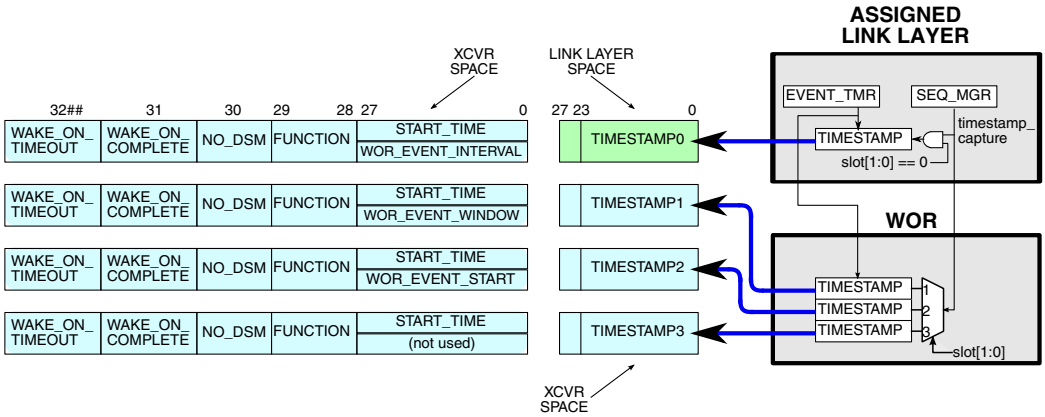



**Figure 64-117. TIMESTAMP REGISTER DISTRIBUTION**

Software should be aware of the fact that the designating the SLOT 0 timestamp to reside in the assigned Link Layer's TIMESTAMP register means that the TIMESTAMP regsters in Wake-On-Radio, are not all in one address space: TIMESTAMP0 resides in the assigned Link Layer's address space, while TIMESTAMP1-3 register reside in XVCR address space, alongside all the other WOR registers.

## TIMESTAMP STATUS

A timestamp will be captured into a TIMESTAMPx register on any slot assigned to FUNCTION RX, whenever an Access Address or SFD is detected during the receive window. Whenever a timestamp is captured as described above, a 3-bit read-only status register is updated with information regarding status of the reception which occured during the slot. The timestamp status, TIMESTAMPx_STS[2:0], becomes valid, and final (for the current slot), at the end of the receive window, and is made available to MCU when it is next awakened, along with the TIMESTAMP itself. The status information can be used by software to ascertain the validity of the timestamp. The following diagram describes timestamp status.

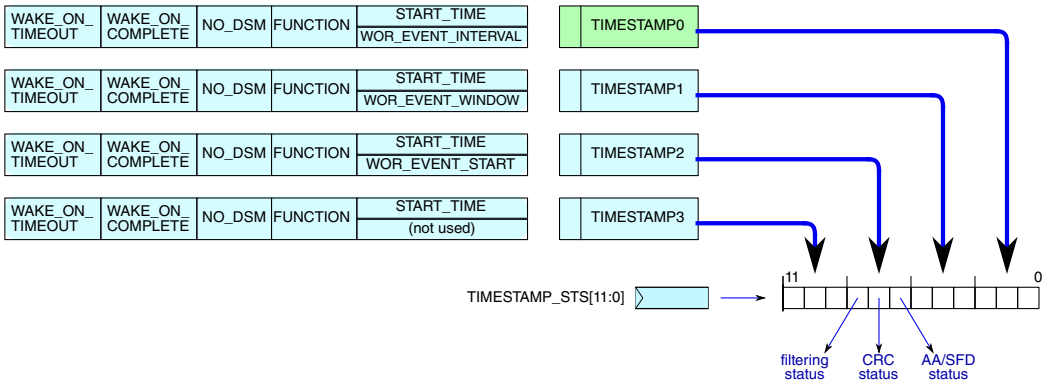



**Figure 64-118. TIMESTAMP STATUS**

A breakdown of the 3-bit status field, per timestamp, follows in the table below.

- 0: AA/SFD status
- 1: CRC status
- 2: Packet Filtering status

For any WOR slot assigned to RX, the 3-bit timestamp status field for that slot is cleared to 000 at the instant that WOR state WAIT_SEQ_START is reached. This is prior to the launch of the receive operation. (The bits are not cleared if the slot's function is not RX.) Once reception begins, the individual timestamp status bits are updated in realtime as the circumstances of the reception warrant. The 3-bit timestamp status is considered valid at the end of the receive window, and the bits reflect reception status per the following table.

| TIMESTAMPx_STS bit | Meaning if 0 | Meaning if 1 |
|---|---|---|
| [0] AA/SFD status | No AA or SFD was detected during the receive window. | At least one AA or SFD was detected. RX recycles can lead to multiple occurrences. |
| [1] CRC status | For the packet associated with last (most recently) detected AA or SFD, CRC verification failed. | For the packet associated with last (most recently) detected AA or SFD, CRC verification passed. |
| [2] Packet Filter status | For the packet associated with last (most recently) detected AA or SFD, at least 1 packet filtering rule was violated. | For the packet associated with last (most recently) detected AA or SFD, all packet filtering rules passed. |

Packet filtering rules, and hence the meaning of the TIMESTAMPx_STS[2] bit, vary across protocols, as summarized in the following table.

| PROTOCOL | MEANING OF TIMESTAMP_STSx[2] (PACKET FILTER) |
|---|---|
| 802.15.4 | Reflects the Packet Processor FILTERFAIL flag |
| GENERIC_FSK | There are 3 packet filter criteria:<br>1. H0 matching |

| PROTOCOL | MEANING OF TIMESTAMP_STSx[2] (PACKET FILTER) |
|---|---|
| | 2. H1 matching<br>3. LENGTH_MAX compliance (maximum packet length verification) |

Timestamp status bits are sticky. Once they become valid at the end of the receive window (end of WAIT_SEQ_END state), they remain valid until the *next* time the same slot number comes up again, and WOR reaches WAIT_SEQ_START state on that slot number.

In general, the MCU does not need to be notified on "bad packets." However, timestamp status may still be of value to WOR software even without Data Indication on any given slot. Notification policy can be modified at the protocol level, to enable RXIRQ on bad packets, mainly for debug purposes. If notification policy is modified by configuring the assigned Link Layer accordingly, this will affect Wake-On-Radio was well, as an rx_irq_trig will be generated on otherwise "bad" packets. See Section State Machine, specifically the WAIT_SEQ_END state, for details on how WOR responds to rx_irq_trig during this state, to determine how modifying notification rules would affect WOR response.

WOR software can use timestamp status to gauge the validity of the timestamp in any given slot. The following tables provides some guidelines.

| TIMESTAMPx_STS | TIMESTAMPx REGISTER VALIDITY | MEANING |
|---|---|---|
| 0b000 | stale | No AA or SFD was received during this timeslot. The TIMESTAMPx register was not updated by HW and continues to hold the timestamp captured after the last AA or SFD detection on this slot number. |
| 0bX01 | invalid | At least 1 AA or SFD was detected during this timeslot. For the last (most recent) AA or SFD detected, CRC verification failed. TIMESTAMP register holds the EVENT_TMR captured for this reception. Because CRC failed, the TIMESTAMP should be deemed invalid, but this decision is left to the application. |
| 0b011 | valid | At least 1 AA or SFD was detected during this timeslot. For the last (most recent) AA or SFD detected, CRC verification passed. TIMESTAMP register holds the EVENT_TMR captured for this reception, and can be deemed valid. However Packet Filtering failed, so typically there would be no Data Indication. |
| 0b111 | valid | At least 1 AA or SFD was detected during this timeslot. For the last (most |

| TIMESTAMPx_STS | TIMESTAMPx REGISTER VALIDITY | MEANING |
|---|---|---|
| | | recent) AA or SFD detected, CRC verification passed. TIMESTAMP register holds the EVENT_TMR captured for this reception, and is valid. Packet Filtering passed. |

#### 64.3.8.3.7 Interrupt Management

The Wake-On-Radio scheduler is protocol-neutral, so it has no interrupts of its own. WOR hardware utilizes the assigned Link Layer's interrupts for all MCU wakeups and notifications. In Wake-On-Radio, the MCU is in a low-power state most of the time. So, interrupts are allowed to assert only when it is mandatory to wake the MCU out of its low power state, to tend to required Link Layer or WOR tasks. WOR hardware modifies the interrupt activity of the assigned Link Layer by dynamically blocking and/or allowing Link Layer interrupts to propagate to the MCU, under control of the current slot descriptor specifications. Each of the 3 supported protocol engines have been modified to allow WOR to manage its RF-related interrupts. The RF-related interrupts have nearly identical functions and meanings across the 3 protocols:

- TXIRQ (or TX_IRQ)
- RXIRQ (or RX_IRQ)
- SEQ_END_IRQ (or SEQIRQ)

The modifications to the protocol engines, involve an AND gate inserted between each interrupt trigger, and the interrupt status bit, allowing or blocking the trigger from asserting the status bit. A gating signal, wor_allow_irq, is provided by WOR to each protocol engine to enable/disable the AND gate. The gating signal is updated dynamically by the WOR state machine based on state, RF operation status, and the current descriptor's WAKE_ON_* fields. See Section State Machine, specifically the description of the WAKE_SEQ_END state, for more details on how and when WOR allows and blocks RF-related interrupts. For protocols not assigned to WOR, interrupts are always allowed.

The raw interrupt triggers themselves (the inputs to the AND gates), namely, tx_irq_trig and rx_irq_trig, are passed directly to WOR hardware, so that WOR state machine can react to RF operation status in realtime, and sequence through the states as specified by the current slot's descriptor.

WOR manages one other Link Layer interrupt, for non-RF-related notifications. Each protocol engine has an existing interrupt, WAKE_IRQ, to notify MCU on wakeup from Manual DSM mode (not Wake-On-Radio). See Section Manual DSM for more details on

this DSM control scheme. For any protocol, Wake-On-Radio and Manual DSM are mutually-exclusive control schemes, so when a protocol is assigned to WOR, its WAKE_IRQ is re-purposed, and re-named, to WOR_IRQ.

Like the "allow/block" gating WOR uses to manage the RF-related interrupts, WOR uses 2 signals to either block or *force* a WAKE_IRQ / WOR_IRQ:

- **wor_block_wake_irq:** prevents the (legacy) WAKE_IRQ from asserting at DSM exit, this is *mandatory* for Manual DSM, but is *conditional* for WOR.
- **wor_force_wake_irq:** WOR needs to assert WOR_IRQ under 3 conditions (see below)

There are 3 conditions for which WOR needs to compel the Link Layer to assert WOR_IRQ. WOR maintains 3 read-only status flags to indicate each of these conditions. The status flags exist in XCVR (WOR) address space, even though the WOR_IRQ resides in the protocol-specific address space. The 3 conditions, and the associated status flag for each, are described in the following table.

- **WOR_NO_RF_FLAG:** START_TIME has been reached on a slot with FUNCTION = NO_RF and WAKE_ON_COMPLETE=1
- **WOR_MAX_SLOT_FLAG:** In Constant-Interval scheduling mode, the WOR_SLOT_COUNT[7:0] has reached WAKE_ON_NTH_SLOT[7:0]
- **WOR_DSM_EXIT_FLAG:** An unscheduled, asynchronous DSM "Early Exit" has occurred

To service an WOR_IRQ, the interrupt service routine should query the 3 read-only status flags to determine the cause of the MCU wake up. At least one of these flags will be set. It is possible that more than 1 of the flags is set. For the WOR_NO_RF_FLAG, WAKE_ON_COMPLETE would typically not be set on a NO_RF slot, since the purpose of NO_RF is to allow "DSM cascading"; only for debug purpose has this flag been provided. Servicing of the other 2 flags is beyond the scope of this document.

Clear the WOR_IRQ by writing a 1 to this bit in the protocol-specific address space. However the 3 WOR flags are "sticky". Any flag which is set will remain set until a new WOR session has begun, which means taking WOR to DISABLED by setting SLOTS_USED=0, followed by re-enabling by setting SLOTS_USED>0.

In initializing the assigned Link Layer for WOR operation, it is *essential* that software enable (unmask) the 3 RF-related interrupts, as well as WAKE_IRQ / WOR_IRQ. It is equally essential that all non-WOR related interrupt sources are *disabled* when a protocol is assigned to WOR, to prevent unnecessary MCU wakeups. The following tables describe all the Link Layer interrupts, their controllability (or lack of) by WOR, and the required enabling/unmasking by software.

## INTERRUPT CONFIGURATION (802.15.4)

| INTERRUPT | WOR CONTROL | SW INIT | COMMENTS |
|---|---|---|---|
| TXIRQ | gated by wor_irq_allow | ENABLE | Asserted after packet transmission when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT |
| RXIRQ | gated by wor_irq_allow | ENABLE | Asserted after good packet reception when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT |
| SEQIRQ | gated by wor_irq_allow | ENABLE | 1. Asserted after any good packet transmission or reception when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT<br>2. Asserted after bad (or no) packet reception when: WAKE_ON_TIMEOUT |
| CCAIRQ | unaffected | DISABLE | Not used in WOR |
| RXWTRMRKIRQ | unaffected | DISABLE | Not used in WOR |
| FILTERFAIL_IRQ | unaffected | DISABLE | Not used in WOR |
| PLL_UNLOCK_IRQ | unaffected | OPTIONAL | If enabled, would cause an immediate MCU LLWU on occurrence of a PLL unlock event. |
| WAKE_IRQ / WOR_IRQ | blocked by wor_block_wake_irq, forced by wor_force_wake_irq | ENABLE | 1. Asserted when START_TIME reached on a slot with FUNCTION=NO_RF and WAKE_ON_COMPLETE=1<br>2. Asserted in CI mode when WOR_SLOT_COUNT==WAKE_ON_NTH_SLOT<br>3. Asserted after an unscheduled "DSM Early Exit" |
| TMRxIRQ | unaffected | DISABLE | Not used in WOR |
| TSM_IRQ | unaffected | DISABLE | Not used in WOR |

## INTERRUPT CONFIGURATION (non-802.15.4)

| INTERRUPT | WOR CONTROL | SW INIT | COMMENTS |
|---|---|---|---|
| TX_IRQ | gated by wor_irq_allow | ENABLE | Asserted after packet transmission when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT |

*Table continues on the next page...*

| INTERRUPT | WOR CONTROL | SW INIT | COMMENTS |
|---|---|---|---|
| RX_IRQ | gated by wor_irq_allow | ENABLE | Asserted after good packet reception when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT |
| SEQ_END_IRQ | gated by wor_irq_allow | ENABLE | 1. Asserted after any good packet transmission or reception when: WAKE_ON_COMPLETE \|\| WAKE_ON_TIMEOUT<br>2. Asserted after bad (or no) packet reception when: WAKE_ON_TIMEOUT |
| NTW_ADR_IRQ | unaffected | DISABLE | Not used in WOR |
| T1_IRQ | unaffected | DISABLE | Not used in WOR |
| T2_IRQ | unaffected | DISABLE | Not used in WOR |
| PLL_UNLOCK_IRQ | unaffected | OPTIONAL | If enabled, would cause an immediate MCU LLWU on occurrence of a PLL unlock event. |
| WAKE_IRQ / WOR_IRQ | blocked by wor_block_wake_irq, forced by wor_force_wake_irq | ENABLE | 1. Asserted when START_TIME reached on a slot with FUNCTION=NO_RF and WAKE_ON_COMPLETE=1<br>2. Asserted in CI mode when WOR_SLOT_COUNT==WAKE_ON_NTH_SLOT<br>3. Asserted after an unscheduled "DSM Early Exit" |
| TSM_IRQ | unaffected | DISABLE | Not used in WOR |
| RX_WATERMARK_IRQ | unaffected | DISABLE | Not used in WOR |

### 64.3.8.3.8 Channel Hop Table

Typically, in either Timeslot or Constant-Interval scheduling mode, RF operations on consecutive slots use different RF channels. Using Wake-On-Radio to program slot specifications for the next few RF operations in advance, would be of little value if the MCU had to wake up on every slot to program the assigned Link Layer's CHANNEL_NUM register. Thus, Wake-On-Radio provides hardware automation for channel hopping on a slot-by-slot basis.

A table of channel hop frequencies is maintained by hardware. The number of entries in the table (hop sequence length) is programmable. Prior to starting a WOR session, software will program the desired sequence of channel numbers into the table. After WOR is activated by setting SLOTS_USED>0, The WOR state machine controls the index (read pointer) into the table. The index is typically incremented at the end of every slot, when the slot counter is incremented. The increment is +1 modulo "hop sequence length". The RF operation on the current slot, is then carried out on the RF channel pointed to by the hop table index. If necessary, WOR or Link Layer software can update the hop table index at any time.

Rather than adding a channel number field to the descriptor, to save area, the hop table is decoupled from the descriptor table. The hop tables utilizes a small section of Packet RAM which is not otherwise used by the supported WOR protocols. WOR uses the uppermost 48 of those words of Packet RAM #1 for the channel hop table, starting at RAM word address 1104 (decimal), of Packet RAM #1, and ending at address 1151.

The channel information entered into the hop table, can use the (legacy) 7-bit "mapped channel" format, which allows channels to be specified on a 1MHz boundary; or one of 2 new 16-bit formats that allows channel frequency of higher resolution. The option for the legacy format, provides backwards compatability to existing software, which does not use WOR, to program RF channel information the way it does today, without any changes. The inclusion of the new formats allows additional flexibility for programming channels which deviate from a 1MHz boundary, coupled with the automation built in to WOR. The format selected, determines:

1. how the hop table is organized in RAM
2. how software programs the hop table
3. how the WOR parses the table during execution of the current slot
4. and how the PLL digital block interprets the frequency word extracted from the table.

Selecting the hop table configuration is by way of the HOP_TBL_CFG[2:0] register. Based on this setting, the source of channel information, can either be the CHANNEL_NUM[6:0] bit register that exists in each protocol engine today (legacy), or the new hop table, according to the following table:

Source of RF channel information, based on HOP_TBL_CFG[2:0]:

0: Assigned Link Layer's CHANNEL_NUM register (legacy)
1: WOR Hop Table
2: WOR Hop Table
3: WOR Hop Table
4-7: Reserved

To allow for the legacy option, the 3 protocol-specific CHANNEL_NUM registers are now routed to WOR, which multiplexes them with the hop table output based on HOP_TBL_CFG and WOR_PROTOCOL settings. WOR then outputs 3 16 bit frequencies words, one per protocol, to PLL Digital, conveying the required channel information.

Hop table organization is determined by HOP_TBL_CFG, as per the following table:

Hop Table Organization, based on HOP_TBL_CFG[2:0]:

0: N/A
1: Table size is 96 entries. Two 7-bit CHANNEL_NUM fields are packed into a 16-bit hop frequency word for RAM storage
2: Table size is 48 entries. Each 16-bit hop frequency word in RAM contains a specification for a single RF channel
3: Table size is 48 entries. Each 16-bit hop frequency word in RAM contains a specification for a single RF channel
4-7: Reserved

The content of the hop frequency word provided to PLL Digital, and how it is interpreted by that block, also varies with HOP_TBL_CFG, as per the following table:

PLL Digital Interpretation of Hop Frequency Word, based on HOP_TBL_CFG[2:0]:

0: Lower 7 bits of hop frequency word contain a mapped channel (legacy format). Upper 9 bits are 0
1: Lower 7 bits of hop frequency word contain a mapped channel (legacy format). Upper 9 bits are 0
2: All 16 bits are valid. See Chapter *PLL Digital* for interpretation
3: All 16 bits are valid. See Chapter *PLL Digital* for interpretation
4-7: Reserved

The following diagram summarizes the routing of channel hop signalling amongst the supported protocol engines, the WOR scheduler, the PLL Digital, and Packet RAM. It also shows the 2 hop table configurations, and hop frequency word generation, based on HOP_TBL_CFG. The existing (legacy) configuration is on the left, and the new, WOR-based configuration is on the right.

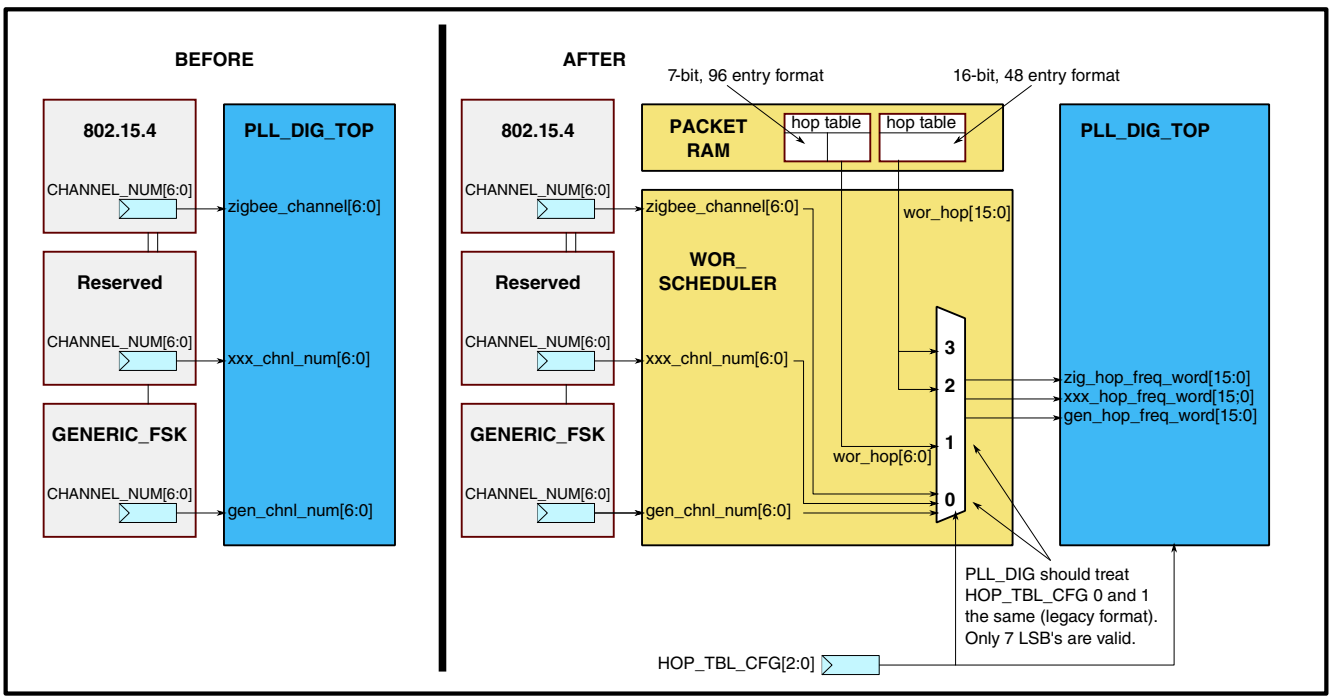



**Figure 64-119. CHANNEL HOP SIGNAL ROUTING AND CONFIGURATION**

The number of entries in the table is determined by programmable register HOP_SEQ_LENGTH[6:0]. If HOP_TBL_CFG=1, HOP_SEQ_LENGTH determines the number of 7-bit CHANNEL_NUM entries in the Packet RAM table, which are packed 2 per RAM word. HOP_SEQ_LENGTH must not exceed 96 in this case. If HOP_TBL_CFG=2 or HOP_TBL_CFG=3, HOP_SEQ_LENGTH determines the number of 16-bit format entries in the Packet RAM table, one per RAM word. HOP_SEQ_LENGTH must not exceed 48 in this case.

At the start of new WOR "session" (setting SLOTS_USED to a non-zero value), WOR hardware initalizes the table index (read pointer) to 0. (Software can modify this index, see below.) After slot execution completes, as the WOR state machine increments the slot counter to service the next slot, the hop table index is incremented by +1 (modulo HOP_SEQ_LENGTH). An exception is made to this rule, if the FUNCTION field of the slot descriptor is set to NO_RF, the hop index is *not* incremented when the slot counter is incremented to advance to the next slot. The NO_RF function is used for "DSM cascading".

Software can take control of the hop table index at any time, by writing a new desired index to the NEW_HOP_IDX[6:0], and with the UPDATE_HOP_IDX register set to 1. These 2 steps can be performed as a single, atomic IPS bus write. WOR hardware will update the index accordingly. From this point forward, the hop-table hardware incrementing rules, discussed in the previous paragraph, apply. This way, software can execute a one-time "jump" in the hop table indexing.

The read-only register WOR_HOP_IDX[6:0] allows software to read the instantaneous, hardware-controlled hop table index at any time. Similarly, the read-only register WOR_HOP_FREQ_WORD[15:0] allows software to read the instantaneous, hardware-generated frequency word sent to PLL digital. This word is the output of the hop table configuration selection mux shown in the diagram above, the protocol-specific word currently selected by WOR_PROTOCOL.

A diagram showing the RAM-based, 96-entry hop table (HOP_TBG_CFG=1), where it is situated in Packet RAM space, how the channel information is organized in the table, and how WOR hardware indexes the table, is shown below.

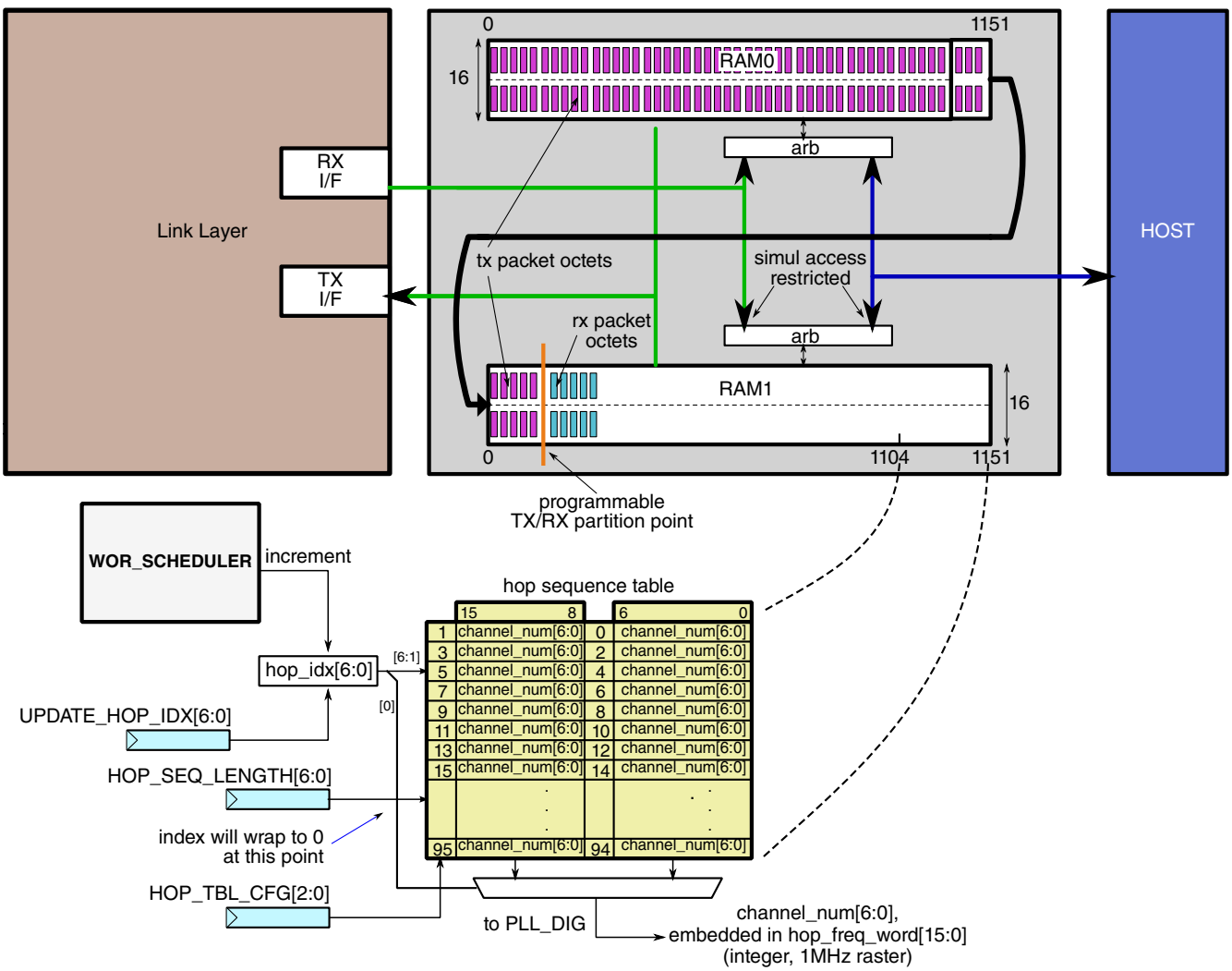



**Figure 64-120. 96-ENTRY HOP TABLE**

A diagram showing the RAM-based, 48-entry hop table (HOP_TBG_CFG=2 or 3), where it is situated in Packet RAM space, how the channel information is organized in the table, and how WOR hardware indexes the table, is shown below.

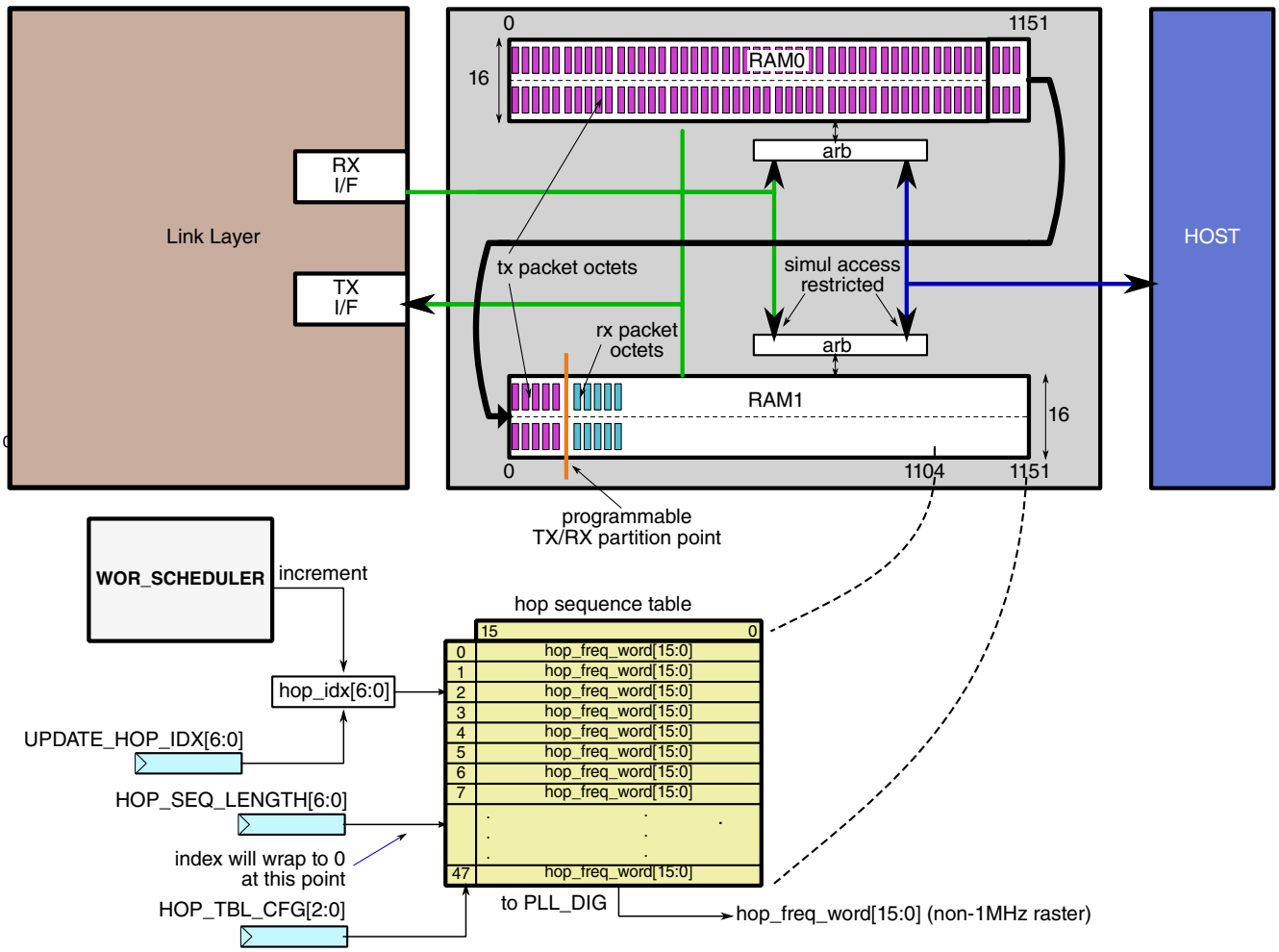



**Figure 64-121. 96-ENTRY HOP TABLE**

## PROGRAMMING INFORMATION AND MEMORY MAP

The upper 96 bytes of Packet RAM are reserved for the WOR hop table. The table is fixed in this position, and does not move based on HOP_TBL_CFG. For purposes of programming the hop table, Packet RAM should be accessed through XCVR address space (see instructions below), since the hop table is not protocol-specific. The following diagram illustrates how the hop table is memory-mapped into XCVR space.

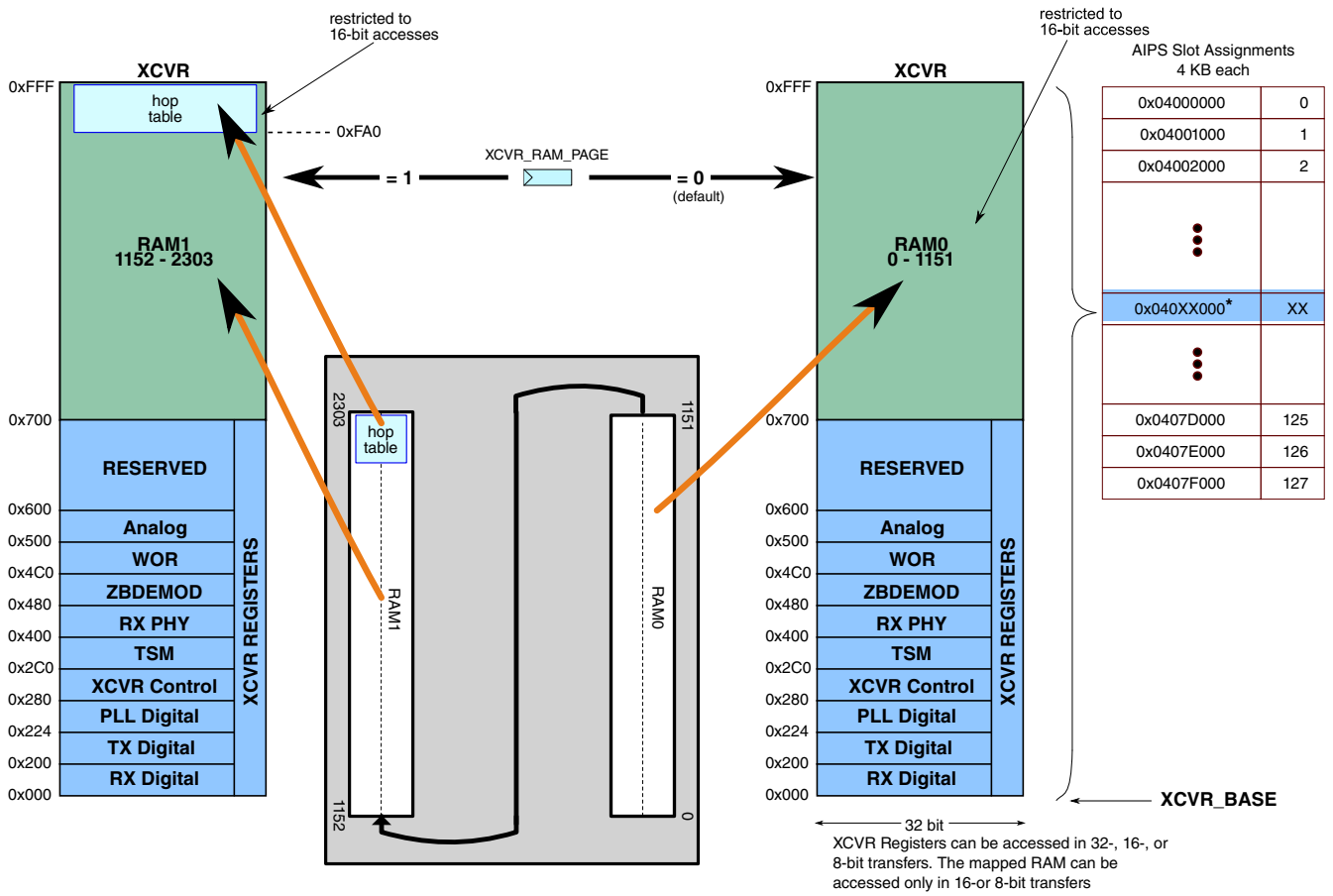



**Figure 64-122. MEMORY MAPPED HOP TABLE**

As shown, the hop table base is at: **XCVR_BASE + 0xFA0**, of Packet RAM #1 (PACKET_RAM_CTRL[RAM_PAGE]=1 in XCVR space)

The WOR hop table should not be accessed (no writes or reads) while an RF operation is underway. There is no hardware to de-conflict XCVR RAM accesses with those originating in the protocol engines. Aside from this restriction, the hop table can be accessed at any time. A convenient time to modify the hop table, if required, would be during the WOR WAIT_RESUME state.

To access Packet RAM through XCVR space, the PACKET_RAM_CTRL[XCVR_RAM_ALLOW] bit must be set. Follow the instructions below to program the WOR hop table.

1. Set PACKET_RAM_CTRL[XCVR_RAM_ALLOW]=1 (to allow exclusive XCVR access to Packet RAM)
2. Set PACKET_RAM_CTRL[RAM_PAGE]=1 (to select Packet RAM #1)
3. Program the hop table starting at XCVR_BASE + 0xFA0
4. Program HOP_CTRL[HOP_SEQ_LENGTH] with the desired size of the hop sequence

5. Program HOP_CTRL[HOP_TBL_CFG] to select the desired table configuration and enable hopping during WOR
6. Set PACKET_RAM_CTRL[XCVR_RAM_ALLOW]=0 (to allow Link Layer access to Packet RAM)

#### 64.3.8.3.9 DSM Early Exit

As described in Section State Machine, for every slot, WOR hardware computes the duration of the upcoming DSM cycle based on descriptor START_TIME, current time, and window-widening, during the COMP_DURATION1 and COMP_DURATION2 states. WOR outputs the desired duration on wor_dsm_duration[23:0]. The Radio System Integration Module (RSIM) then executes this DSM for the prescribed number of 32.768KHz clock cycles, and when complete, sends the actual duration back to WOR using rsim_dsm_duration[23:0]. Normally, these 2 durations will match precisely.

Conditions exist where the DSM cycle may occasionally need to be cut short, due to circumstances beyond the control of the radio. The SoC, starting with the MCU, must be awakened out of its low power state before the WOR DSM duration expires. When this happens, software handling the event which triggered the wakeup, will determine if the radio needs to be awakened as well. If this is the case, software will set the RSIM_WOR_DSM_EXIT bit. This will trigger an unscheduled wakeup process out of DSM by the RSIM deep sleep state machine. The wakeup will complete before the wor_dsm_duration[23:0] time has expired. RSIM will capture the *actual* DSM duration, cut short by the early exit, into rsim_dsm_duration[23:0]

After DSM wakeup, WOR hardware always performs this check during the COMP_WW state:

**rsim_dsm_duration[23:0] == wor_dsm_duration[23:0]**

If this check fails, WOR hardware declares an "DSM Early Exit", and takes the following steps:

1. Forces the assigned Link Layer to assert its WAKE_IRQ / WOR_IRQ (see Section Interrupt Management)
2. Sets the WOR_DSM_EXIT_FLAG in WOR space
3. Transitions to the WAIT_RESUME state and waits for software to service the WOR_IRQ.

Software should then take whatever steps are necessary to configure the radio to respond to the conditions which triggered the early exit. Servicing of a DSM early exit interrupt is beyond the scope of this document.

#### 64.3.8.3.10 Manual DSM

Prior to Wake-On-Radio, implementation of the 2.4GHz used a software-driven, "manual" DSM. This DSM control scheme does not have any of the hardware automation associated with WOR, and software must perform such tasks as computing the Event Timer addback time, window-widening, etc.

Manual DSM still exists as an alternative to Wake-On-Radio. Manual mode can be of value in a multiprotocol setting. See Section Multiprotocol for more details.

Any of the 3 protocols supported by Wake-On-Radio, can utilize Manual DSM as an alternative to Wake-On-Radio. In short, Manual DSM works by programming a desired DSM start time into the MAN_SLEEP register, and a desired DSM wake time into the MAN_WAKE register. The MAN_SLEEP and MAN_WAKE registers are referenced to the 32.768KHZ DSM_TIMER, the radio's low-power (DGO) timer. MAN_SLEEP, MAN_WAKE, and DSM_TIMER reside in RSIM address space.

When a match on DSM_TIMER to MAN_SLEEP occurs, RSIM places the radio in deep sleep mode. The RF Oscillator is shut off, and the Link Layer's Event Timer is frozen in place. When a match on DSM_TIMER to MAN_WAKE occurs, the Event Timer resumes from where it left off. The RF Oscillator, is re-started as part of the DSM wake up process, and is running before MAN_WAKE is reached. At the instant the Event Timer resumes, the Link Layer asserts its WAKE_IRQ. The MCU should use the WAKE_IRQ ISR to compute the precise amount of time the radio was in DSM, convert this value to microseconds, and add this value to the current Link Layer Event Timer, to restore the timer to where it would have been, had DSM not occurred. There is a 1us loss-of-precision in the Event Timer timebase incurred on every DSM wakeup. This timebase error is random, and averages out over many DSM cycles.

Regarding the DSM wakeup process, there is additional programmability to configure the lead time for power-domain wakeup and oscillator re-start, relative to DSM exit (MAN_WAKE). This configurability is covered by the RSIM_DSM_PWR_OFFSET register. See the *Radio System Integration Module* chapter for more details.

#### 64.3.8.3.11 Multiprotocol

The 2.4GHz radio supports 4 protocols, any 2 of which can be enabled for multiprotocol use on an alternating, time-shared basis. Both of the active protocols may require DSM. The following section describes the rules to be applied when assigning a protocol to a DSM control scheme.

The radio has 3 DSM control schemes:

1. BLE-internal
2. Wake-On-Radio

3. Manual DSM

The BLE-internal DSM controller applies to the BLE protocol engine only, and is the only DSM option for BLE.

Wake-On-Radio can be assigned to any one of the following protocols (see Section *Assign To Protocol*):

1. 802.15.4
2. GENERIC_FSK

Manual DSM can be assigned to any one of the following protocols (see Section *Manual DSM*):

1. 802.15.4
2. GENERIC_FSK

#### 64.3.8.3.12 Hardware Multiplier

Wake-On-Radio requires 4 multiplies while executing each slot. The multiplications take place in the following WOR states (See Section State Machine):

1. COMP_DURATION1: to compute the upcoming DSM duration, excluding window widening
2. COMP_DURATION2: to compute the subtractor to the upcoming DSM duration, for window widening
3. COMP_ADDBACK: to compute the Event Timer addback value based on RSIM-provided DSM duration
4. COMP_WW: to compute the DSM-component of window-widening incurred during the just-completed DSM cycle

For maximum area and power saving, there is 1 hardware multiplier instance, which is timeshared for the 4 multiplies. Also, because the state machine has been architected to allow sufficient time to perform these computations before the results are needed, the multiplier uses a serial, add-and-accumulate construction. The multiplicands are 24 bits and 14 bits. The 24 bit operand is appropriately sized for the "duration" calculations, since this is the bit-width of the DSM timer. The 14 bit operand is appropriately sized to achieve the precision of the required calculations, as described in the earlier sections.

For each calculation, the multiplier requires 14 clocks (14 shifts). Each multiply takes place within the confines of a single state of duration 1µs. There are 32 periods of the reference oscillator during that period (26 for a 26MHz crystal), more than enough to complete the multiply. The clock to the multiplier runs at the reference oscillator rate, but runs only during the 4 1µs states listed above, to maximize power savings.

The multiplier output is 42 bits. Depending on the calculation, the required precision, the positioning of the integer bits within the product, etc., the bits of the product tapped for each calculation varies. Also, depending on the calucation, combinational rounding or truncation of the product occurs, external to the multiplier.

#### 64.3.8.3.13 Clocks and Gating

Because of the intensive interaction between Wake-On-Radio and 3 protocol engines it supports, the WOR hardware runs at 1MHz. This facilitates signalling between the WOR and the assigned Link Layers, with no additional pulse-stretching or signal-conditioning required. Like the protocol engines and the TSM, the CGM's cg1m_en gating signal is used to generate the common 1MHz, so the phasing of the clocks across all the communicating modules remains in sync.

All WOR clocks are conditioned on WOR_EN=1. This bit should be set, and remain set, whenever Wake-On-Radio is in use (it should not be toggled or modified dynamically by software, since module clock-gating is based on it). This bit should not be set if WOR is not in use; that way, the WOR hardware block will consume zero dynamic power.

In total, there are 4 clocks used by WOR hardware:

1. **ipg_clk_wor_1m**: Most of the WOR hardware runs off this 1MHz clock, including the state machine, descriptor processing, and hop table indexing. All link layer interface outputs are generated with this clock, and link layer inputs sampled by it.
2. **ipg_clk_wor_ww_1m**: Identical in timing to ipg_clk_wor_1m, but used only by functions that require window widening to be enabled. The additional clock gating granularity saves power when widening is disabled.
3. **ipg_clk_wor_comp**: This is the "compute clock" used by the serial multilier. This clock runs at the reference oscillator frequency (derivative of xcvr_ref_clk_gated), but only during the 4 1µs-long states (per slot) during which the multiplier is in use; See Section Hardware Multiplier)
4. **ipg_clk_wor_timestamp**: This is the "timestamp clock". Because the protocol engines determine the timestamp-capture timing, based on AA/SFD detection, the clock-gate enable for the capture is sent to the CGM, which routes the resulting timetamp clock itself to WOR so that it can update its TIMESTAMP1,2, and 3 registers. This clock will typically pulse once per RX slot, and not at all for slots of other types.

All gating for the WOR clocks is performed in CGM. WOR provides the gating-enable signal, and CGM provides the clock.

All WOR clocks are in the Reference Oscillator domain. There are no clock domain crossings inside the module.

No WOR critical timing paths are anticipated, nor any timing exceptions (e.g., multicycle paths).

## 64.3.9 Cyclic Redundancy Check and Data Whitening Module (CRCW)

### 64.3.9.1 Introduction

The cyclic redundancy check and data whitening module (CRCW) contains logic necessary to provide CRC and Whitening for the radio link layer functions. Packet data is provided one bit at a time by the link layer functions on transmit, or by the radio physical layer on receive. Control and mode input signals dictate how the CRCW interprets and modifies the bit stream. The CRCW does not add any latency to the data as it is transmitted or received.

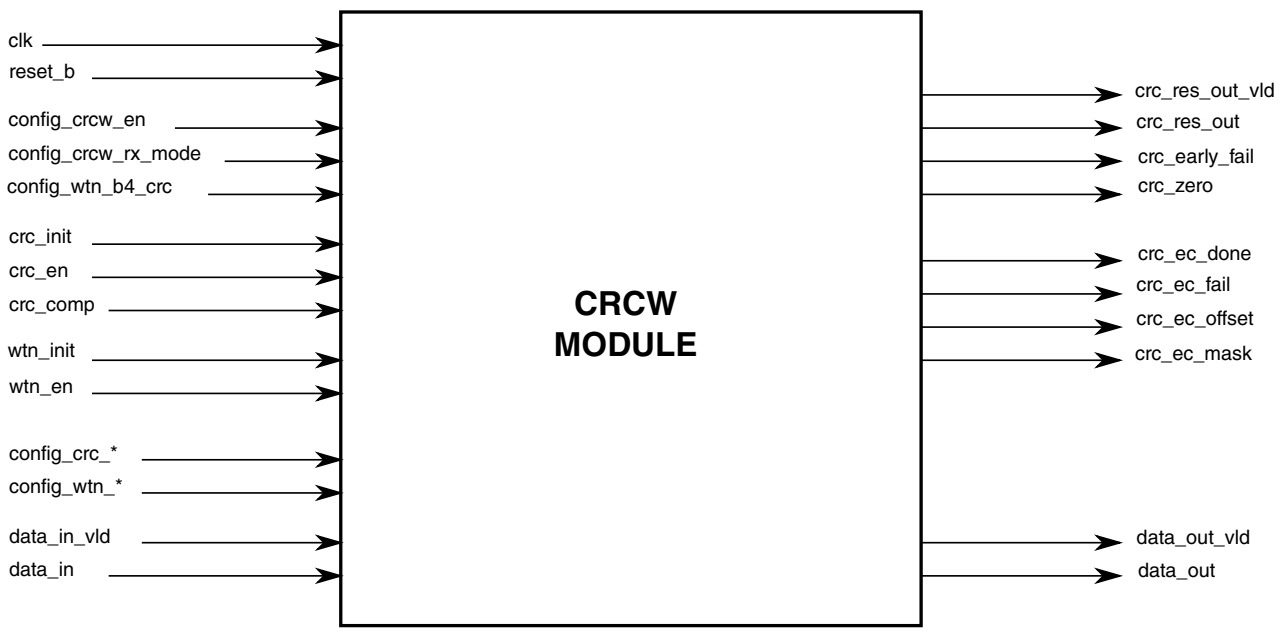



**Figure 64-123. CRCW Block Diagram**

#### 64.3.9.1.1 Features

- Zero added data latency
- One module used for both transmit and receive
- All modes work at bit rates up to 1/10$^{th}$ the system clock rate
- Configurable CRC
  - 8/16/24/32-bit CRC register width

- Programable polynomial and seed values
- CRC value byte and bit order
- Payoad input bit reflect

- Configurable Data Whitening
  - 1-bit to 9-bit LFSR width
  - Programable polynomial and seed values
  - Galois or Fibonacci type LFSR
  - Input data bit reflect
- Optional CRC Error Correction
  - Correct limited burst errors
  - Seperate settings for short and long packets
  - Offset/Mask reported to software for final data correction

#### 64.3.9.1.2 Operation

The CRCW module contains the functionality to perform transmit CRC insertion, receive CRC checking, and data whitening. Additionally, the module can perform data error correction on received packets using the CRC syndrome if configured appropriately.

##### 64.3.9.1.2.1 CRC Calculation

The CRC value is calculated using a configurable LFSR as depicted in Figure 64-124. The CRC calculation polynomial value is loaded into register *P*. The seed is loaded into register *SR* at initilization. The result is in register *SR* after all bits have been processed.

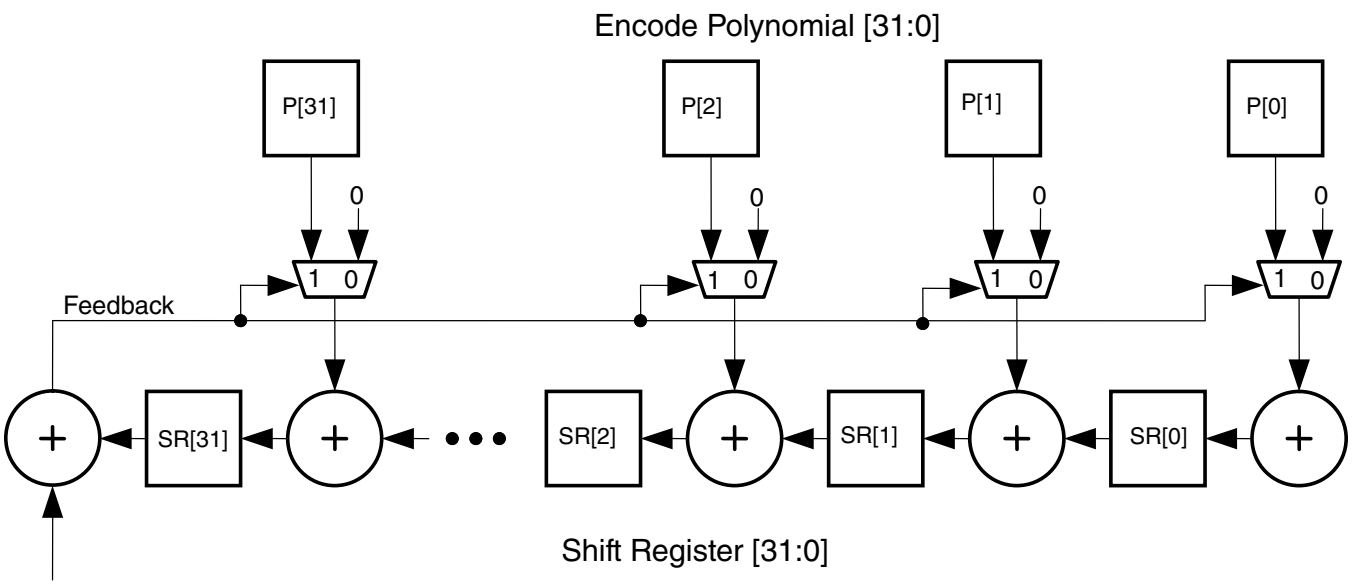
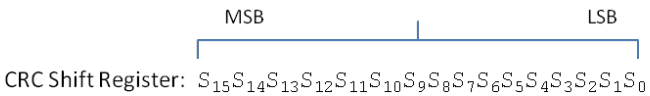



**Figure 64-124. CRC Calculation Shift Register**

Data flows from the link layer function to the CRCW module during transmit before being sent to the radio physical layer. All bits in each packet are provided by the link layer function, including placeholder bits for the CRC value. The link layer function marks the CRC calculation start and stop points in the bit stream. The link layer also indicates the start of the CRC placeholder bits. The CRCW module overwrites the placeholder CRC bits with the CRC result, ordered as specified by the configuration options. Figure 64-125 is a list of example 16-bit CRC output modes.




| REF IN | BYTE ORDER | REF OUT | Order of Transmitted Bits (shifted out first ..... shifted out last) |
|---|---|---|---|
| X | 0 | 0 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $S_{15}S_{14}S_{13}S_{12}S_{11}S_{10}S_9S_8S_7S_6S_5S_4S_3S_2S_1S_0$ |
| X | 0 | 1 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $S_8S_9S_{10}S_{11}S_{12}S_{13}S_{14}S_{15}S_0S_1S_2S_3S_4S_5S_6S_7$ |
| X | 1 | 0 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $S_7S_6S_5S_4S_3S_2S_1S_0S_{15}S_{14}S_{13}S_{12}S_{11}S_{10}S_9S_8$ |
| X | 1 | 1 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $S_0S_1S_2S_3S_4S_5S_6S_7S_8S_9S_{10}S_{11}S_{12}S_{13}S_{14}S_{15}$ |

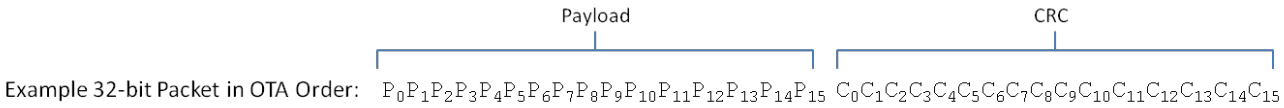
**Figure 64-125. CRC Output Bit Order**

### 64.3.9.1.2.2 CRC Check

During reception, data flows from the radio physical layer to the CRCW module before being sent to the link layer function. All bits in each packet are provided by the radio physical layer. Just as in transmit, the link layer function marks the CRC calculation start and stop points in the bit stream. The link layer also indicates the start of the received CRC bits.

The CRCW calculates the expected CRC result just as in transmit, but compares the received CRC bits as they arrive, ordered as specified by the configuration options, instead of inserting them into the packet. An early fail signal is asserted if a CRC bit miscompare is detected.

The CRC register will contain the expected CRC when either of the *config_crc_byte_order* or *config_crc_ref_in* CRC configuration options are set. The CRC register will contain the syndrome of the CRC error calculation when the both are not set.




| REF IN | BYTE ORDER | REF OUT | Order of Bits Used in CRC Calculation (shifted in first ..... shifted in last) |
|---|---|---|---|
| 0 | 0 | 0 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $C_0C_1C_2C_3C_4C_5C_6C_7C_8C_9C_{10}C_{11}C_{12}C_{13}C_{14}C_{15}$ |
| 0 | 0 | 1 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $C_7C_6C_5C_4C_3C_2C_1C_0C_{15}C_{14}C_{13}C_{12}C_{11}C_{10}C_9C_8$ |
| 0 | 1 | 0 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $C_8C_9C_{10}C_{11}C_{12}C_{13}C_{14}C_{15}C_0C_1C_2C_3C_4C_5C_6C_7$ |
| 0 | 1 | 1 | $P_0P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}$ $C_{15}C_{14}C_{13}C_{12}C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1C_0$ |
| 1 | 0 | 0 | $P_7P_6P_5P_4P_3P_2P_1P_0P_{15}P_{14}P_{13}P_{12}P_{11}P_{10}P_9P_8$ $C_0C_1C_2C_3C_4C_5C_6C_7C_8C_9C_{10}C_{11}C_{12}C_{13}C_{14}C_{15}$ |
| 1 | 0 | 1 | $P_7P_6P_5P_4P_3P_2P_1P_0P_{15}P_{14}P_{13}P_{12}P_{11}P_{10}P_9P_8$ $C_7C_6C_5C_4C_3C_2C_1C_0C_{15}C_{14}C_{13}C_{12}C_{11}C_{10}C_9C_8$ |
| 1 | 1 | 0 | $P_7P_6P_5P_4P_3P_2P_1P_0P_{15}P_{14}P_{13}P_{12}P_{11}P_{10}P_9P_8$ $C_8C_9C_{10}C_{11}C_{12}C_{13}C_{14}C_{15}C_0C_1C_2C_3C_4C_5C_6C_7$ |
| 1 | 1 | 1 | $P_7P_6P_5P_4P_3P_2P_1P_0P_{15}P_{14}P_{13}P_{12}P_{11}P_{10}P_9P_8$ $C_{15}C_{14}C_{13}C_{12}C_{11}C_{10}C_9C_8C_7C_6C_5C_4C_3C_2C_1C_0$ |

**Figure 64-126. CRC Input Bit Order**

### 64.3.9.1.2.3 Data Whitening

The data whitener behaves identically in both transmit (whiten) and receive (de-whiten) operating modes - there is no distinction. The LFSR used in this operation is designed to use either Fibonacci or Galois types of calculations. In either case, the LSB of the shift register is used to whiten each data bit as they are streamed through the CRCW.

An initialization signal for the data whiten logic loads the shift register with the seed value. The initialization signal can be used to re-initialize the shift register during a packet. A protocol specific state machine controls a signal to mark the start and stop points for data whitening in the data stream.

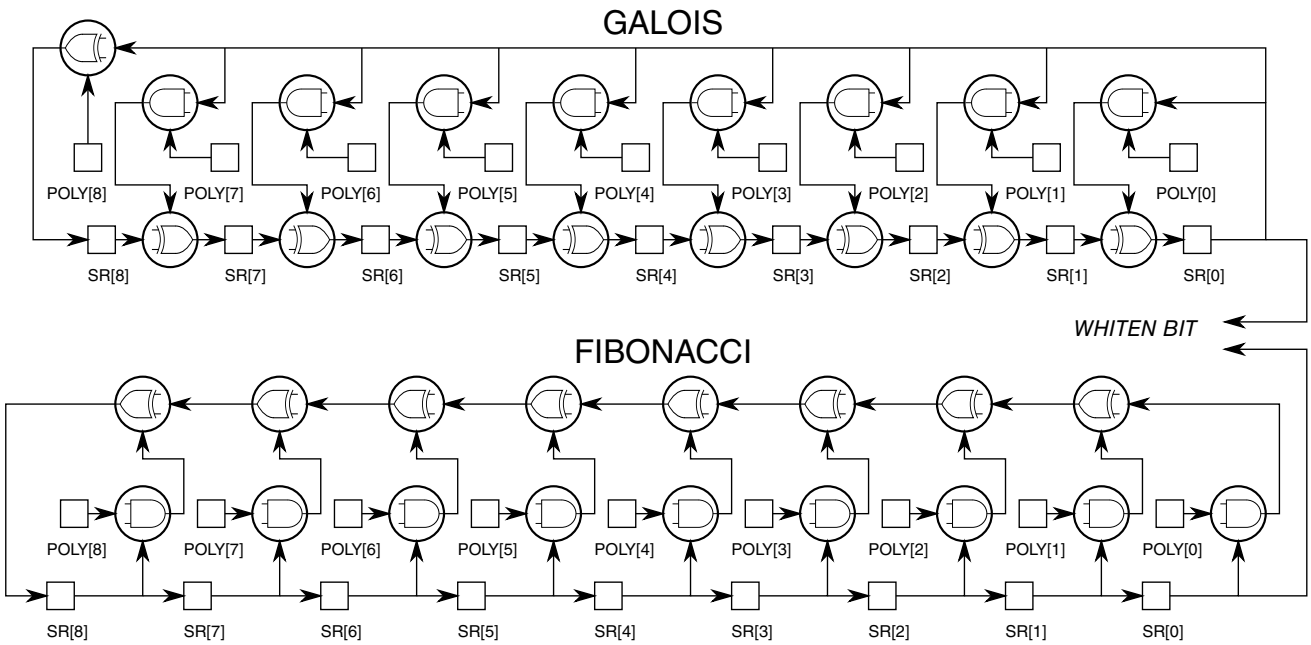



**Figure 64-127. Data Whitening Shift Register**

### 64.3.9.1.2.4 Error Correction

The CRC engine can optionally provide a level of error correction. This is accomplished by using the syndrome to locate the position of a possible burst error. This is especially useful for short burst errors. The error correction feature is controlled by configuration bits. A configuration setting defines the length of a short packet from zero to 31 bytes, and two different settings control the aggressiveness of error correction for short and long packets. More aggressive values attempt to correct longer burst errors, but potentially result in more false error correction solutions. Error correciton can be enabled for short packets and disasbled for long packets, or vice versa. The error correction feature is completely disabled when the CRC value is not received MSb first (*config_crc_byte_order* or *config_crc_ref_in*are enabled) because the error syndrome is not calculated.

While correcting random errors becomes difficult very quickly, it is fairly easy to correct a small burst error. Once the syndrome has been computed, all effects of the random message bits have been removed and the syndrome is the CRC value associated with the error signal. Reversing the CRC decoding process with the syndrome as the initial value of the LFSR will predict where the errors occurred in the payload. The structure of the CRC error correction LFSR is shown in Figure 64-128. The shifting process looks for a range of zeros in the least significant bits of the LFSR, counting the number of shifting operatations. The number of shifting operations determines the error location from the end of the packet accross all bits used in the CRC calculation.

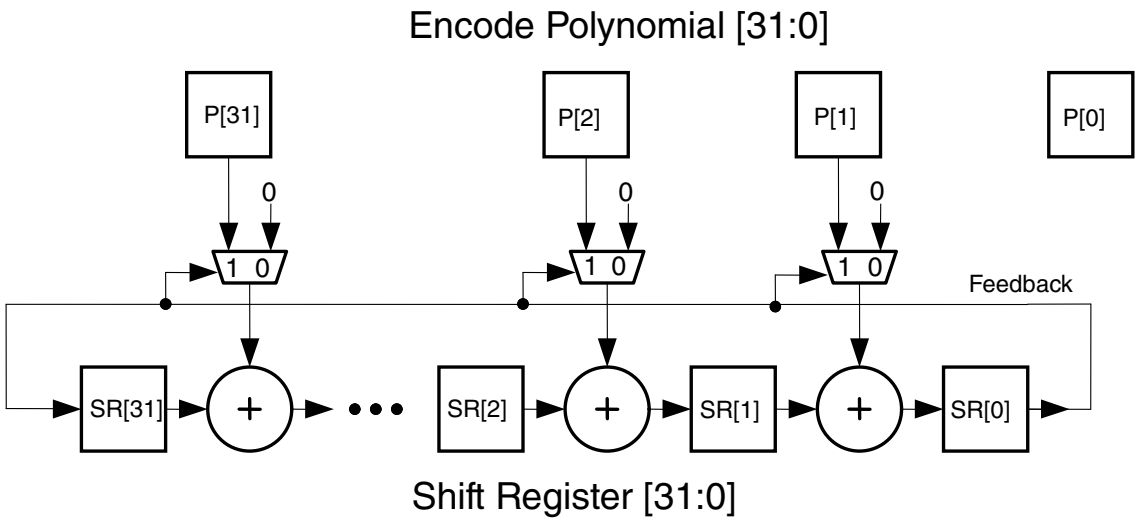



**Figure 64-128. CRC Correction Shift Register**

The CRCW module will perform the algorithm described above if enabled and the results of the CRC receive decode operation results in a syndrome that indicates an error. The logic performes one iteration per system clock cycle and can take up to *crc-included-bits* x *system clock cycles* to complete. The result of the error correction algorythm is an offset and XOR mask when successful. The offset is in bytes and counts only those bytes that are used in the CRC calculation, including the CRC value. The offset and mask values are zero on failure. The CRCW module will report if the algorythm was successful and the burst error position and mask so that upstream software can perform the correction.

### 64.3.9.2 Interface Timing

The CRCW interface timing requirements are described below. All signals except *reset_b* should be driven synchronous to *clk*. Some configuration options required a minumum number of clocks between data input bits or between the initialize input signals and the first data input bit.

The maximum input bitrate is 1/10$^{th}$ the system clock rate as shown in Figure 64-129.

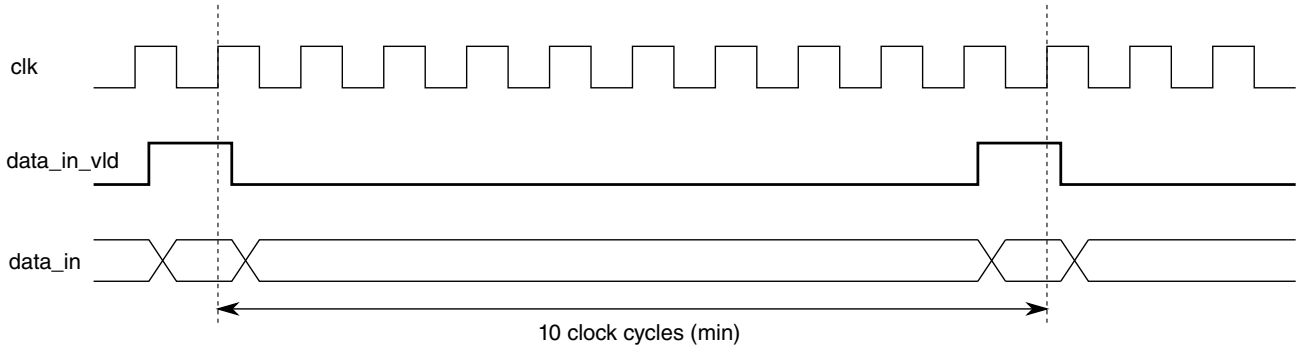



**Figure 64-129. Maximum Bitrate Timing Diagram**

## 64.3.9.2.1 Initialization

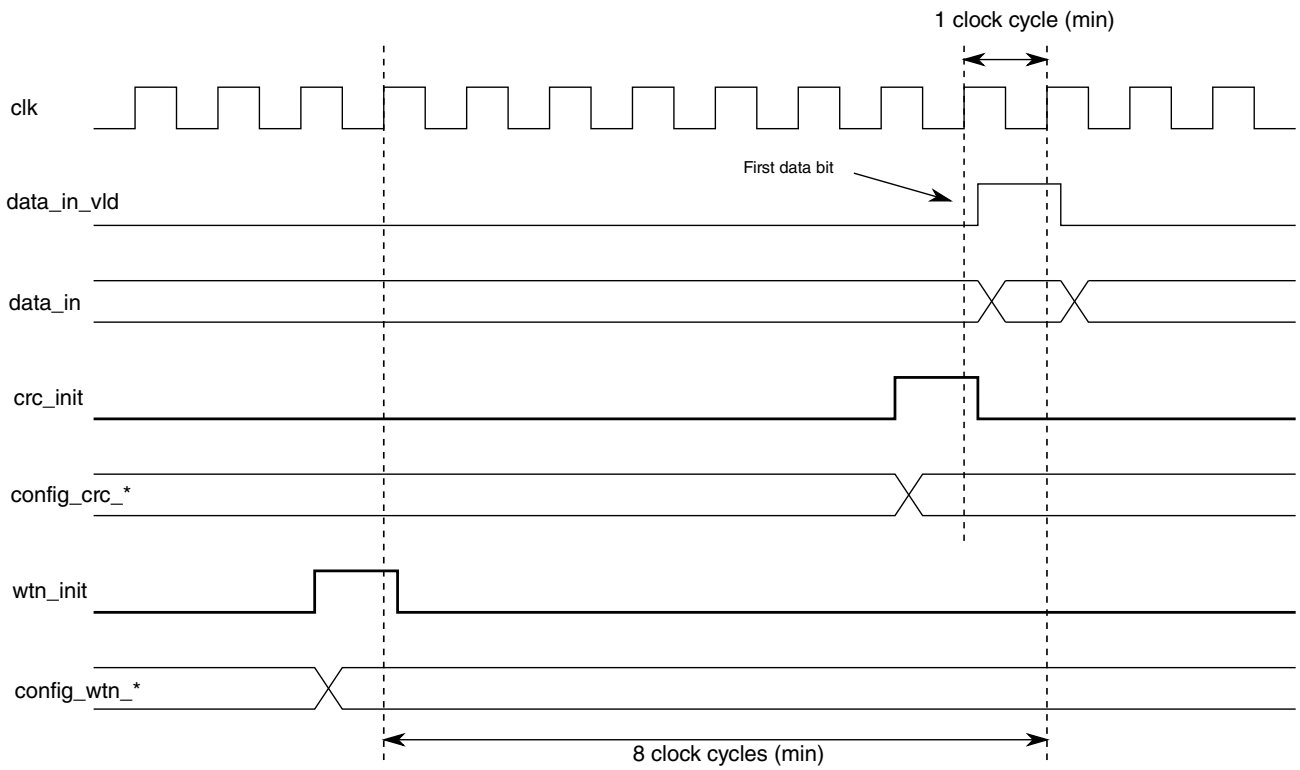



**Figure 64-130. Initialization Timing Diagram**

### 64.3.9.2.2 Whiten Re-Initialization

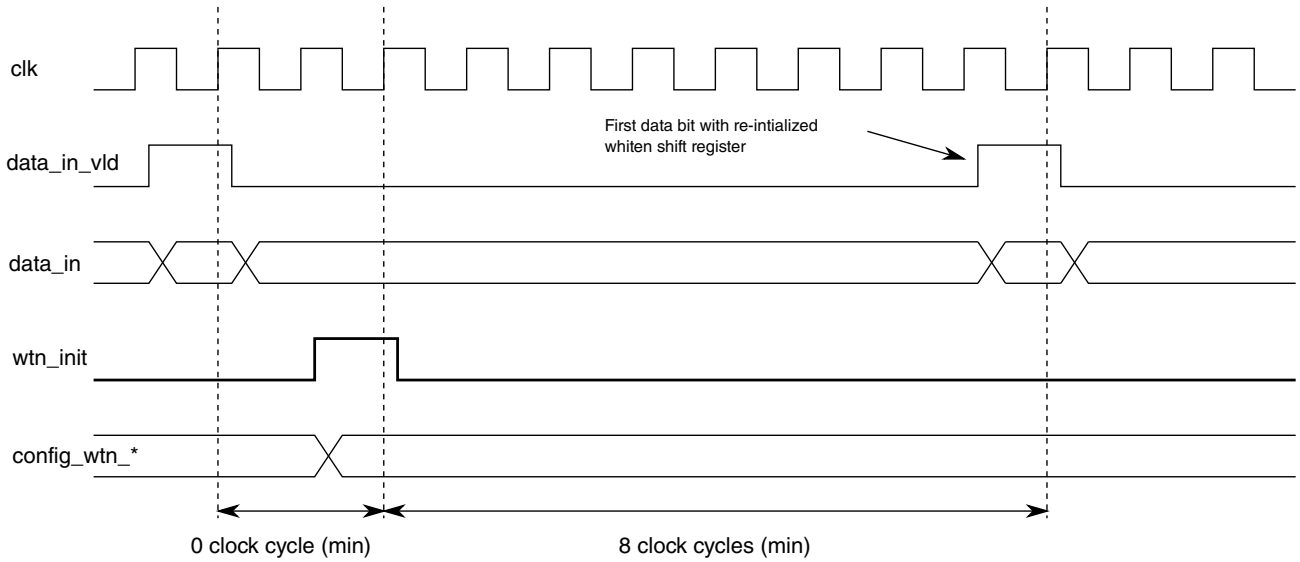



**Figure 64-131. Whiten Re-Initialization Timing Diagram**

### 64.3.9.2.3 CRC/WTN Enable

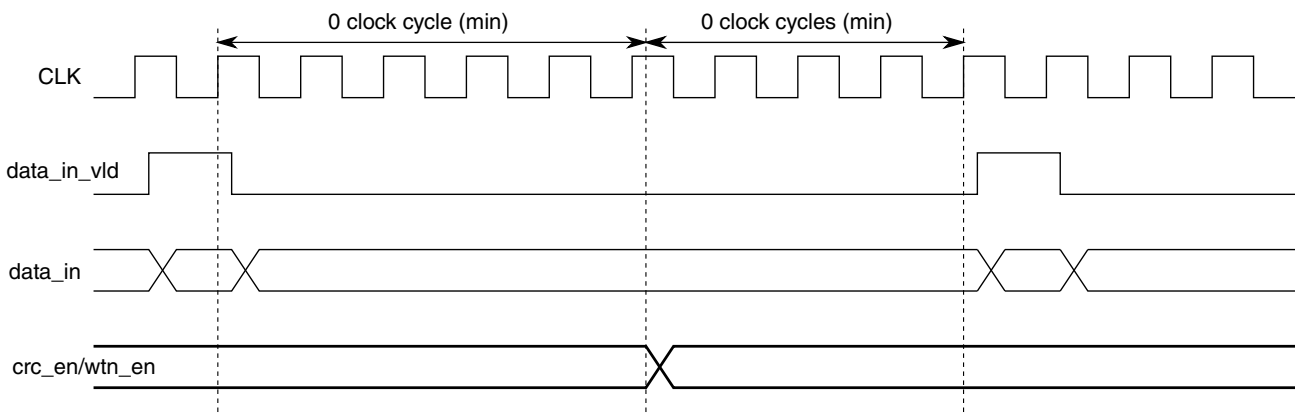



**Figure 64-132. CRC/WTN Enable Timing Diagram**

### 64.3.9.2.4 Results

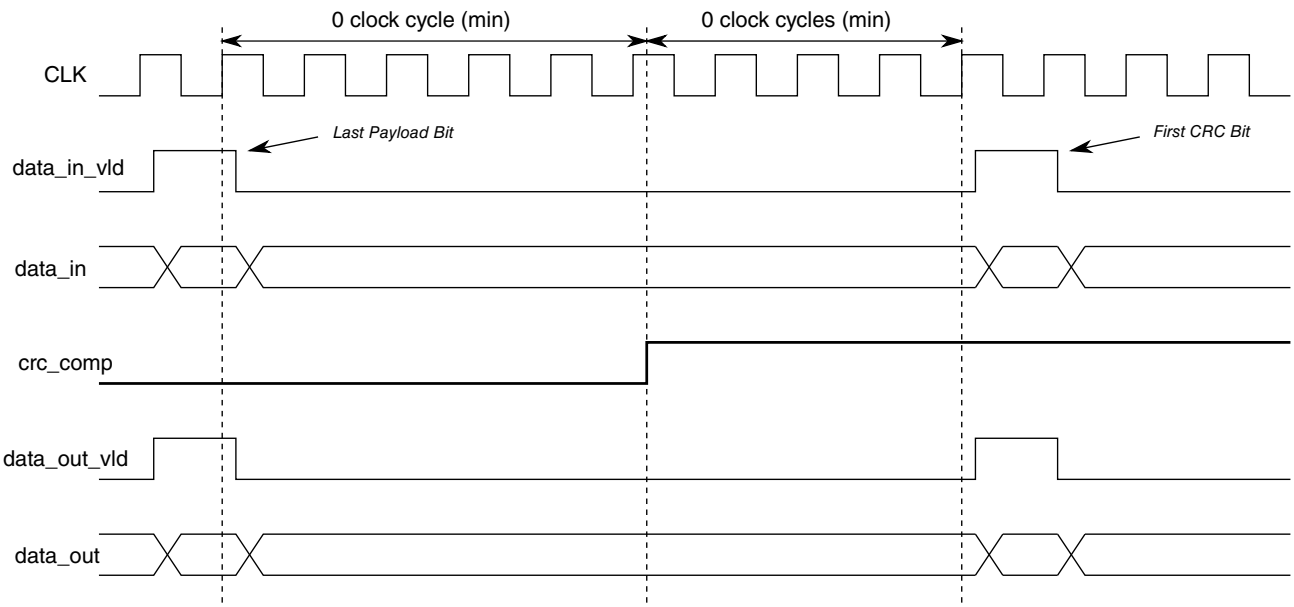



**Figure 64-133. CRC Complete Timing Diagram**

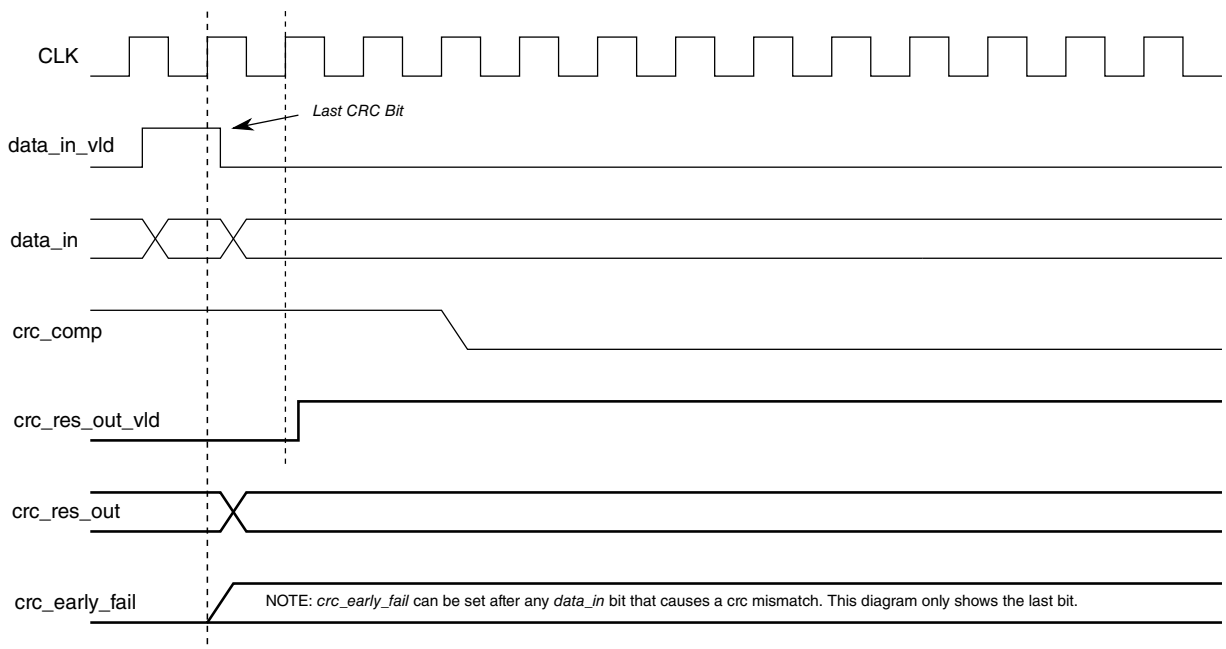



**Figure 64-134. CRC Result Out Valid Timing Diagram**

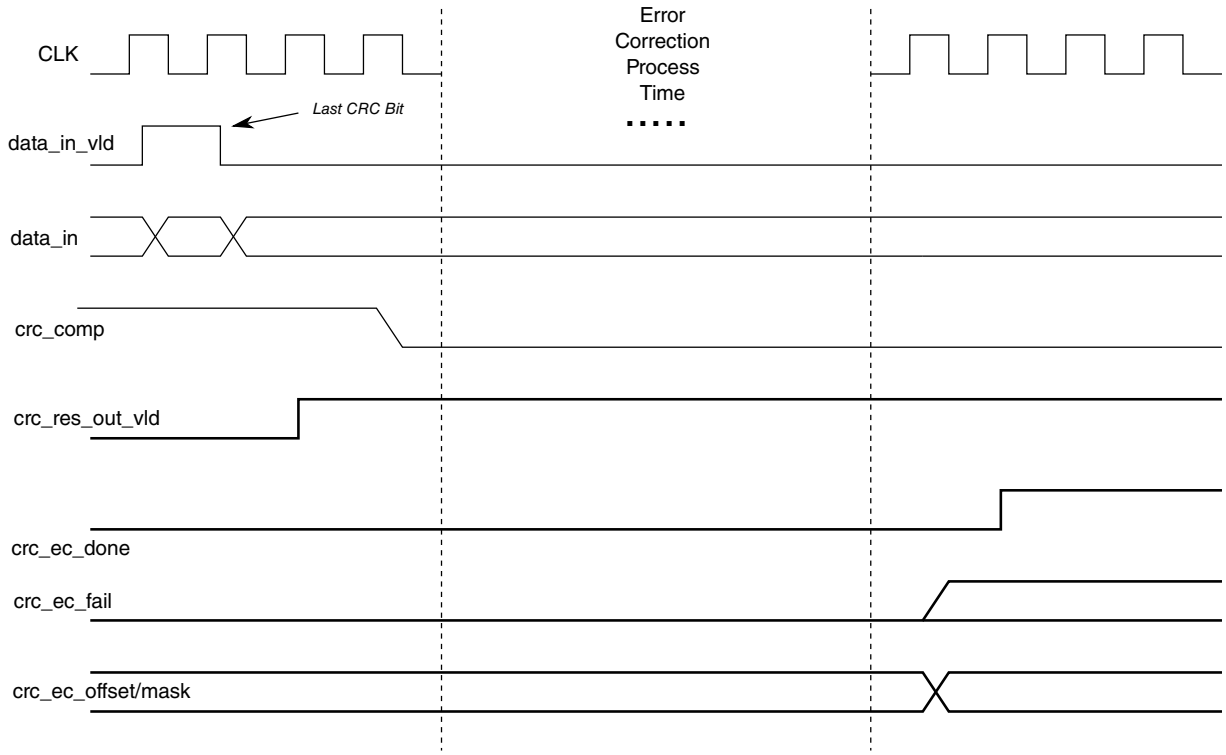



**Figure 64-135. CRC Error Correction Done Timing Diagram**

## 64.3.10 Other Transceiver Memory Map and Register Definition

### 64.3.10.1 XCVR_MISC register descriptions

#### 64.3.10.1.1 XCVR_CTRL_ADDR Memory map

XCVR_MISC base address: 4103_0280h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TRANSCEIVER CONTROL (XCVR_CTRL) | 32 | RW | 0010_1000h |
| 4h | TRANSCEIVER STATUS (XCVR_STATUS) | 32 | W1C | See description. |
| 8h | BLE ARBITRATION CONTROL (BLE_ARB_CTRL) | 32 | RW | 0000_0000h |
| Ch | OVERWRITE VERSION (OVERWRITE_VER) | 32 | RW | 0000_0000h |
| 10h | DIGITAL TEST MUX CONTROL (DTEST_CTRL) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

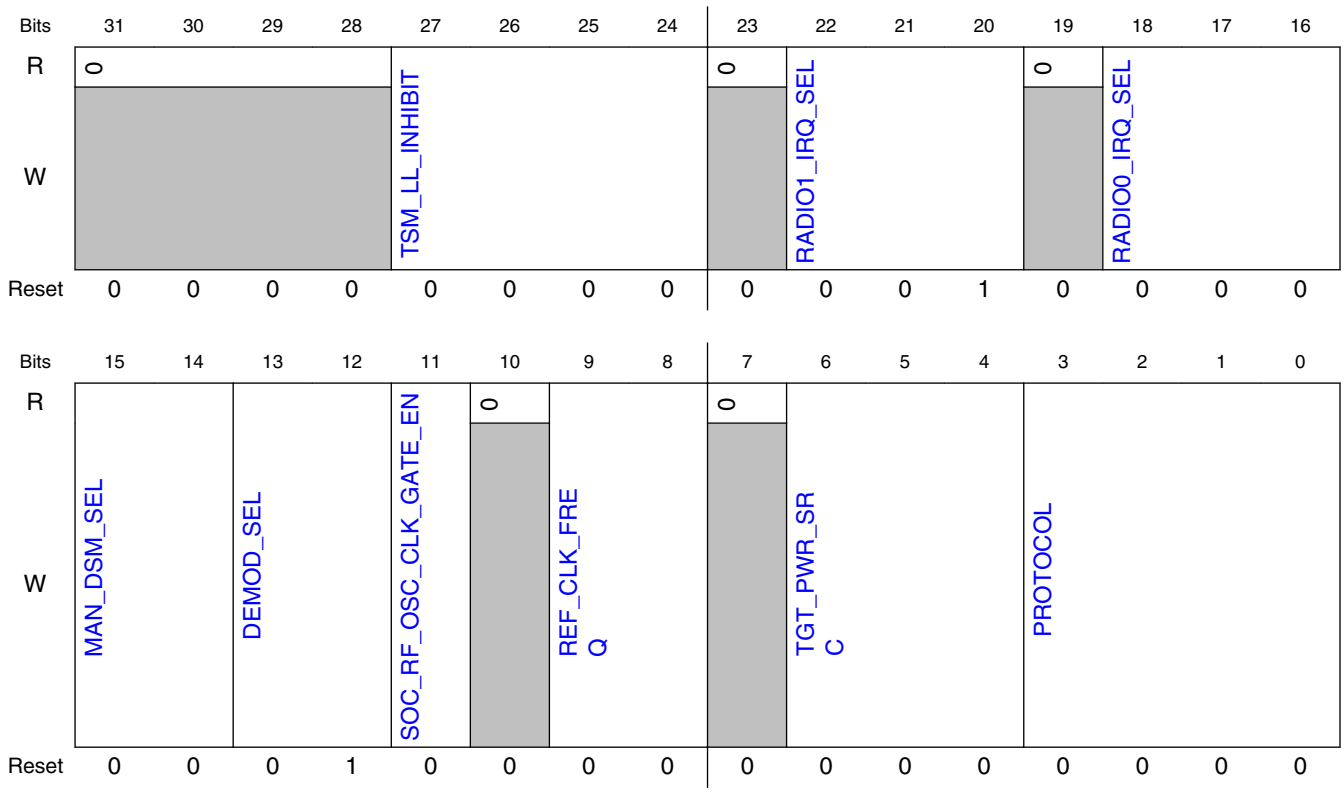
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 14h | TRANSCEIVER DMA CONTROL (DMA_CTRL) | 32 | RW | 0000_0300h |
| 18h | TRANSCEIVER DMA DATA (DMA_DATA) | 32 | RO | 0000_0000h |
| 1Ch | PACKET RAM CONTROL (PACKET_RAM_CTRL) | 32 | RW | 0000_0000h |
| 20h | PACKET RAM DEBUG RAM STOP ADDRESS (RAM_STOP_ADDR) | 32 | RO | 0000_0000h |
| 24h | FAD CONTROL (FAD_CTRL) | 32 | RW | 0000_F080h |
| 28h | LOW POWER PREAMBLE SEARCH CONTROL (LPPS_CTRL) | 32 | RW | 6426_0000h |
| 2Ch | COEXISTENCE CONTROL (COEX_CTRL) | 32 | RW | 0000_0000h |
| 30h | CRC/WHITENER CONFIG REGISTER (CRCW_CFG) | 32 | RW | 0000_0000h |
| 34h | CRC ERROR CORRECTION MASK (CRC_EC_MASK) | 32 | RO | 0000_0000h |
| 38h | CRC RESULT (CRC_RES_OUT) | 32 | RO | 0000_0000h |
| 3Ch | CRC/WHITENER CONFIG 2 REGISTER (CRCW_CFG2) | 32 | RW | 0000_0000h |

## 64.3.10.1.2 TRANSCEIVER CONTROL (XCVR_CTRL)

### 64.3.10.1.2.1 Offset

| Register | Offset |
|---|---|
| XCVR_CTRL | 0h |

### 64.3.10.1.2.2 Diagram




### 64.3.10.1.2.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>TSM_LL_INHIBIT | TSM Per-Link-Layer Inhibit<br>Nominally, any protocol engine can request TSM to start a TX or RX sequence at any time. In a multiprotocol setting, individual protocol engines can (optionally) be selectively blocked from accessing TSM by setting one of the following bits of TSM_LL_INHIBIT[3:0]:<br>**xxx1**: BLE commands to TSM are inhibited<br>**xxx0**: BLE commands to TSM are permitted<br>**xx1x**: 802.15.4 commands to TSM are inhibited<br>**xx0x**: 802.15.4 commands to TSM are permitted<br>**1xxx**: GENERIC_FSK commands to TSM are inhibited<br>**0xxx**: GENERIC_FSK commands to TSM are permitted |
| 23<br>— | Reserved |
| 22-20 | RADIO1_IRQ_SEL<br>Assigns Radio #1 Interrupt (ipi_int_radio1) to a Protocol Engine |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RADIO1_IRQ_SEL | 000b - Assign Radio #1 Interrupt to BLE<br>001b - Assign Radio #1 Interrupt to 802.15.4<br>010b - Radio #1 Interrupt unassigned<br>011b - Assign Radio #1 Interrupt to GENERIC_FSK<br>100b - Radio #1 Interrupt unassigned<br>101b - Radio #1 Interrupt unassigned<br>110b - Radio #1 Interrupt unassigned<br>111b - Radio #1 Interrupt unassigned |
| 19<br>— | Reserved |
| 18-16<br>RADIO0_IRQ_SEL | RADIO0_IRQ_SEL<br>Assigns Radio #0 Interrupt (ipi_int_radio0) to a Protocol Engine<br>000b - Assign Radio #0 Interrupt to BLE<br>001b - Assign Radio #0 Interrupt to 802.15.4<br>010b - Radio #0 Interrupt unassigned<br>011b - Assign Radio #0 Interrupt to GENERIC_FSK<br>100b - Radio #0 Interrupt unassigned<br>101b - Radio #0 Interrupt unassigned<br>110b - Radio #0 Interrupt unassigned<br>111b - Radio #0 Interrupt unassigned |
| 15-14<br>MAN_DSM_SEL | Manual DSM Selector<br>Assigning Manual DSM to a protocol is done by way of the MAN_DSM_SEL[1:0] register, as shown in the following table:<br>0: Manual DSM Not Assigned<br>1: Manual DSM Assigned to 802.15.4<br>2: Manual DSM Assigned to GENERIC_FSK |
| 13-12<br>DEMOD_SEL | Demodulator Selector<br>This bit selects the demodulator used during reception<br>00b - No demodulator selected<br>01b - Use NXP Multi-standard PHY demodulator<br>10b - Use Legacy 802.15.4 demodulator<br>11b - Reserved |
| 11<br>SOC_RF_OSC_CLK_GATE_EN | SOC_RF_OSC_CLK_GATE_EN<br>Enable 3V version of RF OSC Clock for use by the SoC |
| 10<br>— | Reserved |
| 9-8<br>REF_CLK_FREQ | Radio Reference Clock Frequency<br>This register selects the Reference Clock Frequency for the Radio.<br>00b - 32 MHz<br>01b - 26 MHz<br>10b - Reserved<br>11b - Reserved |
| 7<br>— | Reserved |
| 6-4<br>TGT_PWR_SRC | Target Power Source<br>For determining transmit power, the TGT_PWR_SRC[2:0] bits control target power selection, according to the following table. |

*Table continues on the next page...*

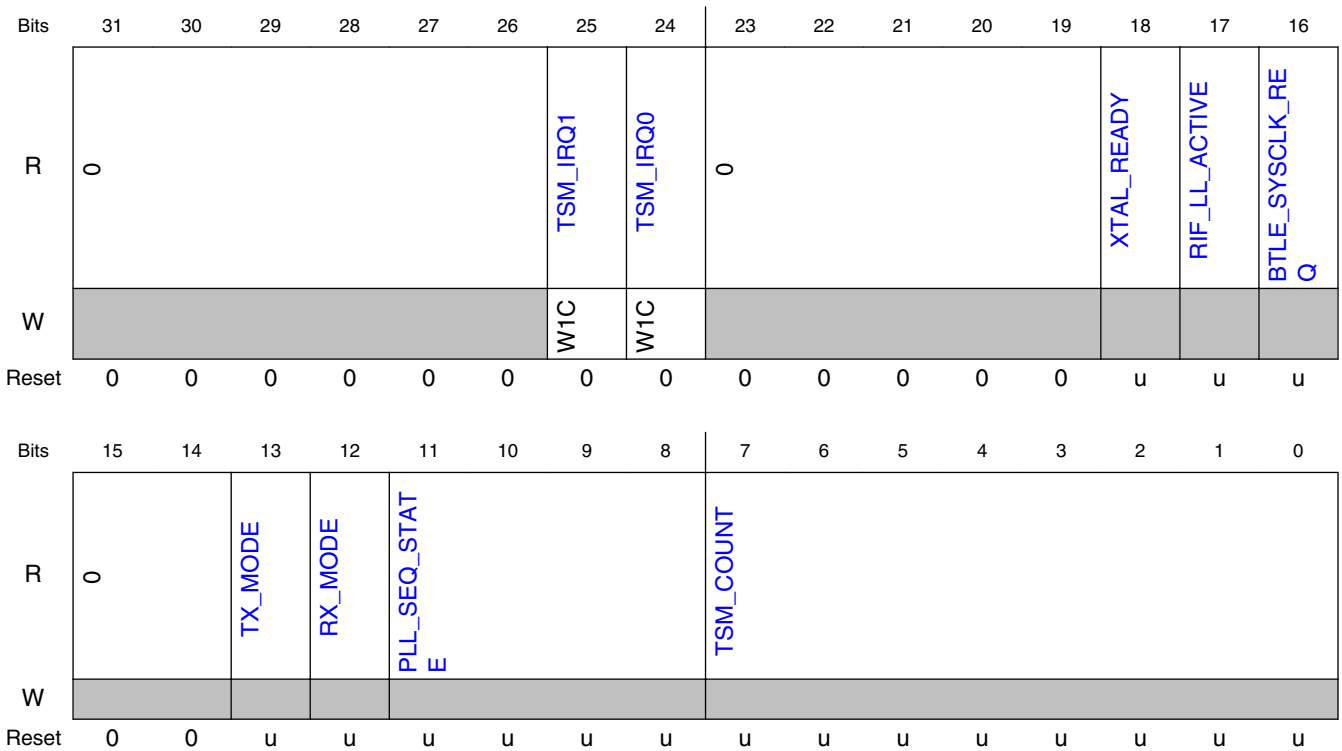
| Field | Function |
|---|---|
| | 0: TARGET POWER SOURCE is PA_POWER[5:0] register (XCVR space)<br>1: TARGET POWER SOURCE is BTLE Link Layer<br>2: TARGET POWER SOURCE is 802.15.4 Link Layer (PA_PWR[5:0] register in 802.15.4 space)<br>3: Reserved<br>4: TARGET POWER SOURCE is GENERIC_FSK Link Layer (TX_POWER[5:0] register in GENERIC_FSK space)<br>5: Reserved<br>6: Reserved<br>7: TARGET POWER SOURCE is determined by the PROTOCOL[3:0] bits |
| 3-0<br>PROTOCOL | Radio Protocol Selection<br>This register selects the Radio Communication Protocol.<br>0000b - BLE<br>0001b - BLE in MBAN<br>0010b - BLE overlap MBAN<br>0011b - Reserved<br>0100b - 802.15.4<br>0101b - 802.15.4j<br>0110b - Radio Channels 0-127 selectable, FSK<br>0111b - Radio Channels 0-127 selectable, GFSK<br>1000b - Generic GFSK, with Gaussian Filter<br>1001b - Generic MSK, O-QPSK encoding<br>1010b - Generic FSK, direct +/- Fdev FSK |

## 64.3.10.1.3 TRANSCEIVER STATUS (XCVR_STATUS)

### 64.3.10.1.3.1 Offset

| Register | Offset |
|---|---|
| XCVR_STATUS | 4h |

## 64.3.10.1.3.2 Diagram




## 64.3.10.1.3.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>TSM_IRQ1 | TSM Interrupt #1<br>0b - TSM Interrupt #1 is not asserted.<br>1b - TSM Interrupt #1 is asserted. Write '1' to this bit to clear it. |
| 24<br>TSM_IRQ0 | TSM Interrupt #0<br>0b - TSM Interrupt #0 is not asserted.<br>1b - TSM Interrupt #0 is asserted. Write '1' to this bit to clear it. |
| 23-19<br>— | Reserved |
| 18<br>XTAL_READY | RF Osciallator Xtal Ready<br>Oscillator warmup count complete.<br>0b - Indicates that the RF Oscillator is disabled or has not completed its warmup.<br>1b - Indicates that the RF Oscillator has completed its warmup count and is ready for use. |
| 17<br>RIF_LL_ACTIVE | Link Layer Active Indication<br>Reflects the state of the BTLE “Link Layer Active” status bit. RIF_LL_ACTIVE is to be used by the host as an 'early' indication to prevent host to do any operations while BTLE IP is doing transceiver operations, so as to reduce the peak power and noise. |
| 16 | BTLE System Clock Request |

*Table continues on the next page...*

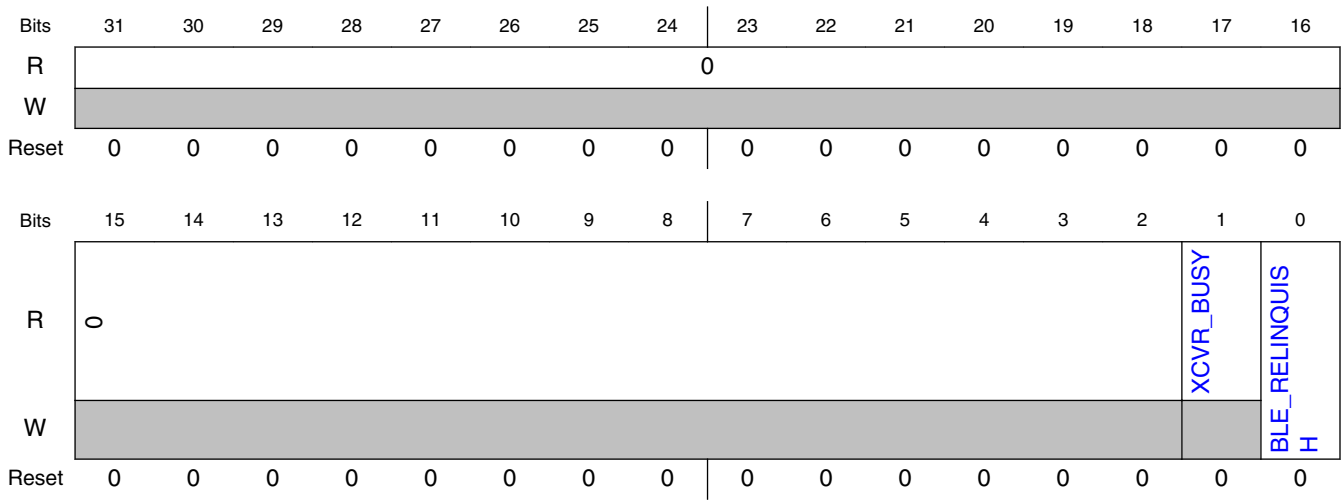
| Field | Function |
|---|---|
| BTLE_SYSCLK_REQ | Reflects the state of the BTLE oscillator request signal. BTLE_SYSCLK_REQ is the BTLE control for the RF Oscillator. BTLE will deassert this signal upon entering DSM (deep sleep mode) to request oscillator turn-off, and will re-assert it prior to exiting DSM. The turn-on leadtime on this signal for exiting DSM, is programmable with the BTLE block. This read-only bit can thus be queried to ascertain the power-state of BTLE. |
| 15-14<br>— | Reserved |
| 13<br>TX_MODE | Transmit Mode<br>Indicates an TX transceiver operation is in progress. |
| 12<br>RX_MODE | Receive Mode<br>Indicates an RX transceiver operation is in progress. |
| 11-8<br>PLL_SEQ_STATE | PLL Sequence State<br>Reflects the state of the PLL digital state machine.<br>0000b - PLL OFF<br>0010b - CTUNE<br>0011b - CTUNE_SETTLE<br>0110b - HPMCAL1<br>1000b - HPMCAL1_SETTLE<br>1010b - HPMCAL2<br>1100b - HPMCAL2_SETTLE<br>1111b - PLLREADY |
| 7-0<br>TSM_COUNT | TSM_COUNT<br>Reflects the instantaneous value of the TSM counter. |

## 64.3.10.1.4 BLE ARBITRATION CONTROL (BLE_ARB_CTRL)

### 64.3.10.1.4.1 Offset

| Register | Offset |
|---|---|
| BLE_ARB_CTRL | 8h |

#### 64.3.10.1.4.2 Diagram




#### 64.3.10.1.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>XCVR_BUSY | Transceiver Busy Status Bit<br>0b - RF Channel in available (TSM is idle)<br>1b - RF Channel in use (TSM is busy) |
| 0<br>BLE_RELINQUISH | BLE Relinquish Control<br>This bit forces the BLE protocol engine to immediately relinquish access to the RF Channel, in favor of another protocol with higher arbitration priority. BLE is denied RF channel access by blocking BLE access to the TSM; when this bit is subsequently cleared, BLE access is restored. |

### 64.3.10.1.5 OVERWRITE VERSION (OVERWRITE_VER)

#### 64.3.10.1.5.1 Offset

| Register | Offset |
|---|---|
| OVERWRITE_VER | Ch |

#### 64.3.10.1.5.2 Function

The Overwrite Version allows software to store a version number of trim and calibration values which are used to overwrite the chip default values in the registers. Typically,

software would perform the overwrite of the defaults in transceiver registers and then write the version number from the file containing the overwrite values into this register.

**NOTE**

This register has no hardware connections, it is simply a designated storage location for a version number.

#### 64.3.10.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | OVERWRITE_VER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.3.10.1.5.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>OVERWRITE_VER | Overwrite Version Number.<br>Points to the version number of the overwrites.h file used to initialize the device; can be used by software to identify a version-controlled set of non-default values to be written into the transceiver's register map. |

### 64.3.10.1.6 DIGITAL TEST MUX CONTROL (DTEST_CTRL)

#### 64.3.10.1.6.1 Offset

| Register | Offset |
|---|---|
| DTEST_CTRL | 10h |

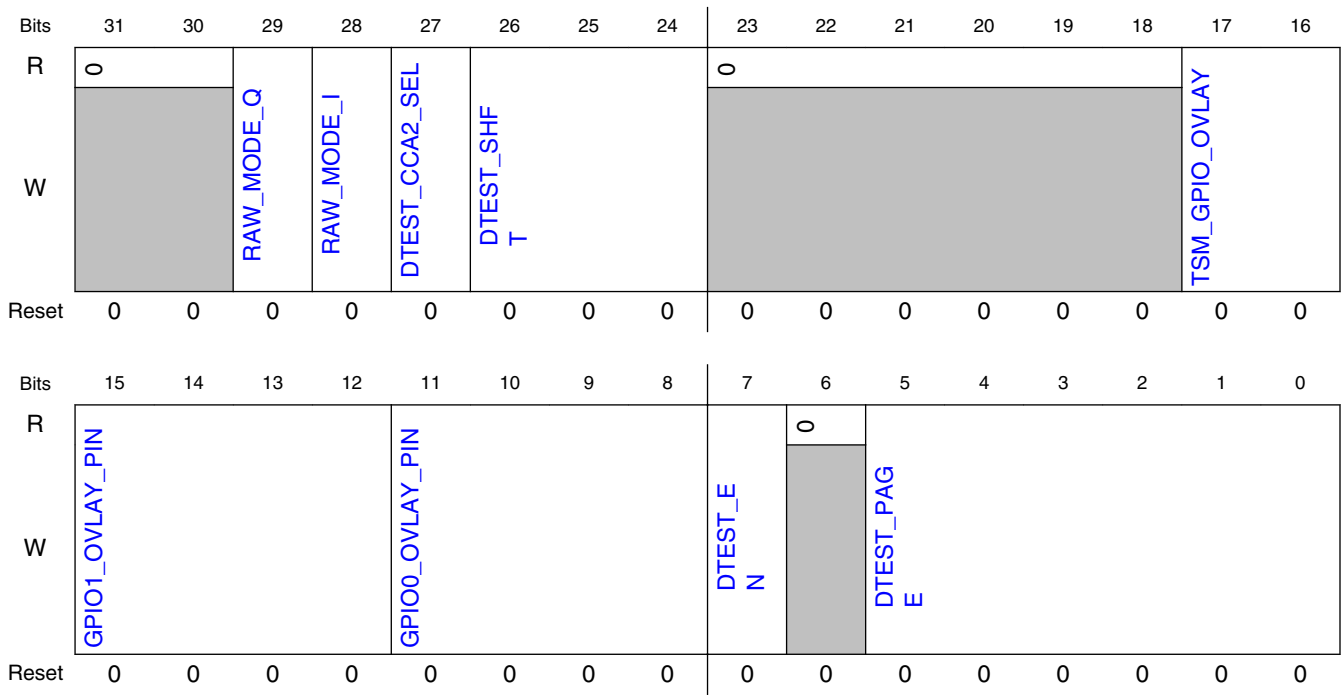
#### 64.3.10.1.6.2 Function

Digital Test Control. Allows selection and enablement of a page of DTEST signals to appear on the SoC DTEST pins.

**NOTE**

This register configures only the transceiver for DTEST mode; since DTEST pads on the SoC are multiplexed with other functions, SoC Port Pin programming is also required for each DTEST output

### 64.3.10.1.6.3 Diagram




### 64.3.10.1.6.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>RAW_MODE_Q | DTEST Raw Mode Enable for Q Channel<br>DTEST Q Channel Raw Mode allows raw, unfiltered ADC samples to be brought out to DTEST pins on the RXDIGIQ DTEST page. In raw mode, 2 5-bit ADC samples are concatenated into a single 12-bit DMA sample. DMA transfers these samples to memory in the same way it transfers filtered samples. Raw mode is only supported when the RX_DIG is programmed to decimate-by-2. The procedure to active Raw mode on Q channel is as follows:<br>1. RX_DIG_CTRL[RX_ADC_RAW_EN]=1<br>2. RX_DIG_CTRL[RX_DEC_FILT_OSR]=0<br>3. DTEST_CTRL[DTEST_PAGE]=0x0E (RXDIGIQ)<br>4. DTEST_CTRL[DTEST_EN]=1<br>5. DTEST_CTRL[RAW_MODE_Q]=1 |
| 28 | DTEST Raw Mode Enable for I Channel |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RAW_MODE_I | DTEST I Channel Raw Mode allows raw, unfiltered ADC samples to be brought out to DTEST pins on the RXDIGIQ DTEST page. In raw mode, 2 5-bit ADC samples are concatenated into a single 12-bit DMA sample. DMA transfers these samples to memory in the same way it transfers filtered samples. Raw mode is only supported when the RX_DIG is programmed to decimate-by-2. The procedure to active Raw mode on I channel is as follows:<br>1. RX_DIG_CTRL[RX_ADC_RAW_EN]=1<br>2. RX_DIG_CTRL[RX_DEC_FILT_OSR]=0<br>3. DTEST_CTRL[DTEST_PAGE]=0x0E (RXDIGIQ)<br>4. DTEST_CTRL[DTEST_EN]=1<br>5. DTEST_CTRL[RAW_MODE_I]=1 |
| 27<br>DTEST_CCA2_SEL | DTEST CCA Mode 2 Selector<br>This bit selects which internal node from the CCA2 module, appears on the DTEST ZBCCA2 page in the field cca2_max_or_sym[7:0], according to the following table:<br>0b - cca2_max_or_sym[7:0] = cca2_cnt_sym[7:0]<br>1b - cca2_max_or_sym[7:0] = cca2_cnt_max[7:0] |
| 26-24<br>DTEST_SHFT | DTEST Shift Control<br>This register field DTEST_SHFT[1:0] control the amount of "arithmetic shift", which can optionally be applied to DTEST output busses. DTEST_SHFT affects only 2 DTEST output busses:<br>PLL_RIPPLE_COUNTER[18:0] on DTEST page: PLLRIPPLE (0x02). Shift is to the left (magnitude increasing)<br>RX_DIG_IQ[11:0] on DTEST page: RXDIGIQ (0x0E). Shift is to the right (magnitude decreasing)<br>The bits of PLL_RIPPLE_COUNTER[18:0], an unsigned value, are shifted by DTEST_SHFT[2:0] according to the following table<br>0: PLL_RIPPLE_COUNTER[18:5] appear on DTEST[13:0]<br>1: PLL_RIPPLE_COUNTER[17:4] appear on DTEST[13:0]<br>2: PLL_RIPPLE_COUNTER[16:3] appear on DTEST[13:0]<br>3: PLL_RIPPLE_COUNTER[15:2] appear on DTEST[13:0]<br>4: PLL_RIPPLE_COUNTER[14:1] appear on DTEST[13:0]<br>5: PLL_RIPPLE_COUNTER[13:0] appear on DTEST[13:0]<br>6: Reserved<br>7: Reserved<br>The bits of RX_DIG_IQ[11:0], a signed value, are shifted/sign-extended by DTEST_SHFT[1:0] according to the following table:<br>0: These bits appear on DTEST[13:2] : RX_DIG_IQ[11:0] (no shift)<br>1: These bits appear on DTEST[13:2] : RX_DIG_IQ[11],RX_DIG_IQ[11:1] (right shift by 1)<br>2: These bits appear on DTEST[13:2] : RX_DIG_IQ[11],RX_DIG_IQ[11],RX_DIG_IQ[11:2] (right shift by 2)<br>3: These bits appear on DTEST[13:2] : RX_DIG_IQ[11:0] (no shift) |
| 23-18<br>— | Reserved |
| 17-16<br>TSM_GPIO_OVLAY | TSM GPIO Overlay Pin Control<br>The TSM-controlled output GPIO0_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. Similarly, the TSM-controlled output GPIO1_TRIG_EN can be routed to any DTEST pin, replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[0] = 1, the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[1] = 1, the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[0] = 0 and |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | TSM_GPIO_OVLAY[1] = 0, there is no overlay, and the DTEST Page Table dictates the node that appears on each DTEST pin.<br>00b - there is no overlay, and the DTEST Page Table dictates the node that appears on each DTEST pin.<br>01b - the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear.<br>10b - the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear.<br>11b - the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear, and the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear. |
| 15-12<br>GPIO1_OVLAY_PIN | GPIO 1 Overlay Pin<br>The TSM-controlled output GPIO1_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[1] = 1, this register selects the DTEST pin onto which GPIO1_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[1] = 0, this register is ignored, and the DTEST Page Table dictates the node that appears on each DTEST pin. |
| 11-8<br>GPIO0_OVLAY_PIN | GPIO 0 Overlay Pin<br>The TSM-controlled output GPIO0_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[0] = 1, this register selects the DTEST pin onto which GPIO0_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[0] = 0, this register is ignored, and the DTEST Page Table dictates the node that appears on each DTEST pin. |
| 7<br>DTEST_EN | DTEST Enable<br>DTEST enable<br>0b - Disables DTEST. The DTEST pins assume their mission function.<br>1b - Enables DTEST. The contents of the selected page (DTEST_PAGE) will appear on the DTEST output pins. |
| 6<br>— | Reserved |
| 5-0<br>DTEST_PAGE | DTEST Page Selector<br>DTEST Page signal assignments<br>000000b - PLLFREQCAL<br>000001b - PLLBESTDIFF<br>000010b - PLLRIPPLE<br>000011b - PLLHPMCAL<br>000100b - PLLVCOMOD<br>000101b - PLLUNLOCK<br>000110b - PLLCYCSLIP<br>000111b - PLLCHAN<br>001000b - TXWARMUP<br>001001b - TXPOWER<br>001010b - TXFREQWORD<br>001011b - RXWARMUP<br>001100b - RXADC<br>001101b - RXDMA<br>001110b - RXDIGIQ<br>001111b - RXDMA2<br>010000b - RXINPH<br>010001b - RSSI0<br>010010b - RSSI1<br>010011b - AGC0<br>010100b - AGC1 |

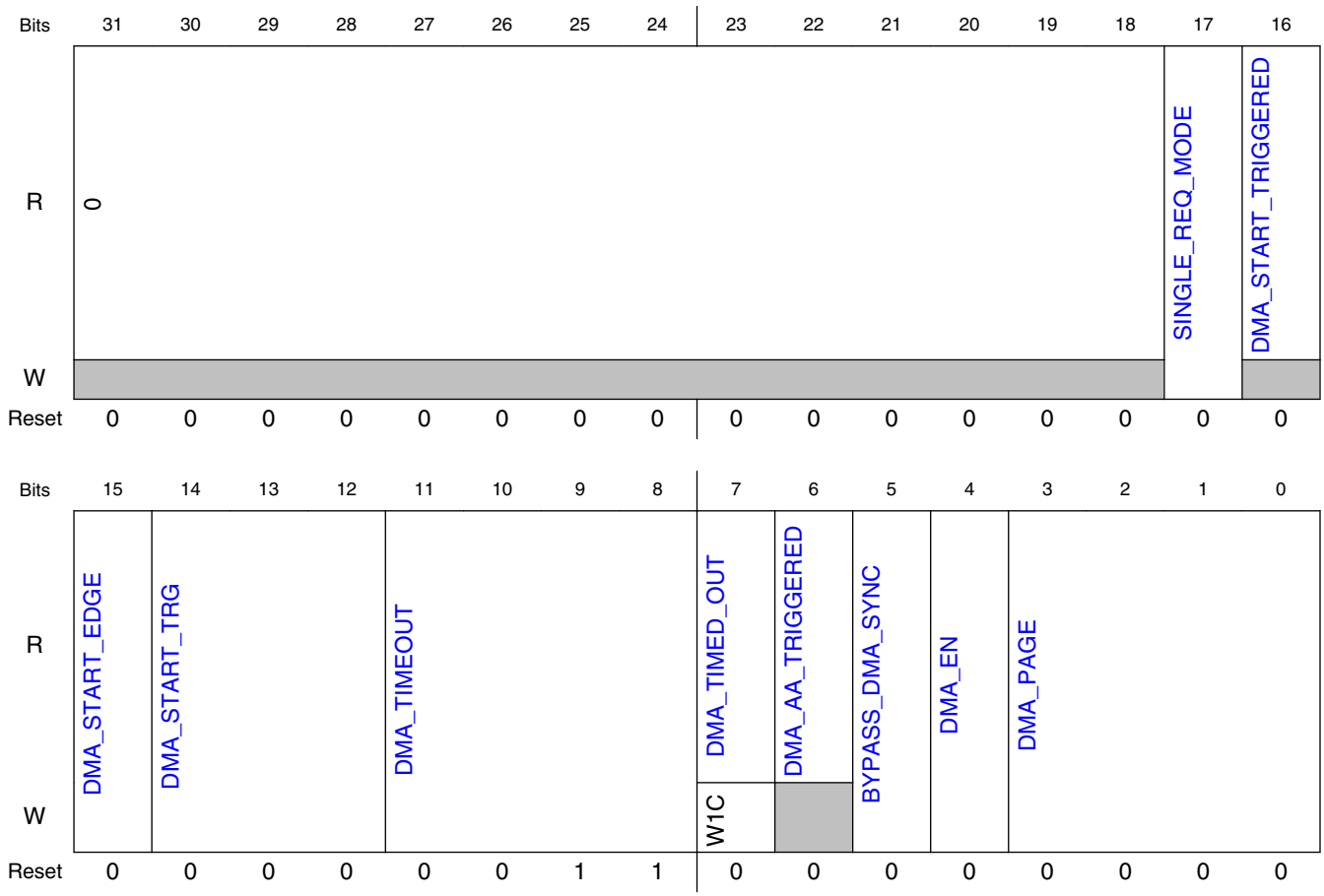
| Field | Function |
|---|---|
| | 010101b - DCOC0<br>010110b - DCOC1<br>010111b - DCOC2<br>011000b - DCOC3<br>011001b - TSM<br>011010b - MTTSMCAL<br>011011b - MTADV<br>011100b - MTINIT<br>011101b - MTSCAN<br>011110b - MTCONN<br>011111b - MTDTM<br>100000b - MTADVXCV<br>100001b - MTCONXCV<br>100010b - MTDTM2<br>100011b - DSM<br>100100b - PHY_FSK_STATE<br>100101b - PHY_CFO_EST_PD<br>100110b - PHY_CFO_EST_PD2<br>100111b - PHY_EARLY_LATE<br>101000b - PHY_FSK_DEMOD<br>101001b - PHY_AA_SEARCH<br>101010b - PHY_DATA_OUT<br>101011b - PHY_SAMP_TIME<br>101100b - CCA_ED_LQI<br>101101b - CCA_ED_LQI2<br>101110b - Reserved<br>101111b - Reserved<br>110000b - Reserved<br>110001b - Reserved<br>110010b - Reserved<br>110011b - Reserved<br>110100b - Reserved<br>110101b - Reserved<br>110110b - Reserved<br>110111b - Reserved<br>111000b - Reserved<br>111001b - Reserved<br>111010b - RCCAL<br>111011b - AUXPLLFCAL<br>111100b - GENFSKTX<br>111101b - GENFSKRX<br>111110b - GENFSKSTATE<br>111111b - GENFILTER |

## 64.3.10.1.7 TRANSCEIVER DMA CONTROL (DMA_CTRL)

### 64.3.10.1.7.1 Offset

| Register | Offset |
|---|---|
| DMA_CTRL | 14h |

### 64.3.10.1.7.2 Diagram




### 64.3.10.1.7.3 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17<br>SINGLE_REQ_MODE | DMA Single Request Mode<br>Configures the transceiver to transfer the entire block of DMA data with only a single initial request (ipd_req_radio_rx). The DMA2 controller must be configured accordingly. In this mode, the transceiver will use **ips_xfr_wait** to pace the individual transactions. Single Request Mode should not be used with DMA Pages 11, 12, or 13, because the data rate is too low and therefore ips_xfr_wait would remain asserted for excessively long periods.<br>0b - Disable Single Request Mode. The transceiver will assert ipd_req_radio_rx whenever it has a new sample ready for transfer.<br>1b - Enable Single Request Mode. A single initial request by the transceiver will transfer the entire DMA block of data |
| 16 | DMA Start Trigger Occurred |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DMA_START_TRIGGERED | This read-only status bit becomes set, when, during a DMA session (DMA_PAGE > 0), the trigger source selected by DMA_START_TRG[2:0] occurs, with the edge sensitivity selected by DMA_START_EDGE. To clear this bit, set DMA_PAGE=0. |
| 15<br>DMA_START_EDGE | DMA Start Trigger Edge Selector<br>DMA_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 14-12<br>DMA_START_TRG | DMA Start Trigger Selector<br>The DMA Start Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| 11-8<br>DMA_TIMEOUT | DMA Timeout<br>In DMA Single Request Mode, adverse consequences may result if the transceiver's ips_xfr_wait signal is asserted for an excessively long period of time, which could occur due to mis-programming of the DMA2 controller, and/or the transceiver itself. DMA Timeout forces the transceiver's ips_xfr_wait low after *N* microseconds and sets the **DMA_TIMED_OUT** status bit, where *N*=DMA_TIMEOUT. DMA_TIMEOUT is only relavent when SINGLE_REQ_MODE=1, otherwise the transceiver does not assert ips_xfr_wait. |
| 7<br>DMA_TIMED_OUT | DMA Transfer Timed Out<br>Status bit indicates that the transceiver DMA, while operating in Single Request Mode, asserted ips_xfr_wait for a period in excess of the programmed **DMA_TIMEOUT** setting, resulting in a timeout condition where the transceiver has forced ips_xfr_wait low. After a timeout, before resuming DMA operations, set DMA_PAGE=0 to disable DMA, and write 1 to this bit to clear it.<br>0b - A DMA timeout has not occurred<br>1b - A DMA timeout has occurred in Single Request Mode since the last time this bit was cleared |
| 6<br>DMA_AA_TRIGGERED | DMA Access Address triggered<br>This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of DMA transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DMA_PAGE to initiate DMA transfers. This bit is intended for use with DMA_PAGE=13, but is available on all pages. Its usage is optional. |
| 5<br>BYPASS_DMA_SYNC | Bypass External DMA Synchronization<br>When using transceiver DMA with SINGLE_REQ_MODE=0, synchronization external to the transceiver must be bypassed in order to minimize bus access latency. Using DMA in this mode also requires that the MCU and transceiver both are configured to operate in the RF Oscillator clock domain.<br>0b - Don't Bypass External Synchronization. Use this setting if SINGLE_REQ_MODE=1.<br>1b - Bypass External Synchronization. This setting is mandatory if SINGLE_REQ_MODE=0. |
| 4<br>DMA_EN | DMA Enable<br>Setting DMA_EN=1 enables clock gating to the Transciever DMA engine. This bit should be set prior to the start of a DMA session, and remain set for the duration of the session. |
| 3-0<br>DMA_PAGE | Transceiver DMA Page Selector<br>Selects the page of receiver data for storage to system memory when using Transceiver DMA Debug Mode. Setting this register to a non-zero value enables the DMA interface logic. Setting this register to zero disables the interface. **Note:** DMA_PAGE should be set to a non-zero value to begin an acquisition, |

| Field | Function |
|---|---|
| | only after the start trigger has been programmed (DMA_START_TRG), if a trigger is desired. The available DMA pages are listed below.<br>0000b - DMA Idle<br>0001b - RX_DIG I and Q<br>0010b - RX_DIG I Only<br>0011b - RX_DIG Q Only<br>0100b - RAW ADC I and Q<br>0101b - RAW ADC I Only<br>0110b - RAW ADC Q only<br>0111b - DC Estimator I and Q<br>1000b - DC Estimator I Only<br>1001b - DC Estimator Q only<br>1010b - RX_DIG Phase Output<br>1011b - Reserved<br>1100b - Demodulator Soft Decision<br>1101b - Demodulator Data Output<br>1110b - Demodulator CFO Phase Output<br>1111b - Reserved |

## 64.3.10.1.8 TRANSCEIVER DMA DATA (DMA_DATA)

### 64.3.10.1.8.1 Offset

| Register | Offset |
|---|---|
| DMA_DATA | 18h |

### 64.3.10.1.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMA_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMA_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.1.8.3 Fields

| Field | Function |
|---|---|
| 31-0 | DMA Data Register |

| Field | Function |
|---|---|
| DMA_DATA | This register is the singular address location at which the SoC DMA controller accesses samples from the transceiver for transfer to system memory. During DMA operation, the contents of this register depend on the DMA Page selected. The DMA Data register is intended for DMA purposes only, not Host IPS accesses, but Host IPS bus access to this register is not prohibited. |

## 64.3.10.1.9 PACKET RAM CONTROL (PACKET_RAM_CTRL)

### 64.3.10.1.9.1 Offset

| Register | Offset |
|---|---|
| PACKET_RAM_CTRL | 1Ch |

### 64.3.10.1.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RAM1_CE_ON_OVRD | RAM1_CE_ON_OVRD_EN | RAM0_CE_ON_OVRD | RAM0_CE_ON_OVRD_EN | RAM1_CLK_ON_OVRD | RAM1_CLK_ON_OVRD_EN | RAM0_CLK_ON_OVRD | RAM0_CLK_ON_OVRD_EN | PB_PROTECT | DBG_STOP_TRIGGERED | DBG_START_TRIGGERED | DBG_STOP_EDGE | DBG_STOP_TRG | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DBG_START_EDGE | DBG_START_TRG | | | DBG_SOFT_INFO_SEL | DBG_AA_TRIGGERED | DBG_RAM_FULL | | ALL_PROTOCOLS_ALLOW | XCVR_RAM_ALLOW | XCVR_RAM_PAGE | DBG_EN | DBG_PAGE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.1.9.3 Fields

| Field | Function |
|---|---|
| 31<br>RAM1_CE_ON_OVRD | Override value for RAM1 CE (Chip Enable)<br>When RAM1_CE_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM1 CE. This bit is ignored when RAM1_CE_ON_OVRD_EN=0. |
| 30<br>RAM1_CE_ON_OVRD_EN | Override control for RAM1 CE (Chip Enable)<br>0b - Normal operation.<br>1b - Use the state of RAM1_CE_ON_OVRD to override the RAM1 CE. |
| 29<br>RAM0_CE_ON_OVRD | Override value for RAM0 CE (Chip Enable)<br>When RAM0_CE_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM0 CE. This bit is ignored when RAM0_CE_ON_OVRD_EN=0. |
| 28<br>RAM0_CE_ON_OVRD_EN | Override control for RAM0 CE (Chip Enable)<br>0b - Normal operation.<br>1b - Use the state of RAM0_CE_ON_OVRD to override the RAM0 CE. |
| 27<br>RAM1_CLK_ON_OVRD | Override value for RAM1 Clock Gate Enable<br>When RAM1_CLK_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM1 Clock Gate Enable. This bit is ignored when RAM1_CLK_ON_OVRD_EN=0. |
| 26<br>RAM1_CLK_ON_OVRD_EN | Override control for RAM1 Clock Gate Enable<br>0b - Normal operation.<br>1b - Use the state of RAM1_CLK_ON_OVRD to override the RAM1 Clock Gate Enable. |
| 25<br>RAM0_CLK_ON_OVRD | Override value for RAM0 Clock Gate Enable<br>When RAM0_CLK_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM0 Clock Gate Enable. This bit is ignored when RAM0_CLK_ON_OVRD_EN=0. |
| 24<br>RAM0_CLK_ON_OVRD_EN | Override control for RAM0 Clock Gate Enable<br>0b - Normal operation.<br>1b - Use the state of RAM0_CLK_ON_OVRD to override the RAM0 Clock Gate Enable. |
| 23<br>PB_PROTECT | Packet Buffer Protect<br>Protect Packet Buffer contents against overwriting by the next received packet. Applies to all protocols except BLE<br>0b - Incoming received packets overwrite Packet Buffer RX contents (default)<br>1b - Incoming received packets are blocked from overwriting Packet Buffer RX contents |
| 22<br>DBG_STOP_TRIGGERED | Packet RAM Debug Stop Triggered<br>This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by DBG_STOP_TRG[3:0] occurs, with the edge sensitivity selected by DBG_STOP_EDGE. To clear this bit, set DBG_PAGE=0. |
| 21<br>DBG_START_TRIGGERED | Packet RAM Debug Start Triggered<br>This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by DBG_START_TRG[2:0] occurs, with the edge sensitivity selected by DBG_START_EDGE. To clear this bit, set DBG_PAGE=0. |
| 20<br>DBG_STOP_EDGE | Packet RAM Debug Stop Trigger Edge Selector<br>DBG_STOP_EDGE selects the edge sensitivity (rising or falling) of the selected stop-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 19-16 | Packet RAM Debug Stop Trigger Selector<br>The Packet RAM Debug Stop Trigger, if desired, can be selected from the sources in the following table: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DBG_STOP_TRG | 0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare3_en<br>8: TSM: pll unlock<br>9: BLE: crc_err_count_incr (increment DTM RX bad packet counter)<br>10: CRC FAIL: 802.15.4 or GEN_FSK only, selected by PROTOCOL[3:0]<br>11: HEADER FAIL: GEN_FSK only<br>12-15: Reserved |
| 15<br>DBG_START_EDGE | Packet RAM Debug Start Trigger Edge Selector<br>DBG_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 14-12<br>DBG_START_TRG | Packet RAM Debug Start Trigger Selector<br>The Packet RAM Debug Start Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| 11<br>DBG_SOFT_INFO_SEL | Packet RAM Debug PHY Soft Info Output Selector<br>When acquiring data in Packet RAM Debug Mode with the DBG_PAGE=DMDSOFT page, this bit selects the signal from the PHY used to capture the soft decision data.<br>0b - PHY output bit_valid_int is used to capture soft decision data<br>1b - PHY output demod_vout is used to capture soft decision data |
| 10<br>DBG_AA_TRIGGERED | Packet Ram Debug Access Address triggered<br>This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of Packet RAM debug transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DBG_PAGE to initiate data transfers to Packet RAM. This bit is intended for use with DBG_PAGE=13, but is available on all pages. Its usage is optional. |
| 9-8<br>DBG_RAM_FULL | DBG_RAM_FULL[1:0]<br>Status Bits indicating that Packet RAM0 (or RAM1) is full, and the Packet RAM Debug engine has attempted to write another word to that RAM. This, and any subsequent write attempts, will not alter the contents of the RAM, once it has reached capacity. Set DBG_PAGE=0 to clear the DBG_RAM_FULL[1:0] bits.<br>**NOTE:** If a Debug Stop Trigger is selected (DBG_STOP_TRG > 0), writes to RAM do not stop when DBG_RAM_FULL bit(s) become set; instead, writes continue after an address wrap-around to 0, and will continue indefinitely until either a stop trigger condition occurs, or the debug session is ended by taking DBG_PAGE=0. The DBG_RAM_FULL[1:0] remain a valid indicator that the RAM(s) has reached full status at least once during the debug session.<br>00b - Neither Packet RAM0 nor RAM1 is full<br>1xb - Packet RAM1 has been filled to capacity.<br>x1b - Packet RAM0 has been filled to capacity. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7<br>ALL_PROTOCOLS_ALLOW | Allow IPS bus access to Packet RAM for any protocol at any time.<br>Each supported protocol has an associated IPS bus, with which it can access protocol-specific registers, as well as Packet RAM. Normally IPS bus access to the Packet RAM is restricted to the protocol currently selected by the XCVR_CTRL[PROTOCOL] register, and access attempts by other protocols will result in an assert of **ips_xfr_err** on the offending IPS interface. When ALL_PROTOCOLS_ALLOW=1, these inhibits are removed, and any IPS bus can access Packet RAM at any time without any error signalling.<br>0b - IPS bus access to Packet RAM is restricted to the protocol engine currently selected by XCVR_CTRL[PROTOCOL].<br>1b - All IPS bus access to Packet RAM permitted, regardless of XCVR_CTRL[PROTOCOL] setting |
| 6<br>XCVR_RAM_ALLOW | Allow Packet RAM Transceiver Access<br>This bit must be set before performing accesses to the Packet RAM using transceiver address space. Transceiver space accesses to Packet RAM are intended for debug purposes only; the individual protocol engines, and the associated IPS busses, have exclusive access to Packet RAM in mission modes. Transceiver space accesses to Packet RAM include direct accesses to RAM at transceiver addresses 0x700 - 0xFFF, as well as Packet RAM Debug Mode. When this bit is set, control of RAM clock gating and RAM chip enables are forced on continuously, taking these controls away from the protocol engines.<br>0b - Protocol Engines, and associated IPS busses, have exclusive access to Packet RAM (mission mode)<br>1b - Transceiver-space access to Packet RAM, including Packet RAM debug mode, are allowed |
| 5<br>XCVR_RAM_PAGE | RAM Page Selector for XCVR Access<br>This bit selects which of the 2 Packet RAM blocks is mapped into XCVR address space starting at XCVR_BASE + 0x700.<br>0b - RAM0 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF<br>1b - RAM1 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF |
| 4<br>DBG_EN | Packet RAM Debug Mode Enable<br>Setting DBG_EN=1 enables clock gating to the Packet RAM Debug engine. This bit should be set prior to the start of a Debug session, and remain set for the duration of the session. |
| 3-0<br>DBG_PAGE | Packet RAM Debug Page Selector<br>Selects the page of receiver data for storage to Packet RAM using Transceiver Packet RAM Debug Mode. Setting this register to a non-zero value enables the Packet RAM Debug Mode interface logic. Setting this register to zero disables the interface. **Note:** DBG_PAGE should be set to a non-zero value to begin an acquisition, only after the start and stop triggers have been programmed (DBG_START_TRG and DBG_STOP_TRG), if a trigger is desired. The available RAM Debug pages are listed below.<br>0000b - Packet RAM Debug Mode Idle<br>0001b - RX_DIG I and Q<br>0010b - Reserved<br>0011b - Reserved<br>0100b - RAW ADC I and Q<br>0101b - Reserved<br>0110b - Reserved<br>0111b - DC Estimator I and Q<br>1000b - Reserved<br>1001b - Reserved<br>1010b - RX_DIG Phase Output<br>1011b - Reserved<br>1100b - Demodulator Soft Decision<br>1101b - Demodulator Data Output<br>1110b - Demodulator CFO Phase Output<br>1111b - Reserved |

# 64.3.10.1.10 PACKET RAM DEBUG RAM STOP ADDRESS (RAM_STOP_ADDR)

## 64.3.10.1.10.1 Offset

| Register | Offset |
|---|---|
| RAM_STOP_ADDR | 20h |

## 64.3.10.1.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | RAM1_STOP_ADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | RAM0_STOP_ADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 64.3.10.1.10.3 Fields

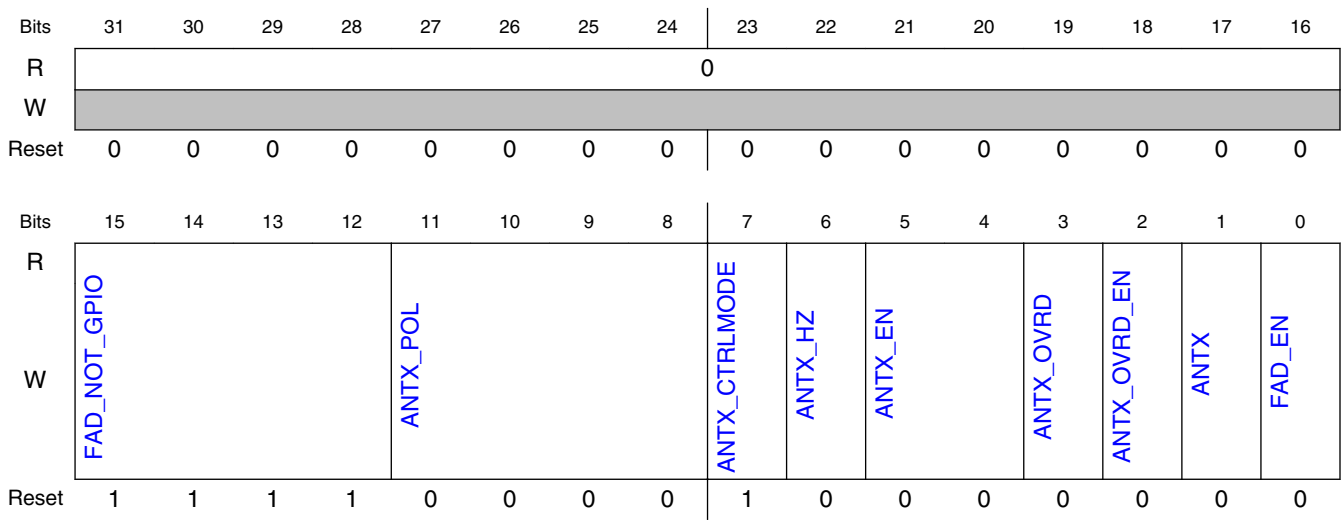
| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-16<br>RAM1_STOP_ADDR | RAM1 Stop Address<br>After a Packet RAM Debug Mode acquisition, RAM1_STOP_ADDR represents the last (most recent) RAM1 location filled with sample data before the acquision terminated. The acquisition terminates when a stop-trigger occurs, or due to a manual termination by setting DBG_PAGE=0. RAM1_STOP_ADDR is in units of RAM "words", where each word represents 2 bytes of XCVR RAM address space. |
| 15-11<br>— | Reserved |
| 10-0<br>RAM0_STOP_ADDR | RAM0 Stop Address<br>After a Packet RAM Debug Mode acquisition, RAM0_STOP_ADDR represents the last (most recent) RAM0 location filled with sample data before the acquision terminated. The acquisition terminates when a stop-trigger occurs, or due to a manual termination by setting DBG_PAGE=0. RAM0_STOP_ADDR is in units of RAM "words", where each word represents 2 bytes of XCVR RAM address space. |

# 64.3.10.1.11 FAD CONTROL (FAD_CTRL)

## 64.3.10.1.11.1 Offset

| Register | Offset |
|---|---|
| FAD_CTRL | 24h |

## 64.3.10.1.11.2 Diagram




## 64.3.10.1.11.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>FAD_NOT_GPIO | FAD versus GPIO Mode Selector<br>**xxx1**: The ANT_A pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**xxx0**: The ANT_A pad is controlled directly by the TSM output gpio0_trig_en<br>**xx1x**: The ANT_B pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**xx0x**: The ANT_B pad is controlled directly by the TSM output gpio1_trig_en<br>**x1xx**: The TX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**x0xx**: The TX_SWITCH pad is controlled directly by the TSM output gpio2_trig_en |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **1xxx**: The RX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**0xxx**: The RX_SWITCH pad is controlled directly by the TSM output gpio3_trig_en |
| 11-8<br>ANTX_POL | FAD Antenna Controls Polarity<br>Control the polarity of the FAD pins:<br>ANTX_POL<0> : This bit currently has no functionality<br>ANTX_POL<1> : This bit currently has no functionality<br>ANTX_POL<2>=1 : invert the TX_SWITCH output<br>ANTX_POL<3>=1 : invert the RX_SWITCH output |
| 7<br>ANTX_CTRLMODE | Antenna Diversity Control Mode<br>When ANTX_CTRLMODE=1 (dual mode):<br>ANT_A=NOT(ANTX) AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>ANT_B=ANTX AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>TX_SWITCH=GPIO2_TRIG_EN<br>RX_SWITCH=GPIO3_TRIG_EN<br><br>When ANTX_CTRLMODE=0 (single mode):<br>ANT_A=NOT(ANTX) AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>ANT_B=ANTX AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>TX_SWITCH=GPIO2_TRIG_EN<br>RX_SWITCH=(GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br><br>GPIO2_TRIG_EN and GPIO3_TRIG_EN are outputs of the Transceiver Sequence Manager (TSM). The TSM timing registers associated with GPIO2_TRIG_EN and GPIO3_TRIG_EN should be programmed with the desired TX_SWITCH and RX_SWITCH timing, before enabling Fast Antenna Diversity. |
| 6<br>ANTX_HZ | FAD PAD Tristate Control<br>0b - ANT_A, ANT_B, RX_SWITCH and TX_SWITCH are actively driven outputs.<br>1b - Antenna controls high impedance- Set ANT_A, ANT_B, RX_SWITCH and TX_SWITCH in high impedance. |
| 5-4<br>ANTX_EN | FAD Antenna Controls Enable<br>For any of the 4 FAD-related outputs {ANT_A, ANT_B, TX_SWITCH, RX_SWITCH}, when the associated register control bit (FAD_NOT_GPIO[x]=1) puts that output in FAD mode, ANTX_EN[1:0] determines which pairs of FAD related outputs are enabled:<br>00b - all disabled (held low)<br>01b - only RX/TX_SWITCH enabled<br>10b - only ANT_A/B enabled<br>11b - all enabled |
| 3<br>ANTX_OVRD | Antenna State Override Value<br>When ANTX_OVRD_EN=1, the FAD antenna state (ANTX) is overriden by the value of this register; otherwise, antenna state is determined by the 802.15.4 FAD state machine, if XCVR_CTRL[DEMOD_SEL]=0b10, or the FSK PHY FAD state machine if XCVR_CTRL[DEMOD_SEL]=0b01. |
| 2<br>ANTX_OVRD_EN | Antenna State Override Enable<br>When ANTX_OVRD_EN=1, the FAD antenna state (ANTX) is overriden by the value of the ANTX_OVRD register; otherwise, antenna state is determined by the 802.15.4 FAD state machine, if XCVR_CTRL[DEMOD_SEL]=0b10, or the FSK PHY FAD state machine if XCVR_CTRL[DEMOD_SEL]=0b01. |
| 1<br>ANTX | Antenna Selection State |

*Table continues on the next page...*

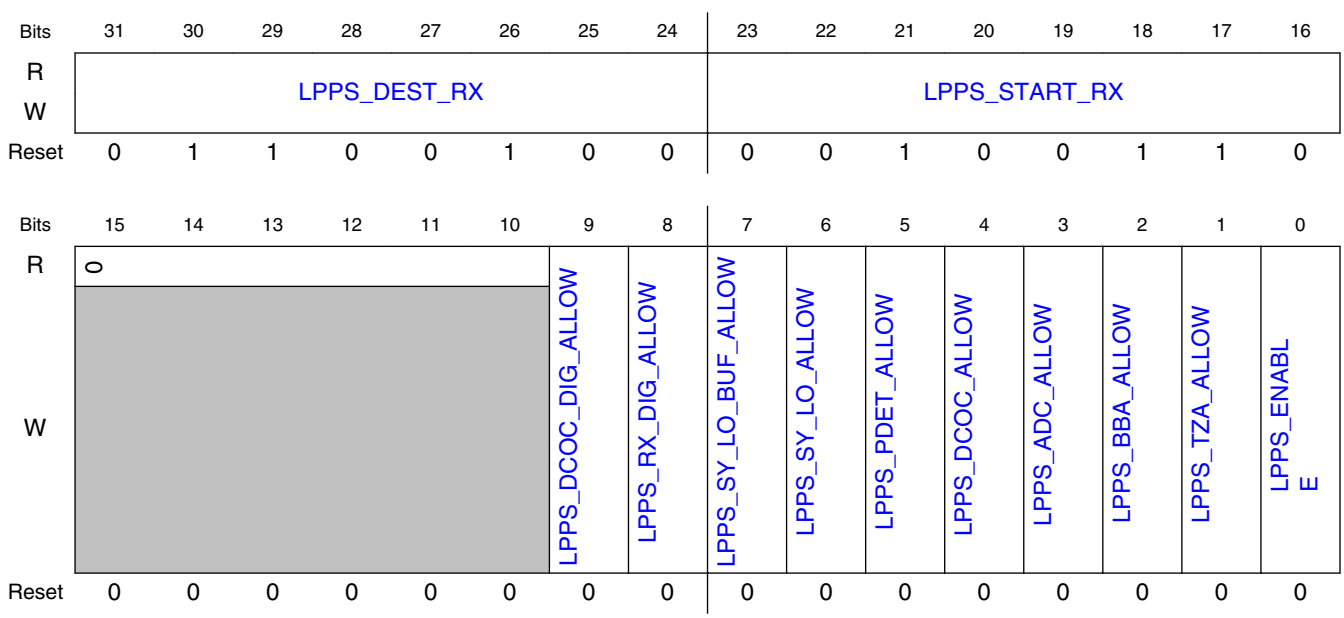
| Field | Function |
|---|---|
| | If FAD_EN=0, the ANTX bit is used to take manual (software) control of the antenna selection, overriding the FAD state machine; in this case, the readback value of ANTX is whatever was last written by the host.<br>If FAD_EN=1, the FAD state machine controls antenna selection, and the readback value of ANTX reflects the machine-selected antenna. |
| 0<br>FAD_EN | Fast Antenna Diversity Enable<br>0b - Fast Antenna Diversity disabled<br>1b - Fast Antenna Diversity enabled |

## 64.3.10.1.12 LOW POWER PREAMBLE SEARCH CONTROL (LPPS_CTRL)

### 64.3.10.1.12.1 Offset

| Register | Offset |
|---|---|
| LPPS_CTRL | 28h |

### 64.3.10.1.12.2 Diagram




### 64.3.10.1.12.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LPPS_DEST_RX | LPPS Fast TSM RX Warmup "Jump-to" Point |

*Table continues on the next page...*

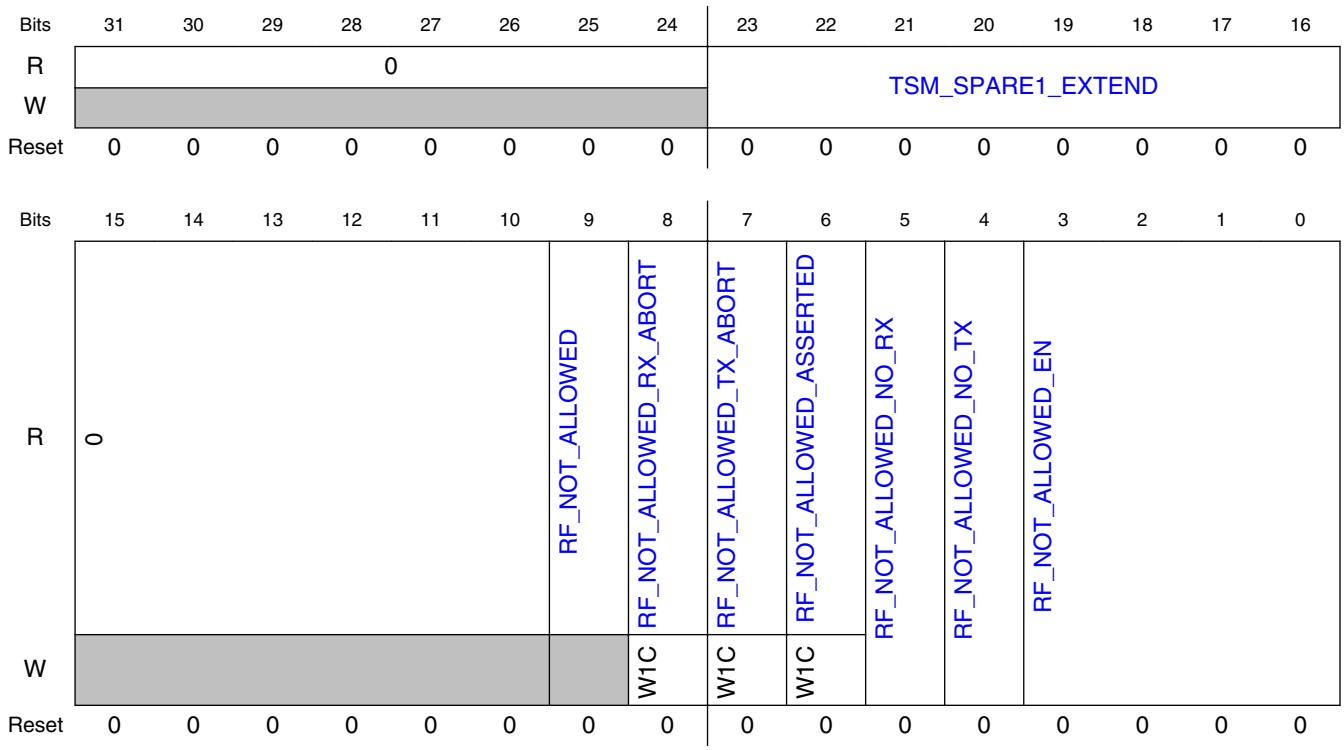
| Field | Function |
|---|---|
| | During a LPPS Fast RX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. In an LPPS Fast RX Warmup, LPPS_START_RX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>LPPS_START_RX = 10<br>LPPS_DEST_RX = 15<br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| 23-16<br>LPPS_START_RX | LPPS Fast TSM RX Warmup "Jump-from" Point<br>During a LPPS Fast RX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. In an LPPS Fast RX Warmup, this count will never be reached; its place in the TSM warmup will be replaced by the contents of LPPS_DEST_RX[7:0], thereby executing the jump. |
| 15-10<br>— | Reserved |
| 9<br>LPPS_DCOC_DIG_ALLOW | LPPS_DCOC_DIG_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: dcoc_en |
| 8<br>LPPS_RX_DIG_ALLOW | LPPS_RX_DIG_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_dig_en |
| 7<br>LPPS_SY_LO_BUF_ALLOW | LPPS_SY_LO_BUF_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_buf_en |
| 6<br>LPPS_SY_LO_ALLOW | LPPS_SY_LO_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_en |
| 5<br>LPPS_PDET_ALLOW | LPPS_PDET_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_pdet_en, rx_tza_pdet_en |
| 4<br>LPPS_DCOC_ALLOW | LPPS_DCOC_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_dcoc_en, rx_tza_dcoc_en |
| 3<br>LPPS_ADC_ALLOW | LPPS_ADC_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_adc_en, rx_adc_i_en, rx_adc_q_en |
| 2<br>LPPS_BBA_ALLOW | LPPS_BBA_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_i_en, rx_bba_q_en |
| 1<br>LPPS_TZA_ALLOW | LPPS_TZA_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_lna_en, rx_tza_i_en, rx_tza_q_en |
| 0<br>LPPS_ENABLE | LPPS_ENABLE<br>Master enable for LPPS mode. Allows correlators to be duty-cycled during Preamble Search, and selected analog and digital blocks to be duty-cycled simultaneously. |

# 64.3.10.1.13 COEXISTENCE CONTROL (COEX_CTRL)

## 64.3.10.1.13.1 Offset

| Register | Offset |
|---|---|
| COEX_CTRL | 2Ch |

## 64.3.10.1.13.2 Diagram




## 64.3.10.1.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>TSM_SPARE1_EXTEND | TSM_SPARE1_EX Extension Duration<br>When TSM output TSM_SPARE1_EN is enabled to assert during any TX or RX sequence and TSM_SPARE1_EXTEND[7:0] > 0, this register specifies the number of microseconds for which TSM_SPARE1_EN is to remain asserted beyond the end of the TSM warmdown. This extension is intended to close any gap in the coexistence output RF_ACTIVE that may occur between consecutive TX/RX or RX/TX TSM operations. |

*Table continues on the next page...*

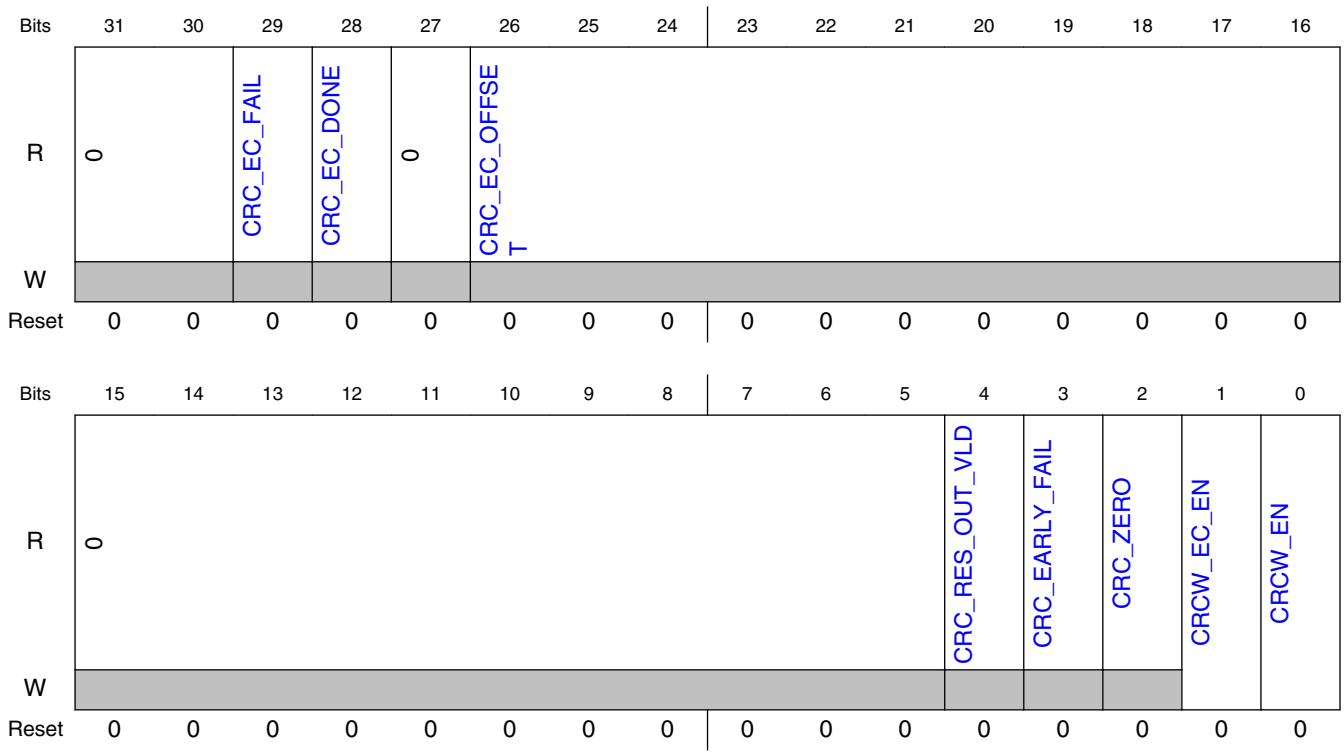
| Field | Function |
|---|---|
| 15-10<br>— | Reserved |
| 9<br>RF_NOT_ALLOWED | RF_NOT_ALLOWED<br>Reflects the instantaneous state of the RF_NOT_ALLOWED pin, synchronized into the RF OSC clock domain. |
| 8<br>RF_NOT_ALLOWED_RX_ABORT | RF_NOT_ALLOWED_RX_ABORT<br>0b - A RX abort due to assertion on RF_NOT_ALLOWED has not occurred<br>1b - A RX abort due to assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 7<br>RF_NOT_ALLOWED_TX_ABORT | RF_NOT_ALLOWED_TX_ABORT<br>0b - A TX abort due to assertion on RF_NOT_ALLOWED has not occurred<br>1b - A TX abort due to assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 6<br>RF_NOT_ALLOWED_ASSERTED | RF_NOT_ALLOWED_ASSERTED<br>0b - Assertion on RF_NOT_ALLOWED has not occurred<br>1b - Assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 5<br>RF_NOT_ALLOWED_NO_RX | RF_NOT_ALLOWED_NO_RX<br>0b - Assertion on RF_NOT_ALLOWED has no effect on RX<br>1b - Assertion on RF_NOT_ALLOWED can abort RX |
| 4<br>RF_NOT_ALLOWED_NO_TX | RF_NOT_ALLOWED_NO_TX<br>0b - Assertion on RF_NOT_ALLOWED has no effect on TX<br>1b - Assertion on RF_NOT_ALLOWED can abort TX |
| 3-0<br>RF_NOT_ALLOWED_EN | RF_NOT_ALLOWED PER-LINK-LAYER ENABLE<br>The coexistence input RF_NOT_ALLOWED can be enabled to selectively abort TX or RX sequences, with individual enables for each supported protocol, subject also to the state of the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits, according to the following table:<br>**xxx1**: RF_NOT_ALLOWED assertions are enabled to abort BLE TX and RX sequences<br>**xxx0**: RF_NOT_ALLOWED assertions are not enabled to abort BLE TX and RX sequences<br>**1xxx**: RF_NOT_ALLOWED assertions are enabled to abort GENERIC_FSK TX and RX sequences<br>**0xxx**: RF_NOT_ALLOWED assertions are not enabled to abort GENERIC_FSK TX and RX sequences |

## 64.3.10.1.14 CRC/WHITENER CONFIG REGISTER (CRCW_CFG)

### 64.3.10.1.14.1 Offset

| Register | Offset |
|---|---|
| CRCW_CFG | 30h |

### 64.3.10.1.14.2 Diagram




### 64.3.10.1.14.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>CRC_EC_FAIL | CRC error correction fail<br>This signal is cleared when *crc_init* is asserted. It is set after the completion of a CRC error correction calculation that does not find a valid data correction solution or if error correction was disabled and the CRC check found an error. |
| 28<br>CRC_EC_DONE | CRC error correction done<br>This signal is cleared when *crc_init* is asserted. It is set after the error correction logic completes processing the syndrome. It is immediately set after a packet is received if no error was detected or if error correction is disabled. Otherwise, it can take up to N system clocks after the packet is received to assert, with N = (number of bits used in the CRC calculation, including the CRC value). |
| 27<br>— | Reserved |
| 26-16<br>CRC_EC_OFFSET | CRC error correction offset<br>This value provides the byte offset within the data packet to which the CRC error correction mask should be XOR-ed. This value is valid if the *crc_ec_done* signal is asserted and the *crc_ec_fail* signal is not asserted. The offset includes only the bytes used in the CRC calculation, including the CRC value. |
| 15-5 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 4<br>CRC_RES_OUT_VLD | CRC result output valid<br>CRC result output valid |
| 3<br>CRC_EARLY_FAIL | CRC error correction fail<br>This signal is cleared when *crc_init* is asserted. It is set after the completion of a CRC error correction calculation that does not find a valid data correction solution or if error correction was disabled and the CRC check found an error. |
| 2<br>CRC_ZERO | CRC zero<br>This signal is asserted at any time that the CRC shift register contains a value of zero |
| 1<br>CRCW_EC_EN | CRC Error Correction Enable<br>CRC Error Correction Enable |
| 0<br>CRCW_EN | CRC calculation enable<br>Input data bits loaded with this signal asserted are used in the CRC calculation. |

## 64.3.10.1.15 CRC ERROR CORRECTION MASK (CRC_EC_MASK)

### 64.3.10.1.15.1 Offset

| Register | Offset |
|---|---|
| CRC_EC_MASK | 34h |

### 64.3.10.1.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_EC_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_EC_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.1.15.3 Fields

| Field | Function |
|---|---|
| 31-0 | CRC error correction mask |

| Field | Function |
|---|---|
| CRC_EC_MASK | This value provides a 32-bit XOR mask that must be applied to the input data packet to correct the burst errors detected by the error correction calculation. This value is valid if the *crc_ec_done* signal is asserted and the *crc_ec_fail* signal is not asserted. |

## 64.3.10.1.16 CRC RESULT (CRC_RES_OUT)

### 64.3.10.1.16.1 Offset

| Register | Offset |
|---|---|
| CRC_RES_OUT | 38h |

### 64.3.10.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_RES_OUT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_RES_OUT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.1.16.3 Fields

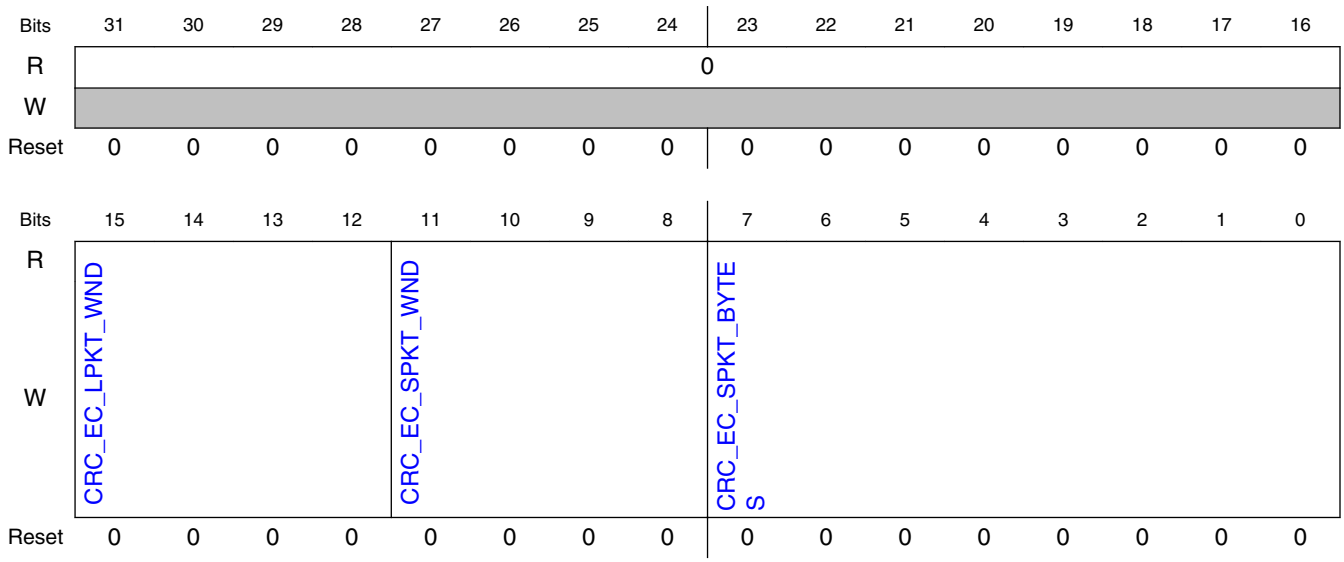
| Field | Function |
|---|---|
| 31-0<br>CRC_RES_OUT | CRC result output<br>This bus provides the instantaneous value of the CRC shift register. |

## 64.3.10.1.17 CRC/WHITENER CONFIG 2 REGISTER (CRCW_CFG2)

### 64.3.10.1.17.1 Offset

| Register | Offset |
|---|---|
| CRCW_CFG2 | 3Ch |

### 64.3.10.1.17.2 Diagram




### 64.3.10.1.17.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>CRC_EC_LPKT_WND | Error correction long packet burst error window<br>This value determines the largest burst error corrected when long packets are processed. A value of zero will disable long packet error correction. The max value is 15. |
| 11-8<br>CRC_EC_SPKT_WND | Error correction short packet burst error window<br>This value determines the largest burst error corrected when short packets are processed. A value of zero will disable short packet error correction. The max value is 15. |
| 7-0<br>CRC_EC_SPKT_BYTES | Error Correction Short Packet Bytes<br>Short packets are defined as those where the number of bytes used in the CRC calculation, including the CRC value, do not exceed this value. The max value is 255. |

## 64.3.10.2 XCVR_PKT_RAM register descriptions

### 64.3.10.2.1 XCVR_PKT_RAM_ADDR Memory map

XCVR_PKT_RAM base address: 4103_0700h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 8FEh | Shared Packet RAM for multiple Link Layer usage. (PACKET_RAM_0 - PACKET_RAM_1151) | 16 | RW | See description. |

#### 64.3.10.2.2 Shared Packet RAM for multiple Link Layer usage. (PACKET_RAM_0 - PACKET_RAM_1151)

##### 64.3.10.2.2.1 Offset

For a = 0 to 1151:

| Register | Offset |
|---|---|
| PACKET_RAM_a | 0h + (a × 2h) |

##### 64.3.10.2.2.2 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MSBYTE | | | | | | | | LSBYTE | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

##### 64.3.10.2.2.3 Fields

| Field | Function |
|---|---|
| 15-8<br>MSBYTE | MSBYTE |
| 7-0<br>LSBYTE | LSBYTE |

### 64.3.10.3 XCVR_PHY register descriptions

**Note**

To decipher number formats such as ‘ufix4en2’ or ‘sfix8en3’:

General form: [u/s]fix[Nb]e[n][Nx]

Where,

| | |
|---|---|
| [<br>u<br>/<br>s<br>] | denotes whether the number is signed or not |
| f<br>i<br>x | denotes that the number is fixed point, 2's complement |
| N<br>b | indicates total number of bits, including a sign bit if signed |
| e | indicates that the number is exponentiated like num x 2^(exponent) |
| n | indicates the exponent is negative |
| N<br>x | indicates the value of the exponent |

For example :

- ufix4en2(1101) = 1101 x 2^(-2) = 11.01 = 3.25 (decimal)
- sfix8en3(10011011) = -1100101 x 2^(-3) = -1100.101 = -12.625 (decimal)

### 64.3.10.3.1 XCVR_RX_PHY_ADDR Memory map

XCVR_PHY base address: 4103_0400h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Preamble Detect Config 0 (PHY_FSK_PD_CFG0) | 32 | RW | 0021_081Eh |
| 4h | Preamble Detect Config 1 (PHY_FSK_PD_CFG1) | 32 | RW | 0ACD_2BDEh |
| 8h | PHY Configuration (PHY_FSK_CFG) | 32 | RW | E180_B070h |
| Ch | PHY Misc. Configuration (PHY_FSK_MISC) | 32 | RW | 0800_0FF0h |
| 10h | PHY BSM Network Address (NTW_ADR_BSM) | 32 | RW | 0000_0000h |
| 14h | PHY Status (FSK_STAT) | 32 | RO | See description. |
| 18h | PHY FAD control (FSK_FAD_CTRL) | 32 | RW | 0E41_8140h |

### 64.3.10.3.2 Preamble Detect Config 0 (PHY_FSK_PD_CFG0)

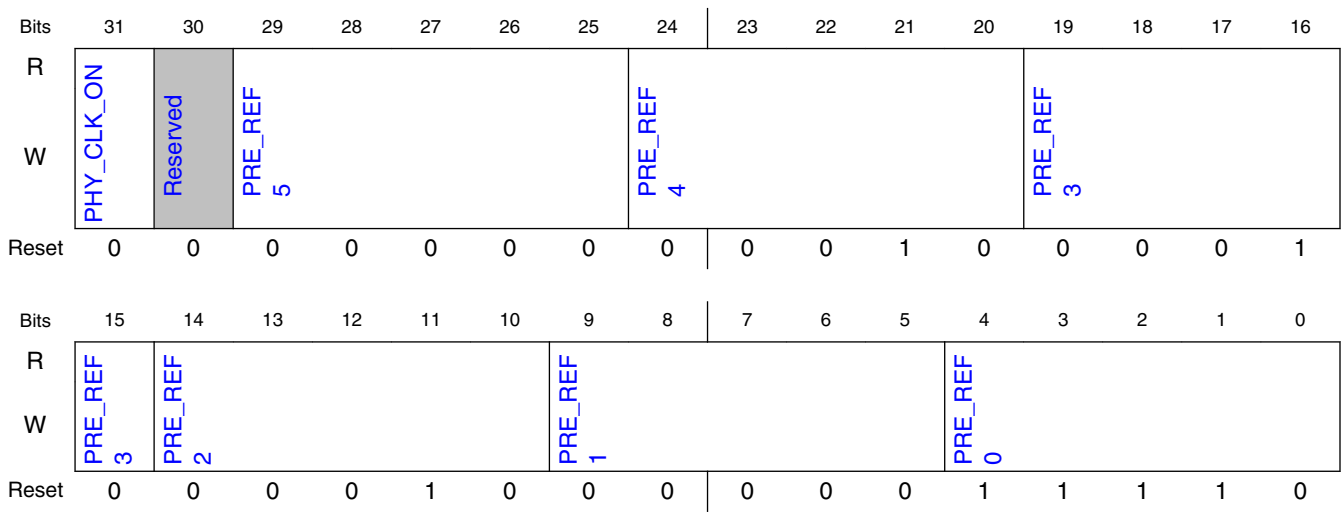
### 64.3.10.3.2.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_PD_CFG0 | 0h |

### 64.3.10.3.2.2 Function

Reference waveform configuration and PHY clock force.

### 64.3.10.3.2.3 Diagram




### 64.3.10.3.2.4 Fields

| Field | Function |
|---|---|
| 31<br>PHY_CLK_ON | Force PHY clock ON<br>Forces the system to keep the PHY clock on. |
| 30<br>— | Reserved. |
| 29-25<br>PRE_REF5 | Reference waveform sample 5 (sfix5en5) |
| 24-20<br>PRE_REF4 | Reference waveform sample 4 (sfix5en5) |
| 19-15<br>PRE_REF3 | Reference waveform sample 3 (sfix5en5) |
| 14-10<br>PRE_REF2 | Reference waveform sample 2 (sfix5en5) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 9-5<br>PRE_REF1 | Reference waveform sample 1 (sfix5en5) |
| 4-0<br>PRE_REF0 | Reference waveform sample 0 (sfix5en5) |

## 64.3.10.3.3 Preamble Detect Config 1 (PHY_FSK_PD_CFG1)

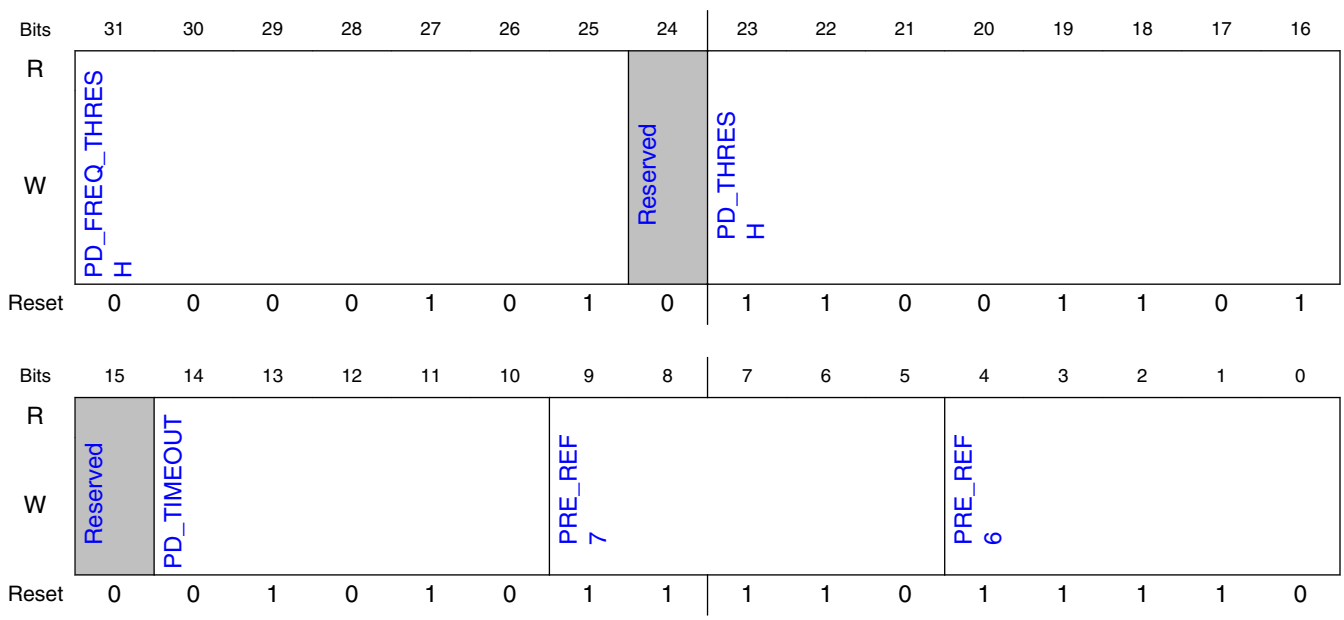
### 64.3.10.3.3.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_PD_CFG1 | 4h |

### 64.3.10.3.3.2 Function

Preamble detect configuration.

### 64.3.10.3.3.3 Diagram



#### 64.3.10.3.3.4 Fields

| Field | Function |
|---|---|
| 31-25<br>PD_FREQ_THRESH | Preamble Detect frequency threshold<br>Frequency range within which 2 preamble detects are considered similar in frequency and, thus, correlated. Format is ufix7en9 and can be estimated by: PD_FREQ_THRESH = round(1024*(Freq_Thresh/(OSR*SYMBOL_RATE)) |
| 24<br>— | Reserved. |
| 23-16<br>PD_THRESH | Preamble detect threshold<br>Preamble detect correlation threshold (ufix8en8). This threshold is used to define high correlation for the preamble detect logic. It is the default threshold used and corresponds to PD MODE A. |
| 15<br>— | Reserved. |
| 14-10<br>PD_TIMEOUT | Preamble Detect Timeout<br>Timeout parameter for preamble detect (in samples). After timeout occurs, a better correlation is no longer searched for, and the acquired frequency estimate is committed. |
| 9-5<br>PRE_REF7 | Reference waveform sample 7 (sfix5en5) |
| 4-0<br>PRE_REF6 | Reference waveform sample 6 (sfix5en5) |

## 64.3.10.3.4 PHY Configuration (PHY_FSK_CFG)

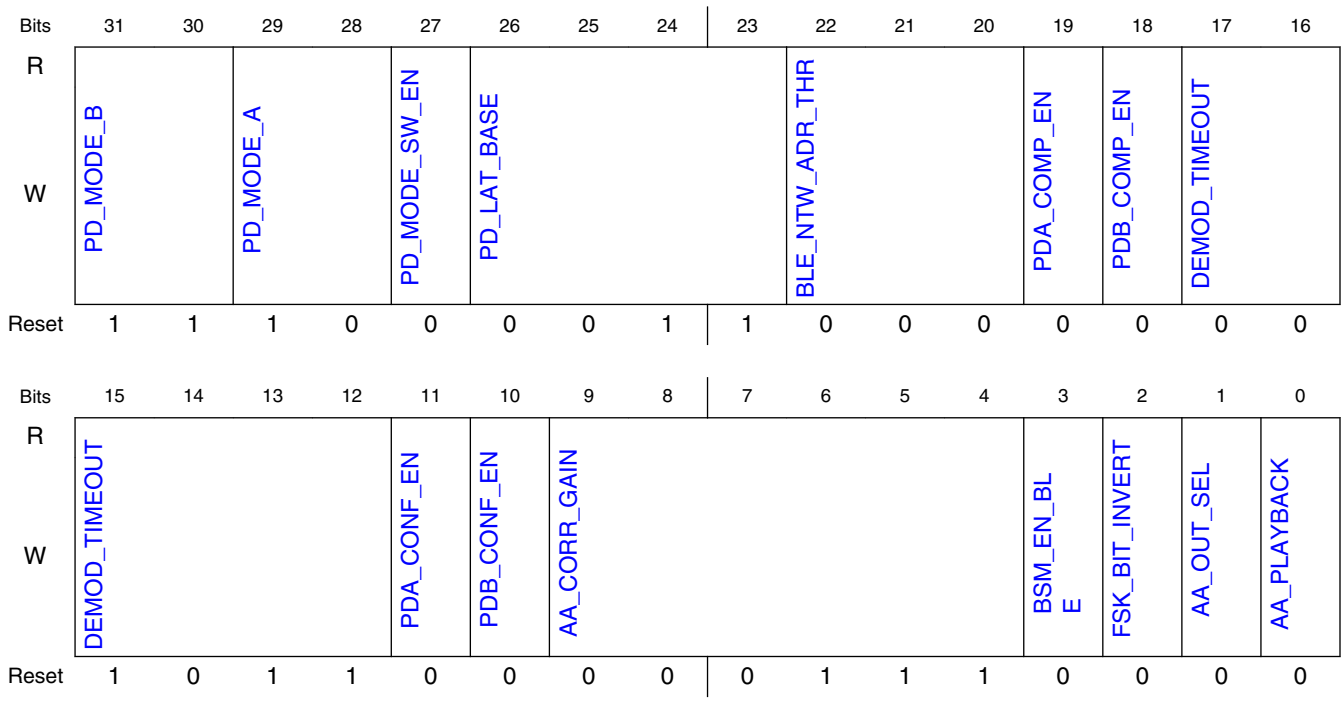
### 64.3.10.3.4.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_CFG | 8h |

### 64.3.10.3.4.2 Function

PHY configuration.

### 64.3.10.3.4.3 Diagram




### 64.3.10.3.4.4 Fields

| Field | Function |
|---|---|
| 31-30<br>PD_MODE_B | Preamble detect configuration for PD Mode B.<br>10b - PD mode 2, pattern based preamble detection.<br>11b - PD mode 3, peak based preamble detection. |
| 29-28<br>PD_MODE_A | Preamble detect configuration for PD Mode A.<br>10b - PD mode 2, pattern based preamble detection.<br>11b - PD mode 3, peak based preamble detection. |
| 27<br>PD_MODE_SW_EN | Allow PD mode switching. When enabled, an external signal is observed to switch between PD mode A and PD mode B. If not enabled, PD mode A is always used. |
| 26-23<br>PD_LAT_BASE | User configurable adjustment to the PD latency, in samples. |
| 22-20<br>BLE_NTW_ADR_THR | Allowable hamming distance for BLE AA search. |
| 19<br>PDA_COMP_EN | Enables compressed correlation in PD Mode A. |
| 18<br>PDB_COMP_EN | Enables compressed correlation in PD Mode B. |

*Table continues on the next page...*

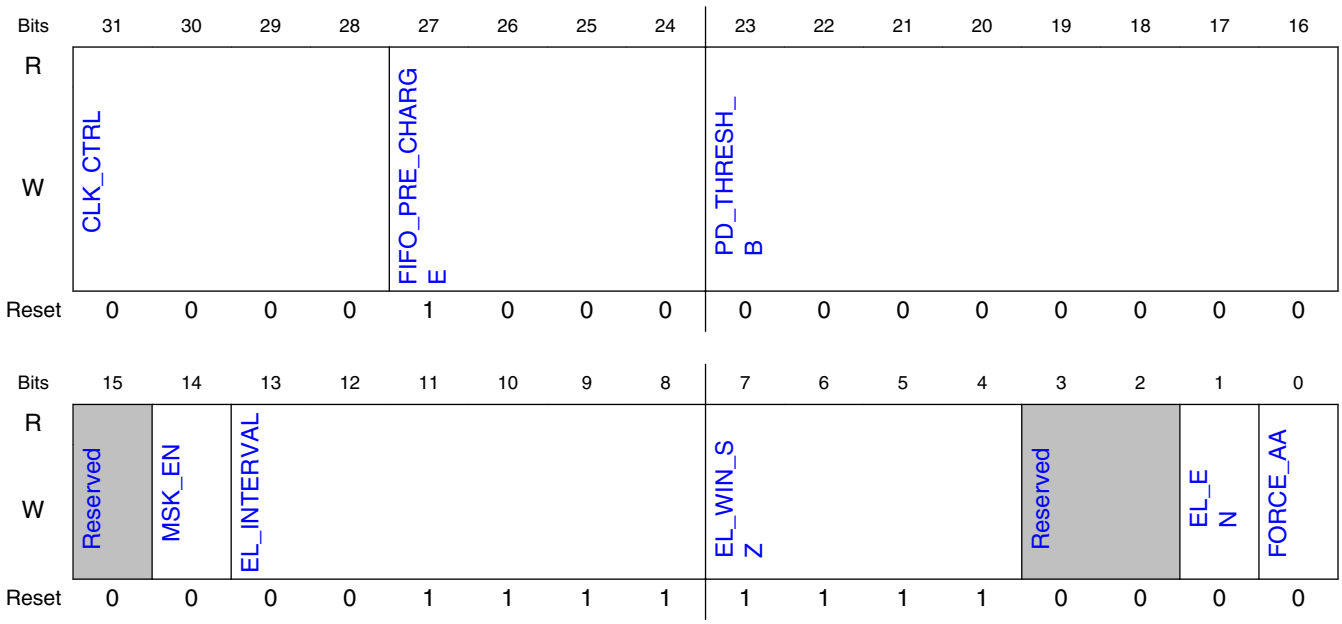
| Field | Function |
|---|---|
| 17-12<br>DEMOD_TIMEOUT | Number of symbols for which the state machine will wait past the expected end of the AA to declare a false PD and resume scanning. The actual timeout for the AA search is calculated as: TIMEOUT = (max_AA_length + DEMOD_TIMEOUT + 1.5) * OSR |
| 11<br>PDA_CONF_EN | Enables preamble AA confirmation for preamble detect mode A. |
| 10<br>PDB_CONF_EN | Enables preamble AA confirmation for preamble detect mode B. |
| 9-4<br>AA_CORR_GAIN | AA correlator gain. Format ufix6en3. This gain is applied to soft bits from the demodulator before they are used for address search synchronization. |
| 3<br>BSM_EN_BLE | Enable BSM mode for BLE. |
| 2<br>FSK_BIT_INVERT | Reserved.<br>0b - Normal demodulation.<br>1b - Invert demodulated bits. This applies at the demodulator, so it affects both AA and the data portions of the packet. |
| 1<br>AA_OUT_SEL | Reserved.<br>0b - When AA playback is enabled, play back desired AA.<br>1b - When AA playback is enabled, play back received AA. |
| 0<br>AA_PLAYBACK | Reserved.<br>0b - PHY will only output bits after the AA.<br>1b - PHY will output the AA, followed by the rest of the packet bits. |

## 64.3.10.3.5 PHY Misc. Configuration (PHY_FSK_MISC)

### 64.3.10.3.5.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_MISC | Ch |

### 64.3.10.3.5.2 Diagram




### 64.3.10.3.5.3 Fields

| Field | Function |
|---|---|
| 31-28<br>CLK_CTRL | Enables various clock gating features. Bits are individually decoded, so any combination is allowable.<br>0001b - Gate off PHY clock when phy_en is not asserted.<br>0010b - Gate off preamble detect clock when pd_enable is not asserted (internal signal).<br>0100b - Gate off AA synchronizer clock when synchronizer is not in use.<br>1000b - Gate off demodulator clock when demodulator is not in use. |
| 27-24<br>FIFO_PRE_CHARGE | Number of FIFO entries to load before sending bits out of the PHY. |
| 23-16<br>PD_THRESH_B | Preamble detect correlation threshold (ufix8en8). This threshold is used to define high correlation for the preamble detect logic. It is the threshold used for PD MODE B. |
| 15<br>— | Reserved. |
| 14<br>MSK_EN | Configures the PHY for MSK decoding. |
| 13-8<br>EL_INTERVAL | Number of symbols between successive EL operation windows. |
| 7-4<br>EL_WIN_SZ | Number of successive symbols over which one EL operation occurs. |
| 3-2<br>— | Reserved. |
| 1 | Enable Early-Late bit timing logic. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| EL_EN | |
| 0<br>FORCE_AA | Forces PHY to assume an AA match has occurred. For debug use only. |

## 64.3.10.3.6 PHY BSM Network Address (NTW_ADR_BSM)

### 64.3.10.3.6.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_BSM | 10h |

### 64.3.10.3.6.2 Function

Address used when BLE BSM mode is enabled.

### 64.3.10.3.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_BSM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_BSM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.3.6.4 Fields

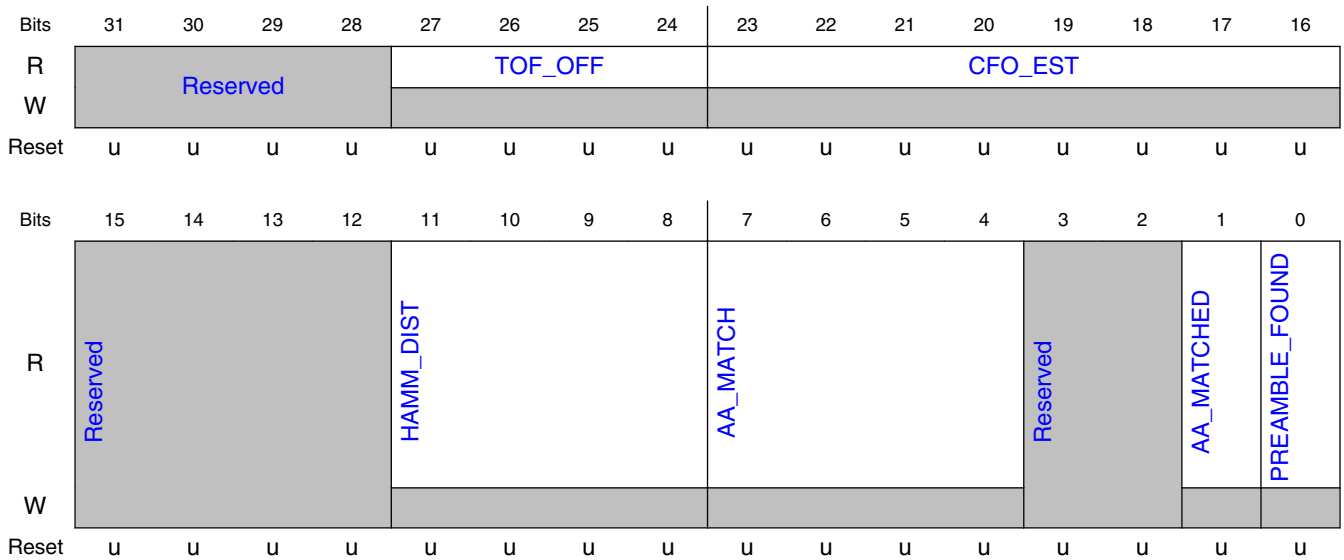
| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_BSM | Address used when BLE BSM mode is enabled. |

## 64.3.10.3.7 PHY Status (FSK_STAT)

### 64.3.10.3.7.1 Offset

| Register | Offset |
|---|---|
| FSK_STAT | 14h |

### 64.3.10.3.7.2 Diagram




### 64.3.10.3.7.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-24<br>TOF_OFF | Timing offset for use in time-of-flight calculation. |
| 23-16<br>CFO_EST | Indicates the currently estimated CFO. Format is ufix7en9 and can be estimated by: CFO = (CFO_EST/1024)*(OSR*BIT_RATE) |
| 15-12<br>— | Reserved. |
| 11-8<br>HAMM_DIST | Indicates the hamming distance witnessed when AA match occurred. |
| 7-4<br>AA_MATCH | Indicates which AA has matched. This will clear when the PHY is re-initialized. |
| 3-2<br>— | Reserved. |

*Table continues on the next page...*

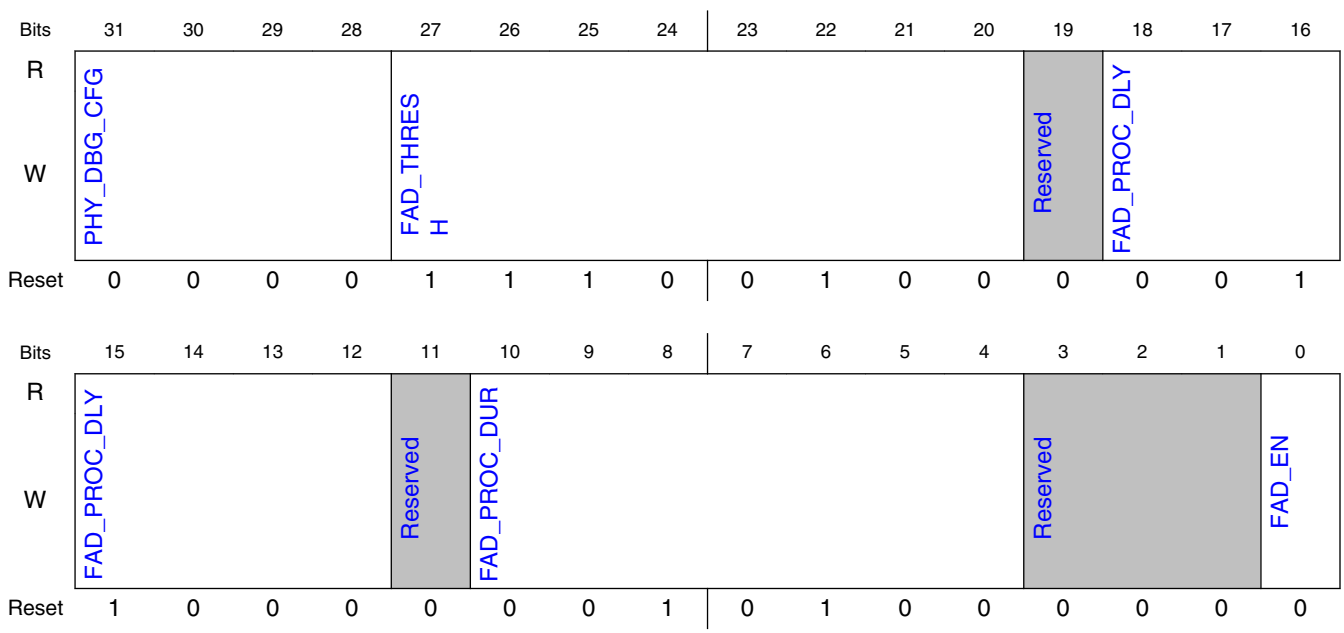
| Field | Function |
|---|---|
| 1<br>AA_MATCHED | Indicates that an AA match has occurred. This will only clear when the PHY is re-intialized. |
| 0<br>PREAMBLE_FOUND | Indicates that a preamble detect is active. |

## 64.3.10.3.8 PHY FAD control (FSK_FAD_CTRL)

### 64.3.10.3.8.1 Offset

| Register | Offset |
|---|---|
| FSK_FAD_CTRL | 18h |

### 64.3.10.3.8.2 Diagram




### 64.3.10.3.8.3 Fields

| Field | Function |
|---|---|
| 31-28<br>PHY_DBG_CFG | Individual bits for debug features. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 27-20<br>FAD_THRESH | Threshold used to compare correlator outputs when scanning. |
| 19<br>— | Reserved. |
| 18-12<br>FAD_PROC_DLY | Time for FAD to stay in WAIT state (see FAD description). |
| 11<br>— | Reserved. |
| 10-4<br>FAD_PROC_DUR | Cumulative time for WAIT and PROC2 states in FAD engine (see FAD description). |
| 3-1<br>— | Reserved. |
| 0<br>FAD_EN | Enable Fast-Antenna-Diversity |

## 64.3.10.4 XCVR_ZBDEMOD register descriptions

### 64.3.10.4.1 XCVR_ZBDEMOD_ADDR Memory map

XCVR_ZBDEM base address: 4103_0480h

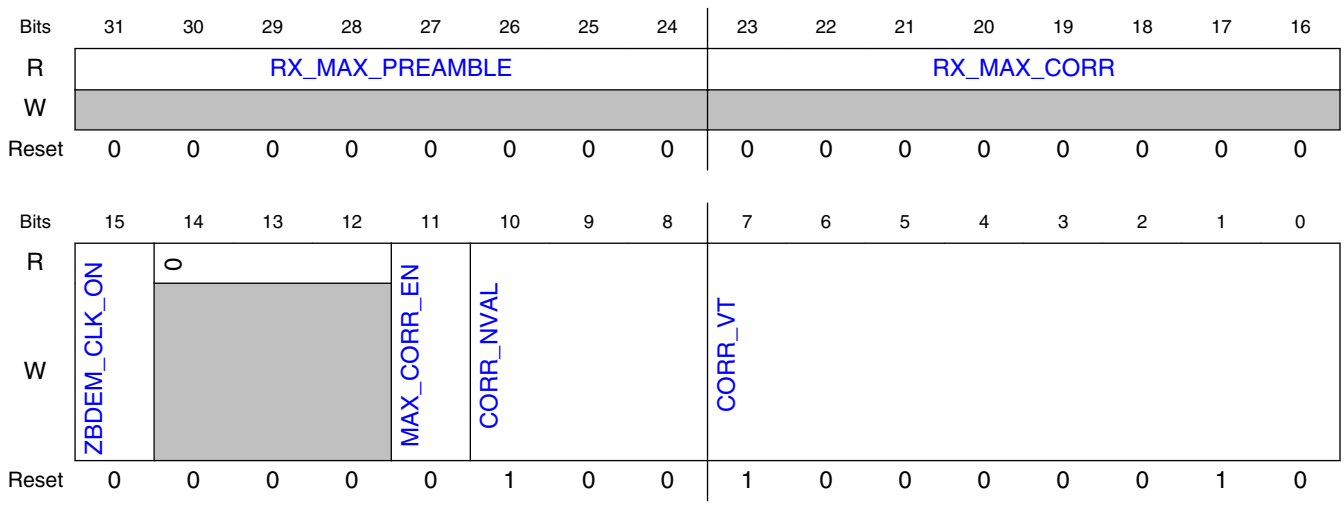
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | 802.15.4 DEMOD CORRELATOR CONTROL (CORR_CTRL) | 32 | RW | 0000_0482h |
| 4h | 802.15.4 DEMOD PN TYPE (PN_TYPE) | 32 | RW | 0000_0001h |
| 8h | 802.15.4 DEMOD PN CODE (PN_CODE) | 32 | RW | 744A_C39Bh |
| Ch | 802.15.4 DEMOD SYMBOL SYNC CONTROL (SYNC_CTRL) | 32 | RW | 0000_0008h |
| 10h | 802.15.4 CCA/LQI SOURCE (CCA_LQI_SRC) | 32 | RW | 0000_000Ch |
| 14h | FAD CORRELATOR THRESHOLD (FAD_LPPS_THR) | 32 | RW | 6060_6082h |
| 18h | 802.15.4 AFC STATUS (ZBDEM_AFC) | 32 | RW | 0000_0001h |
| 1Ch | CCA MODE 2 CONTROL REGISTER (CCA2_CTRL) | 32 | RW | 0000_FF05h |
| 20h | CCA MODE 2 CONTROL REGISTER (CCA2_THRESH) | 32 | RW | 0040_0080h |
| 24h | CCA MODE 2 STATUS REGISTER (CCA2_STATUS) | 32 | RO | 0000_0000h |

# 64.3.10.4.2 802.15.4 DEMOD CORRELATOR CONTROL (CORR_CTRL)

## 64.3.10.4.2.1 Offset

| Register | Offset |
|---|---|
| CORR_CTRL | 0h |

## 64.3.10.4.2.2 Diagram




## 64.3.10.4.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_MAX_PREAMBLE | RX_MAX_PREAMBLE<br>Max correlator during preamble-- max correlator value found during the preamble. |
| 23-16<br>RX_MAX_CORR | RX_MAX_CORR<br>Max correlator after preamble-- max correlator value found in packet after the preamble (refreshed every symbol rate if MAX_CORR_EN=1). |
| 15<br>ZBDEM_CLK_ON | Force 802.15.4 Demodulator Clock On<br>0b - Normal Operation<br>1b - Force 802.15.4 Demodulator Clock On (debug purposes only) |
| 14-12<br>— | Reserved |
| 11<br>MAX_CORR_EN | MAX_CORR_EN<br>Max correlator after preamble enable-- Enable the refresh of the max corr register |
| 10-8 | CORR_NVAL |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CORR_NVAL | Number of consecutively detected zero-symbols required to declare a preamble detected |
| 7-0<br>CORR_VT | CORR_VT<br>Correlator threshold, defines the sensitivity of demod during the preamble search state |

## 64.3.10.4.3 802.15.4 DEMOD PN TYPE (PN_TYPE)

### 64.3.10.4.3.1 Offset

| Register | Offset |
|---|---|
| PN_TYPE | 4h |

### 64.3.10.4.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TX_INV | PN_TYPE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 64.3.10.4.3.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>TX_INV | TX_INV<br>test mode to invert the transmission |
| 0<br>PN_TYPE | PN_TYPE<br>PN Type - Pseudo Noise Chip Code Type (for 802.15.4 = 1) |

## 64.3.10.4.4 802.15.4 DEMOD PN CODE (PN_CODE)

### 64.3.10.4.4.1 Offset

| Register | Offset |
|---|---|
| PN_CODE | 8h |

### 64.3.10.4.4.2 Function

Pseudo Noise Chip Code Seed Value

### 64.3.10.4.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PN_MSB | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PN_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

### 64.3.10.4.4.4 Fields

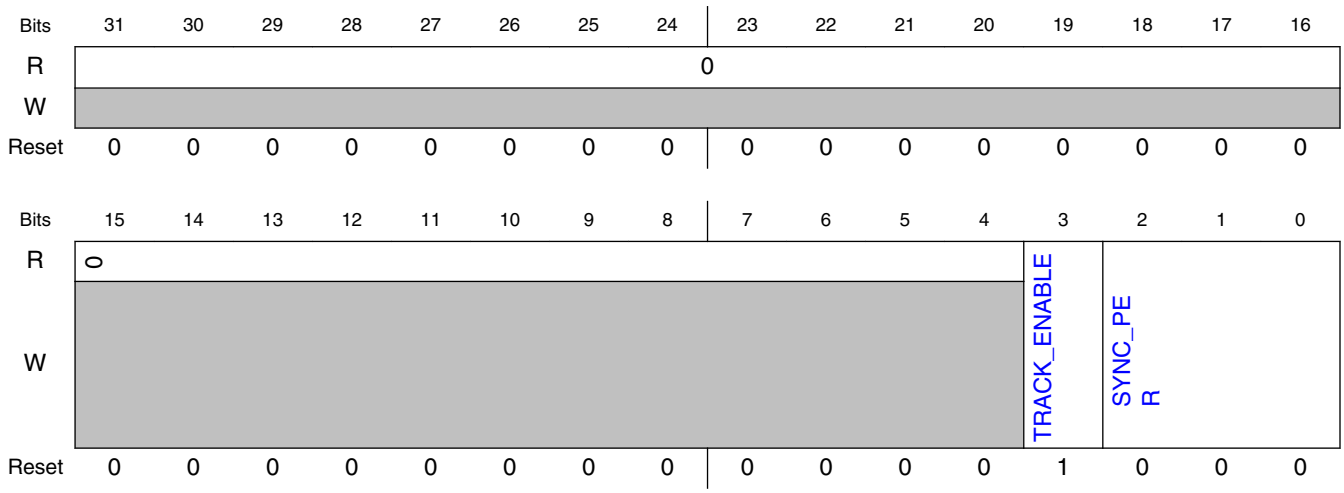
| Field | Function |
|---|---|
| 31-16<br>PN_MSB | PN_MSB<br>PN_CODE MS half |
| 15-0<br>PN_LSB | PN_LSB<br>PN_CODE LS half |

## 64.3.10.4.5 802.15.4 DEMOD SYMBOL SYNC CONTROL (SYNC_CTRL)

### 64.3.10.4.5.1 Offset

| Register | Offset |
|---|---|
| SYNC_CTRL | Ch |

### 64.3.10.4.5.2 Diagram




### 64.3.10.4.5.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>TRACK_ENABLE | TRACK_ENABLE<br>0b - symbol timing synchronization tracking disabled in Rx frontend<br>1b - symbol timing synchronization tracking enabled in Rx frontend (default) |
| 2-0<br>SYNC_PER | Symbol Sync Tracking Period<br>determines update rate for symbol timing, per equation. An early/late measurement is made every 2^SYNC_PER[2:0] symbols. Valid range of SYNC_PER[2:0] is 0 to 4. |

## 64.3.10.4.6 802.15.4 CCA/LQI SOURCE (CCA_LQI_SRC)

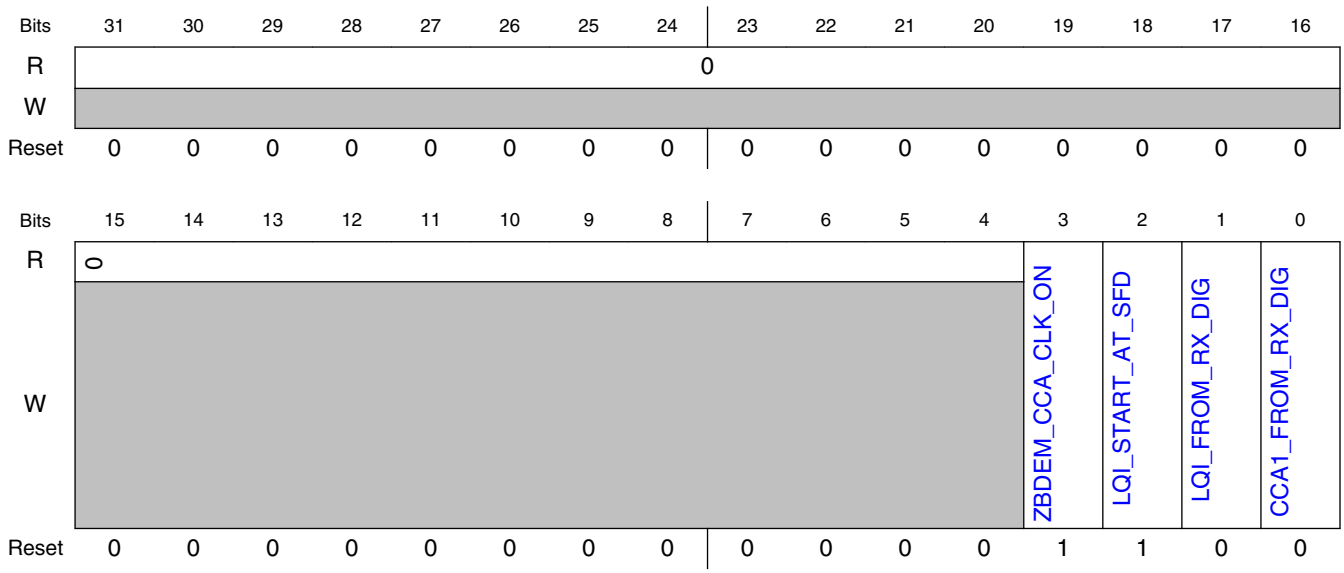
### 64.3.10.4.6.1 Offset

| Register | Offset |
|---|---|
| CCA_LQI_SRC | 10h |

### 64.3.10.4.6.2 Function

Selects the Source of CCA and LQI Information Provided to the 802.15.4 Link Layer

### 64.3.10.4.6.3 Diagram




### 64.3.10.4.6.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>ZBDEM_CCA_CLK_ON | 802.15.4 Demodulator CCA Clock Enable<br>802.15.4 Demodulator CCA Clock Enable. This bit should be left in its default state (1) if any CCA Mode (Modes 1, 2, and 3), Energy Detect, or LQI uses the 802.15.4 Demodulator to compute its value. If 802.15.4 Demodulator is not used to compute any of these, this bit can be cleared to save power. |
| 2<br>LQI_START_AT_SFD | Select Start Point for LQI Computation<br>0b - Start LQI computation at Preamble Detection (similar to previous NXP 802.15.4 products)<br>1b - Start LQI computation at SFD (Start of Frame Delimiter) Detection |
| 1<br>LQI_FROM_RX_DIG | Selects the Source of LQI (Link Quality Indicator) Information Provided to the 802.15.4 Link Layer<br>0b - Use the LQI information computed internally in the 802.15.4 Demod<br>1b - Use the LQI information computed by the RX Digital |
| 0<br>CCA1_FROM_RX_DIG | Selects the Source of CCA1 (Clear Channel Assessment Mode 1) Information Provided to the 802.15.4 Link Layer<br>0b - Use the CCA1 information computed internally in the 802.15.4 Demod<br>1b - Use the CCA1 information computed by the RX Digital |

## 64.3.10.4.7 FAD CORRELATOR THRESHOLD (FAD_LPPS_THR)

### 64.3.10.4.7.1 Offset

| Register | Offset |
|---|---|
| FAD_LPPS_THR | 14h |

### 64.3.10.4.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | LPPS_LP_EN_COUNT | | | | | | | 0 | LPPS_FILL_COUNT | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FAD_FILL1 | | | | | | | FAD_THR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

### 64.3.10.4.7.3 Fields

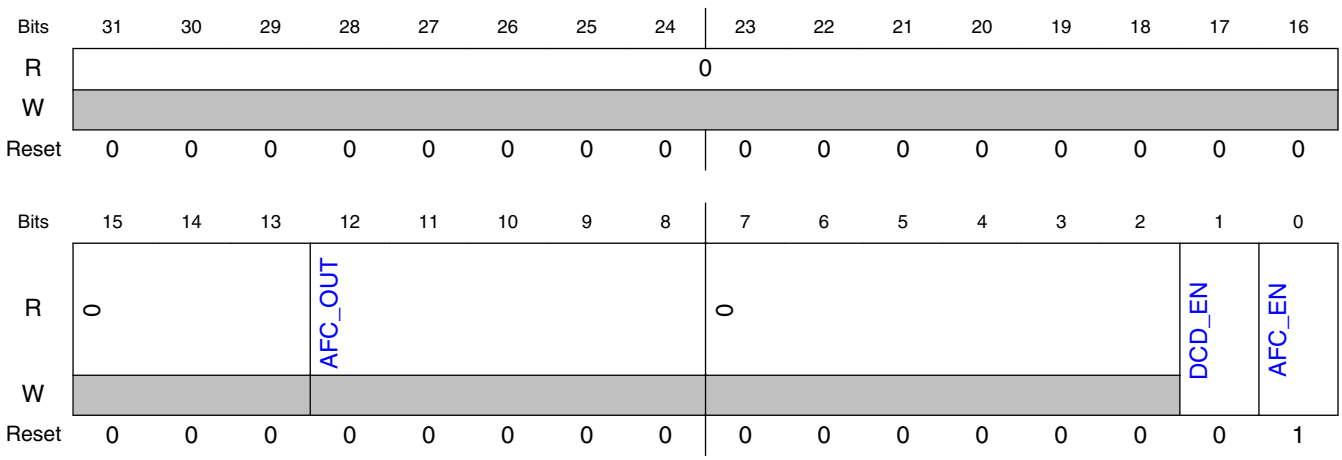
| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-24<br>LPPS_LP_EN_COUNT | LPPS_LP_EN high time<br>Parameter used to configure the duration of time lpps_lp_enable is in high state |
| 23<br>— | Reserved |
| 22-16<br>LPPS_FILL_COUNT | Wait duration after lpps_lp_enable is de-asserted<br>Parameter used to configure the wait duration used for valid samples acquisition after *lpps_lp_enable* is de-asserted. The overall wait duration expressed in base-band samples is determined using formula:<br>*lpps_fill_duration=lpps_fill_count+lpps_tsm_warmup_duration*<br>where *lpps_tsm_warmup_duration* is the duration of the fast TSM warmup, used for the LPPS mode, expressed in samples. |
| 15<br>— | Reserved |
| 14-8<br>FAD_FILL1 | Pre-detection buffer filling duration<br>Interval used to wait for valid samples acquisition after an antenna switch command has been issued. |
| 7-0<br>FAD_THR | FAD_THR<br>Correlator threshold at which the FAD will select the antenna. |

# 64.3.10.4.8 802.15.4 AFC STATUS (ZBDEM_AFC)

## 64.3.10.4.8.1 Offset

| Register | Offset |
|---|---|
| ZBDEM_AFC | 18h |

## 64.3.10.4.8.2 Diagram




## 64.3.10.4.8.3 Fields

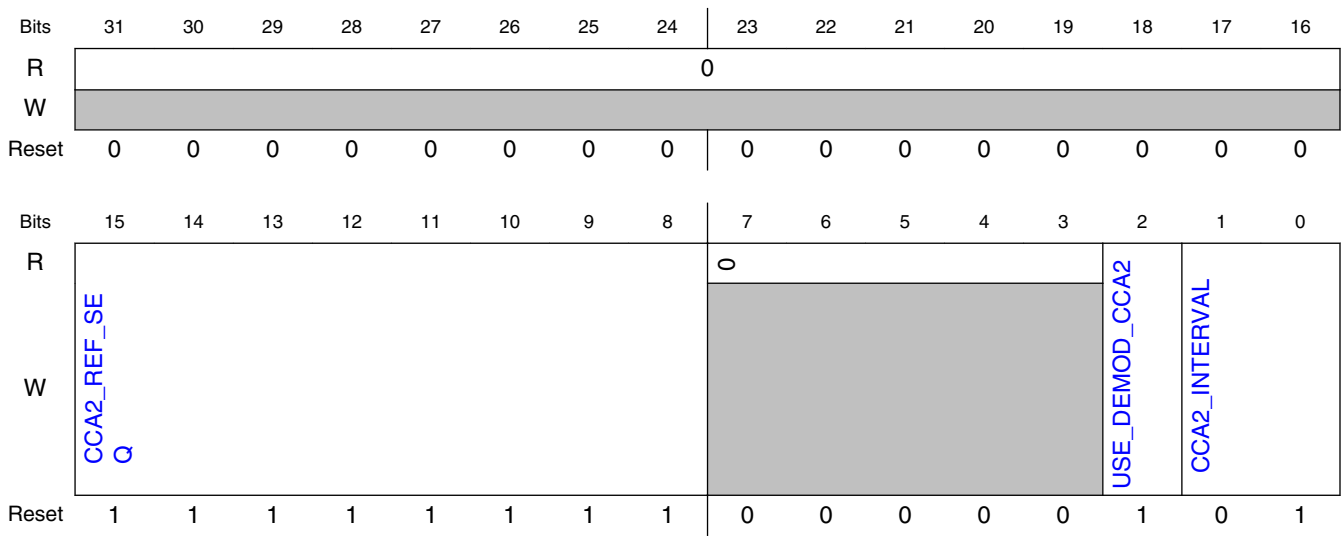
| Field | Function |
|---|---|
| 31-13<br>— | Reserved |
| 12-8<br>AFC_OUT | AFC_OUT<br>AFC Result from the last received packet. Format is Signed, Two's Complement. Each LSB represents 15KHz of Frequency Offset |
| 7-2<br>— | Reserved |
| 1<br>DCD_EN | DCD_EN<br>0b - NCD Mode (default)<br>1b - DCD Mode |
| 0<br>AFC_EN | AFC_EN<br>the AFC Function<br>0b - AFC is disabled<br>1b - AFC is enabled |

# 64.3.10.4.9 CCA MODE 2 CONTROL REGISTER (CCA2_CTRL)

## 64.3.10.4.9.1 Offset

| Register | Offset |
|---|---|
| CCA2_CTRL | 1Ch |

## 64.3.10.4.9.2 Diagram




## 64.3.10.4.9.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CCA2_REF_SEQ | CCA Mode 2 Sequence Address<br>Selection of the sequence addresses to be used. |
| 7-3<br>— | Reserved |
| 2<br>USE_DEMOD_CCA2 | Selects CCA Mode 2 Computation Engine<br>Selects between demodulator-based or standalone CCA Mode 2 Computation Engine<br>0b - Use standalone (new) CCA Mode 2 Engine, decoupled from demodulator<br>1b - Use 802.15.4 demodulator-based (legacy) CCA Mode 2 Engine (default) |
| 1-0 | CCA Mode 2 Measurement Window Duration<br>Specifies the CCA2 measurement window duration: |

| Field | Function |
|---|---|
| CCA2_INTERVAL | 00b - 64 µs<br>01b - 128 µs<br>10b - 256 µs<br>11b - 512 µs |

## 64.3.10.4.10 CCA MODE 2 CONTROL REGISTER (CCA2_THRESH)

### 64.3.10.4.10.1 Offset

| Register | Offset |
|---|---|
| CCA2_THRESH | 20h |

### 64.3.10.4.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CCA2_SYM_THRESH | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CCA2_CNT_THRESH | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 64.3.10.4.10.3 Fields

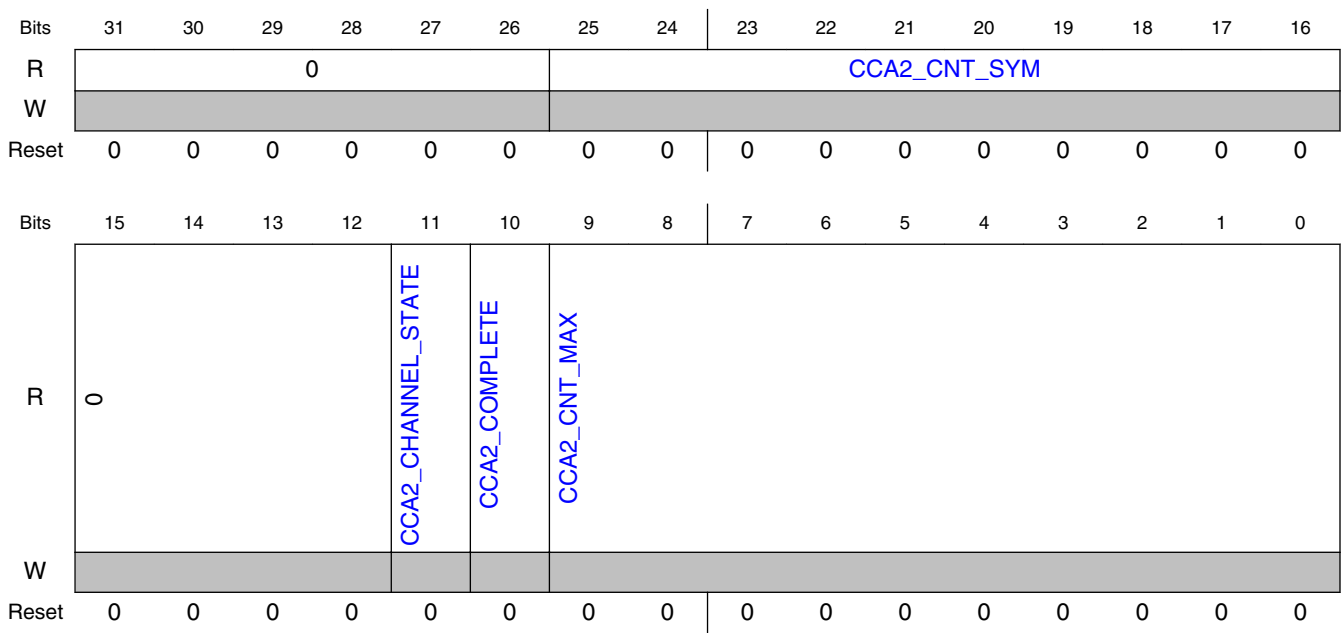
| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>CCA2_SYM_THRESH | CCA Mode 2 Symbol Threshold<br>Threshold used to detect repetition sequence addresses corresponding to correlation products that exceed CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |
| 15-10<br>— | Reserved |
| 9-0<br>CCA2_CNT_THRESH | CCA Mode 2 Count Threshold<br>Threshold to compare count of correlation products exceeding CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |

# 64.3.10.4.11 CCA MODE 2 STATUS REGISTER (CCA2_STATUS)

## 64.3.10.4.11.1 Offset

| Register | Offset |
|---|---|
| CCA2_STATUS | 24h |

## 64.3.10.4.11.2 Diagram




## 64.3.10.4.11.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>CCA2_CNT_SYM | CCA Mode 2 Repetition Sequence Addresses Count<br>Actual count of repetition sequence addresses corresponding to correlation products that exceed CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |
| 15-12<br>— | Reserved |
| 11 | CCA Mode 2 Channel State |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CCA2_CHANNEL_STATE | Indicates whether the channel is idle or busy; 0=idle, 1=busy. Asserted in any of the following cases:<br>1. CCA mode 2 measurement finds the channel busy when CCA2 duration expires<br>2. *rx_preamble_det* signal is asserted (note that, following this event, if *rx_preamble_det* is deasserted and *zb_cca2_complete* is low, *zb_cca2_channel_state* is deasserted also)<br>3. *rx_sfd_detect* signal is asserted |
| 10<br>CCA2_COMPLETE | CCA Mode 2 Measurement Complete<br>Indicates when a CCA mode 2 measurement is complete. Asserted in any of the following cases:<br>1. CCA mode 2 measurement is completed (when CCA2 duration has expired)<br>2. rx_sfd_detect signal is asserted |
| 9-0<br>CCA2_CNT_MAX | CCA Mode 2 Maximum Count<br>Actual count of correlation products exceeding CCA2_CORR_THRESH[7:0] during the measurement interval.<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |

# 64.4 Transceiver Analog

## 64.4.1 Introduction

The 2.4 GHz radio is comprised of both an analog and digital portion. The analog portion of the radio provides the local oscillator (LO), transmit modulation, and receiver down conversion, along with the various circuits needed to support those functions.

The analog block also provides a clock to the digital domain that is to be used as a reference. This clock is derived from an external reference and, in the case of a crystal, should be trimmed to be as accurate as possible. The two supported frequencies for this reference are 26 MHz and 32 MHz.

RF input and output are combined with an integrated balun located on the device. Thus, only a single RF port is needed.

It is also of note that the analog contains several LDO circuits to supply various blocks within the design.

The analog radio is configured and commanded by the digital domain using both static configuration signals and dynamic control signals.

### 64.4.1.1 Features

The 2.4 GHz analog radio block includes the following features:

- Fractional-N Frequency Synthesizer PLL shared by receive and transmit

- Integrated balun
- Transmitter supporting constant envelope modulation for 2.4 GHz ISM and 2.36 GHz MBAN frequency bands (2360 to 2483.5 MHz)
- Direct PLL modulation for transmit
- Direct conversion reciever for 2.4 GHz ISM and 2.36 GHz MBAN frequency bands (2360 to 2483.5 MHz)
- Receive line-up comrised of a Low Noise Amplifier & Mixer, Baseband amplifier and filter, and sigma-delta ADC
- Direct analog diagnostic lines for debug and test
- Multiple LDOs to power analog circuits from line supply
- XO (REF_OSC) ciruit to use and external crystal and provide a reference clock internally at 26 MHz or 32 MHz
- AuxPLL to provide a 2x reference clock
- DC correction DACs for the TZA and BBA stages
- Programmable receiver gains in both the LNA and BBA stages

### 64.4.1.2 Block diagram

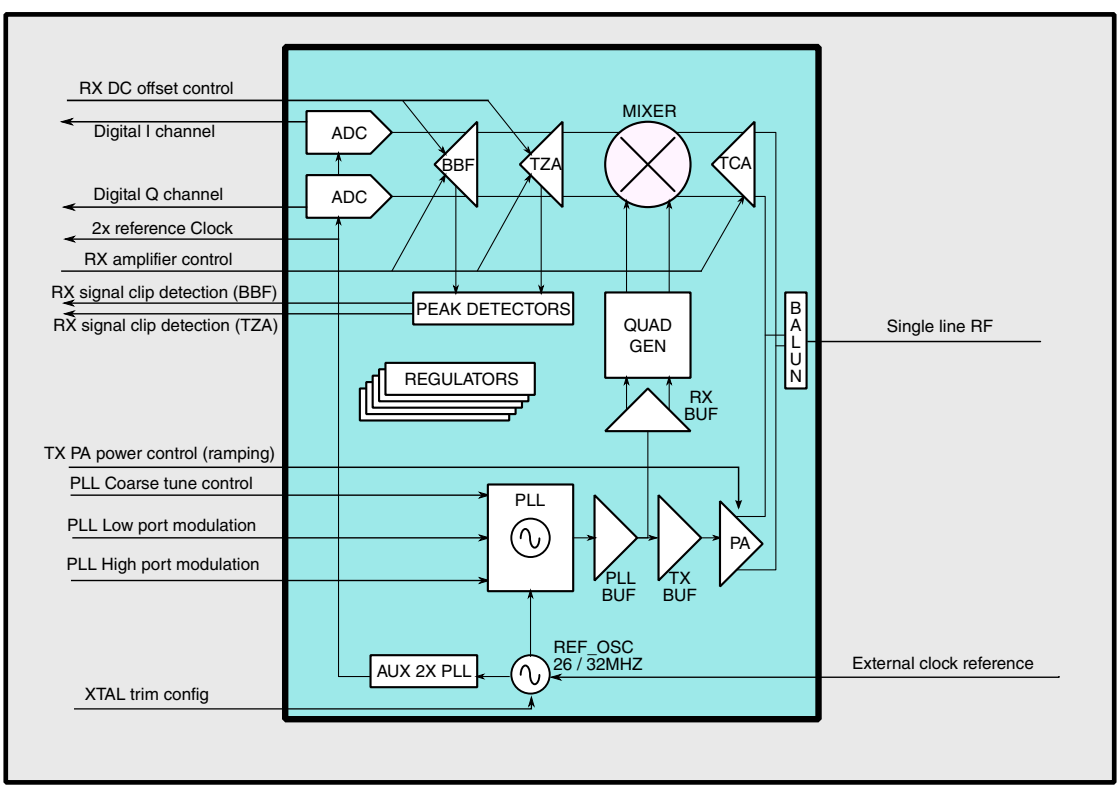



**Figure 64-136. Analog Radio Block diagram**

## 64.4.2 Functional description

The following section briefly describes the functionality of the 2p4GHz analog radio. It is not intended to be a complete description as many of the details are covered by descriptions of the control signals that are provided from the digital domain.

The analog block serves several purposes. Primarily, it provides the necessary circuitry to transmit at 2.4 GHz and receive in the same bands. When configured for transmission, the PLL is locked to the desired frequency (band and channel) as controlled by the PLL digital module. The digital radio then modulates the high port and low port of the PLL to generate the desired RF transmission. For receiver operation, the PLL is locked to the desired band and channel just as in transmit. The mixer provides a down-converted signal at baseband. The signal then goes through the baseband amplifier and is presented to the ADC. The ADC then converts the analog signal to a digital format, where it is observed by the digital domain. The receiver uses a 2x frequency clock to convert, so this clock is also provided to the digital domain from the AuxPLL. The receive path also contains several configurable parameters. For the gain, the LNA and BBA stages can be adjusted by signals from the digital domain. This is typically performed by the AGC circuit. DC offset can be corrected by controlling offset DACs at both stages, as well.

Both transmit and receive utilize an integrated balun. This results in a single RF port for the device and eliminates the need for a board level balun.

Several LDO blocks within the analog block provide the necessary supply to corresponding circuits. There is an LDO for the BBA, etc.

The XO block uses a reference crystal (typically) to produce a stable reference clock for the analog blocks and the digital domain. More details are provided in the section describing this block.

To assist in measurements and debug, multiple diagnostic analog signals are provided. These signals can be configured by inputs from the digital domain and then observed externally or with an ADC resident in the device.

### 64.4.2.1 XO Block

The XO block generates a 26 MHz or a 32 MHz reference clock for the digital domain, PLL, DAC, and AuxPLL blocks of the radio. It generates the output clocks from a crystal mounted externally to the device.

This block is alternatively known as REF_OSC, OSC, or XTAL, ocassionally.

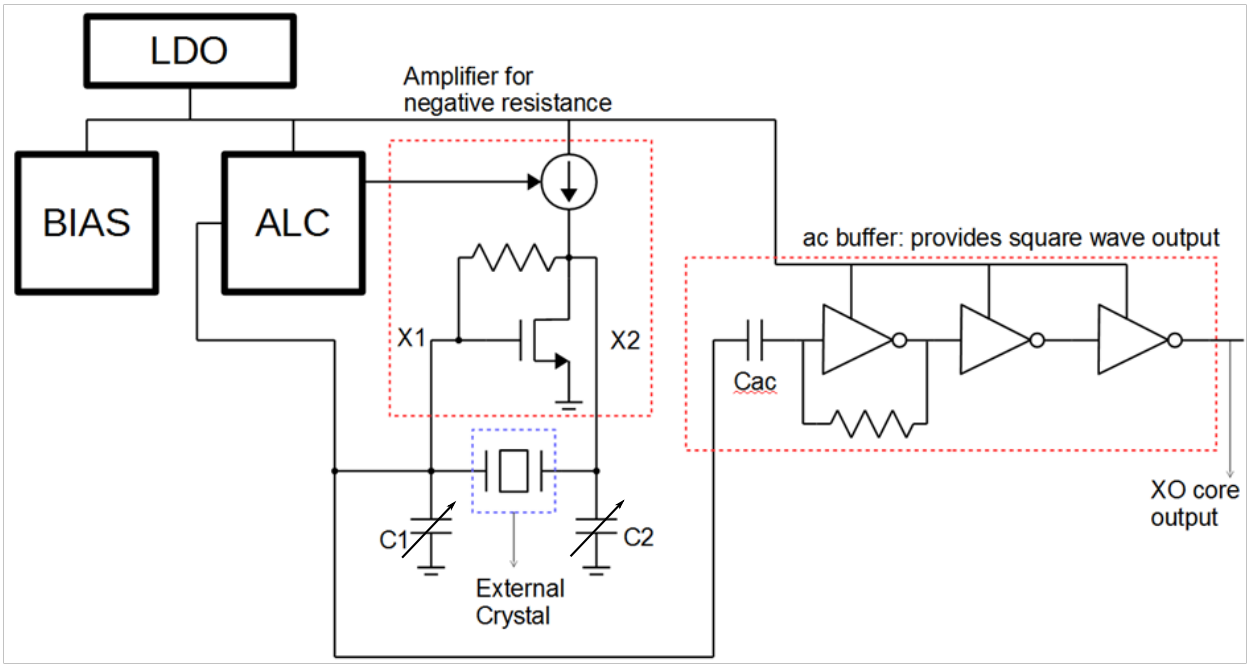



**Figure 64-137. XO circuit diagram**

#### 64.4.2.1.1 XTAL Trim

To enable optimum performance, it is required to trim the XO such that an accurate clock is produced based on the external crystal used. C1 and C2 are tuneable and intended for this purpose. They can be adjusted by configuring the XTAL_TRIM bits in the digital domain.

## 64.5 Link Layer

### 64.5.1 BLE Link Layer

#### 64.5.1.1 Register Descriptions

##### 64.5.1.1.1 Platform Registers

Platform Register Descriptions for the Bluetooth Link Layer

Instruction Register

| Addr | Addr | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | |
|---|---|---|---|---|---|
| 0x00 | 0x000 | COMMAND_REGISTER | WO | Instructions register to send commands to link layer hardware for controlling hardware operations. | 0xXX00 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:11 | | XX |
| conn_index[2:0] | 10:8 | Connection index to specify the command is for which connection engine. | 0 |
| command | 7:0 | 8-bit command from firmware to the link layer controller. See appendix 1 for the list of instructions and their opcodes. The instruction results in the link layer hardware starting/stopping an operation.<br>Notes on use<br>1. Few of the commands will require other configuration registers to be set, before the command is written. Refer to appendix 1 for details of the registers to be set before setting these instructions. | 00 |

## Event clear Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x04 | 0x008 | EVENT_CLEAR | WO | Clears the source of the interrupt. This register is written by firmware to clear interrupts set in the EVENT_STATUS register.<br>One or more interrupts can be cleared in a single write operation, by writing a 1 at the bit fields specific to the interrupts being cleared. It is not required to write a follow-up write with zero as the previous write is not actually stored. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Not used. | XX |
| Dtm_intr_clr | 8 | Clear DTM packet reception interrupt. Write to the register with this bit set to 1, clears the interrupt source. | |
| Reserved | 7:6 | Not used. | |
| Dsm_intr_clr | 5 | Clear deep sleep mode exit interrupt. Write to the register with this bit set to 1, clears the interrupt source. | |
| Sm_intr_clr | 4 | Clear sleep-mode-exit interrupt. Write to the register with this bit set to 1, clears the interrupt source. | |
| Reserved | 3:0 | Not used. | |

## Event status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x04 | 0x008 | EVENT_STATUS | RO | Event (Interrupt) status. Indicates pending events which require servicing by firmware. Each of the status bits is set by the link layer hardware. The bits are set till they are cleared by firmware by writing to appropriate event clear registers. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Not used. | XX |
| Dtm_intr | 8 | DTM packet reception interrupt. | 0 |
| Reserved | 7 | Not used. | XX |
| Enc_intr | 6 | Encryption module interrupt. | 0 |
| Dsm_intr | 5 | Deep sleep mode exit interrupt. This bit is set, when link layer hardware exits from deep sleep mode. The bit is cleared when firmware writes to EVENT_CLEAR register with this bit position set to 1. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Sm_intr | 4 | Sleep-mode-exit interrupt. This bit is set, when link layer hardware exits from sleep mode. The bit is cleared when firmware writes to EVENT_CLEAR register with this bit position set to 1. | 0 |
| Conn_intr | 3 | Connection interrupt. If bit is set to 1, it indicates an event occurred in the connection operation. This interrupt is aggregation of interrupts for all the connections. The source of the event for the specific connection needs to be read from the CONN_INTR_STATUS register specific to the connection. This bit is cleared, when firmware clears ALL interrupts by writing to the CONN_INTR_CLEAR register. | 0 |
| Init_intr | 2 | Initiator interrupt. If bit is set to 1, it indicates an event occurred in the initiating procedure. The source of the event needs to be read from the INIT_STATUS register. This bit is cleared, when firmware clears ALL interrupts by writing to the INIT_INTR_CLEAR register. | 0 |
| Scan_intr | 1 | Scanner interrupt. If bit is set to 1, it indicates an event occurred in the scanning procedure. The source of the event needs to be read from the SCAN_STATUS register. This bit is cleared, when firmware clears ALL interrupts by writing to the SCAN_INTR_CLEAR register. | 0 |
| Adv_intr | 0 | Advertiser interrupt. If bit is set to 1, it indicates an event occurred in the advertising procedure. The source of the event needs to be read from the ADV_STATUS register. This bit is cleared, when firmware clears ALL interrupts by writing to the ADV_INTR_CLEAR register. | 0 |

## Event enable register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x08 | 0x010 | EVENT_ENABLE | RW | Event indications enable. The register enables/masks each of the possible event sources from causing an interrupt. The bit fields mask only the events from interrupting the firmware. However hardware can still generate the interrupt. Firmware, when detects one interrupt, can mask all the interrupts temporarily, by clearing the register value to 0x0000. The interrupts can be masked till the first interrupt is processed and enable the interrupts back. This ensures no new interrupts is missed while firmware is processing one. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:7 | Not used. | XX |
| enc_int_en | 6 | Encryption module interrupt enable.<br>1 – Enable encryption module interrupt to firmware.<br>0 – disable encryption module interrupt to firmware. | 0 |
| Dsm_int_en | 5 | Deep Sleep-mode-exit interrupt enable.<br>1 – enable deep sleep mode exit event to interrupt the firmware.<br>0 – disable deep sleep mode exit interrupt to firmware. | 0 |
| Sm_int_en | 4 | Sleep-mode-exit interrupt enable.<br>1 – enable sleep mode exit event to interrupt the firmware.<br>0 – disable sleep mode exit interrupt to firmware. | 0 |
| Conn_int_en | 3 | Connection interrupt enable.<br>1 – enable connection procedure to interrupt the firmware. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | 0 – disable connection procedure interrupt to firmware. | |
| Init_int_en | 2 | Initiator interrupt enable.<br>1 – enable initiator procedure to interrupt the firmware.<br>0 – disable initiator procedure interrupt to firmware. | 0 |
| Scn_int_en | 1 | Scanner interrupt enable.<br>1 – enable scan procedure to interrupt the firmware.<br>0 – disable scan procedure interrupt to firmware. | 0 |
| Adv_int_en | 0 | Advertiser interrupt enable.<br>1 – enable advertiser procedure to interrupt the firmware.<br>0 – disable advertiser procedure interrupt to firmware. | 0 |

## Wakeup Configuration register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x5C | 0x0B8 | WAKEUP_CONFIG | RW | Wakeup configuration to configure the various offsets while waking up from sleep mode or deep sleep mode. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Dsm_offset_to_wakeup_instant | 15:10 | Number of “slots” before the wake up instant before which the hardware needs to exit from deep sleep mode. The slot is of 0.625ms period. This is a one-time configuration field, which is used every time hardware does an auto-wakeup before the next wakeup instant. | 0 |
| Sm_offset_to_wakeup_instant | 9 | Number of “slots” (1slot = 625 microseconds) before the wake up instant before which the hardware needs to exit from sleep mode. This is a onetime configuration field, | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | which is used every time hardware does an auto-wakeup before the next wakeup instant. | |
| Retain_in_dsm2 | 8 | “1” indicates Connection RAM to be retained during DSM2. | 0 |
| Reserved | 7:5 | Reserved for future use. | 0 |
| Osc_startup_delay[4:0] | 4:0 | Oscillator stabilization/startup delay. This is in X.Y format where X is in terms of number of BT slots (625 us) and Y is in terms of number of clock periods of 16KHz clock input, required for RF oscillator to stabilize the clock output to the controller on its output pin, after oscillator is turned ON. In this period the clock is assumed to be unstable, and so the controller does not turn on the clock to internal logic till this period is over. This means, the wake up from deep sleep mode must account for this delay before the wakeup instant.<br><br>Osc_startup_delay[4:0] is number of clock periods of 16KHz clock<br><br>(Warning: Min. value of Osc_startup_delay [4:0] supported is 1 and Max. value is 9. Therefore programmable range is 1 to 9) | 00 |

## Wakeup Configuration2 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x278 | 0x4F0 | WAKEUP_CONFIG2 | RW | Wakeup configuration to configure the various offsets while waking up from sleep mode or deep sleep mode. | 0x00 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:7 | Not used. | XX |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Osc_startup_delay[11:5] | 6:0 | Oscillator stabilization/startup delay. This is in X.Y format where X is in terms of number of BT slots (625 us) and Y is in terms of number of clock periods of 16KHz clock input, required for RF oscillator to stabilize the clock output to the controller on its output pin, after oscillator is turned ON. In this period the clock is assumed to be unstable, and so the controller does not turn on the clock to internal logic till this period is over. This means, the wake up from deep sleep mode must account for this delay before the wakeup instant.<br><br>`Osc_startup_delay[11:5] is number of slots(625us)` | 00 |

*Note: In case of firmware DSM exit mode: Exit from DSM shall be in synchronization with ref_clk (625 us slots timing) and shall not come out of DSM mode before meeting the oscillator start up delay.It is expected to have 0 to 625 us extra delay to wake-up from DSM, depends upon the assertion of the dsm_exit signal (firmware exit).*

## Sleep Threshold register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x5E | 0x0BC | SLEEP_THRESHOLD | RW | Sleep Threshold register. Stores threshold values for entering sleep mode or deep sleep mode. The threshold values are compared with the inactivity period to determine entry into one of the modes. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sm_threshold[3:0] | 15:12 | Number of inactive “slots” above which the device can enter sleep mode. If the number of slots is below this threshold, then device will not enter sleep mode. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Dsm_threshold[11:0] | 11:0 | Number of inactive “slots” above which the device can enter sleep mode. If the number of slots is below this threshold, then device will not enter deep sleep mode. | 0 |

*Note: Typically dsm_threshold > sm_threshold value. This means the following behavior is expected*

*No. of inactive slots(N) Behavior*

*N < sm_threshold No power save*

*Sm_threshold < N < dsm_threshold Sleep Mode*

*N > dsm_threshold Deep sleep mode*

Wakeup control register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x60 | 0x0C0 | WAKEUP_CONTROL | RW | Wakeup instant register. Program the 16-bit reference clock value for auto-wakeup from deep sleep mode. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Wakeup_instant[15:0] | 15:0 | Instant, with reference to the internal 16-bit clock reference, at which the hardware must wakeup from deep sleep mode. This is calculated by firmware based on the next closest instant where a controller operation is required (like advertiser/ scanner). Firmware reads the next instant of the procedures in the corresponding *_NEXT_INSTANT registers. This value is used only when hardware auto wakeup from deep sleep mode is enabled in the clock control register. | 0 |

*Note: it is recommended to program wakeup_instant such a way that the actual instant to wakeup shall be at least two counts (two slots of 625 us) ahead of reference clock when entering DSM. The actual instant to wakeup is "wakeup_instant – dsm_offset_to_wakeup_instant – osc_startup_delay, and it shall be greater than "reference clock + 2"*

## Clock control register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x62 | 0x0C4 | CLOCK_CONFIG | RW | Clock control and configuration. Controls clock gating and clock switch logic. | 0x0080 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Deep_sleep_mode_en | 15 | Enable deep sleep mode. 1 – enable, 0 – disable.<br>Enables hardware logic related to deep sleep mode to control the deep sleep mode operation. If disabled, the related logic is not executed and hardware cannot enter deep sleep mode. | 0 |
| Sleep_mode_en | 14 | Enable sleep mode. 1 – enable, 0 – disable.<br>Enables hardware to control sleep mode operation. | 0 |
| Dsm_intr_en | 13 | Enable DSM exit interrupt. 1 – enable, 0 – disable.<br>Enables hardware to generate an interrupt while exiting deep sleep mode. When enabled, interrupt is generated independent of whether exit procedure is initiated by hardware or firmware. | 0 |
| sm_intr_en | 12 | Enable SM exit interrupt. 1 – enable, 0 – disable.<br>Enables hardware to generate an interrupt while exiting sleep mode – irrespective of whether it is initiated by hardware or firmware. The interrupt is captured and stored till it gets cleared. | 0 |

*Table continues on the next page...*

| Field | Bit | Description | Reset |
|---|---|---|---|
| | | Disabling this bit mask the sleep mode exit event from hardware & firmware. | |
| Dsm_auto_sleep_en | 11 | Enable deep sleep mode auto entry in hardware.<br>1 – enable hardware to enter DSM automatically<br>0 – disable hardware to enter DSM automatically. | 0 |
| Sm_auto_wkup_en | 10 | Enable sleep mode auto wakeup enable. 1- enable, 0 – disable.<br>Enables hardware to automatically wakeup from sleep mode at the instant = *wakeup_instant – sm_offset_to_wakeup_instant.* The wakeup_insant is the field in the *wakeup control register* described earlier. The *sm_offset_to_wakeup_instant* value is the field described in the *wakeup configuration register.* | 0 |
| Lpo_sel_external | 9 | Select external sleep clock. 1 – External clock, 0 - internal generated clock.<br>The field is used to select either the low power clock input on sleep_clk input pin(of frequency 16.384KHz) directly to run the DSM logic or to use the internal generated reference clock(of 16KHz) for the same. | 0 |
| Lpo_clk_freq_sel | 8 | Clock frequency select. 0 – 32KHz, 1 – 32.768KHz.<br>Base frequency of the sleep_clk input used for generating the internal reference clock of approximate 16Khz frequency. | 0 |
| LLH_idle<br>(READ ONLY field) | 7 | Indicates if hardware is doing any transmit/receive operation or there is a pending interrupt from hardware. This information is used by firmware to decide to program the hardware into deep sleep mode.<br>1 – LL hardware is idle. | 1 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| | | 0 – LL hardware is busy. In this case LL hardware will not enter deep sleep mode, even if firmware gives an enter DSM command. (In this situation hardware generates dsm exit interrupt to inform firmware that DSM entry was not successful). | |
| Phy_clk_gate_en | 6 | Digital PHY clock enable. 1- enable, 0-disable.<br>Enable the Digital PHY to shutdown the clock. When 1, it indicates that controller has an upcoming activity so PHY clock must be turned ON. When 0, it indicates inactivity in the controller. This bit is not used as of now. | 0 |
| Sysclk_gate_en | 5 | Sysclk gate enable. 1- enable, 0 – disable.<br>Enables clock gating of system clock input to the link layer. If 1, it enables the DSM logic to control the clock gate for system clock input from pin. If 0, the DSM logic has no control and the system clock is always ON. | 0 |
| Coreclk_gate_en | 4 | Core clock gate enable. 1 – enable, 0 – disable.<br>Enables gating of clock to the llh_core module in hardware. If 1, the sleep mode/deep sleep mode logic can control the clock gate to shutdown/ wakeup the clock to the module. If 0, the logic has no control and clock is always turned ON. | 0 |
| Conn_clk_gate_en | 3 | Connection block clock gate enable. 1 – enable, 0 – disable.<br>Enables gating of clock to the connection module (llh_connch_top) in hardware. If 1, the sleep mode logic can control the clock gate to shutdown/wakeup the clock to | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | the engine. If 0, the logic has no control and clock to the module is always turned ON. | |
| Init_clk_gate_en | 2 | Initiator block clock gate enable. 1 – enable, 0 – disable.<br>Enables gating of clock to the initiator module (llh_init). If 1, the sleep mode logic can control the clock gate to shutdown/wakeup the clock to the module. If 0, the logic has no control and clock to the module is always turned ON. | 0 |
| Scan_clk_gate_en | 1 | Scan block clock gate enable. 1 – enable, 0 – disable.<br>Enables gating of clock to the scanner module (llh_scan) in hardware. If 1, the sleep mode logic can control the clock gate to shutdown/ wakeup the clock to the module. If 0, the logic has no control and clock to the module is always turned ON. | 0 |
| Adv_clk_gate_en | 0 | Advertiser block clock gate enable. 1 – enable, 0 – disable.<br>Enables gating of clock to the advertiser module (llh_adv) in hardware. If 1, the sleep mode logic can control the clock gate to shutdown/ wakeup the clock to the module. If 0, the logic has no control and clock to the module is always turned ON. | 0 |

## Reference Clock register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x64 | 0x0C8 | TIM_COUNTER_L | RO | 16-bit reference clock used for timing reference in the operation of the hardware. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Clock[15:0] | 15:0 | 16-bit internal reference clock. The clock is a free running clock, incremented by a | 0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| | | 0.625ms periodic pulse. It is used as a reference clock to derive all the timing required as per protocol. | |

## BLE Time Control

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X6C | 0XD8 | TIME_CONTROL | WO | LLH clock frequency configuration. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:8 | Not used | 0x0 |
| bb_clk_freq_minus_1 | 7:3 | The frequency information (FREQ – 1) of the clock input to LL Hardware is configured in this register by the firmware during initialization. This information is used inside LL Hardware to derive timing information for the Bluetooth operations.<br>For example, if the frequency of the input clock is 12 MHz this field shall be programmed to 0xB. (1 less than frequency) | 0x0 |
| RFU | 2:0 | Not used | 0x0 |

## DSM configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X51A | 0XA34 | DSM_CONFIG | RW | DSM configuration register. | 0x0008 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Not used | 15 | Not used | 0x0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| enable_conn_clk_ctrl[7:0] | [14:7] | Each bit corresponds to one connection engine (bit 7 for conn engine 0). Each of the bits if set to 1 will enable LLH to turn on corresponding CONNECTION engine clock on EXIT_SM command from firmware. To use this feature CON-NECTION NAP modes shall be enabled in LLH. | 0x0 |
| enable_init_clk_ctrl | 6 | If this bit is set to ‘1’ then LLH will turn on INIT engine clock on EXIT_SM command from firmware. To use this feature INIT NAP modes shall be enabled in LLH. | 0x0 |
| enable_scan_clk_ctrl | 5 | If this bit is set to ‘1’ then LLH will turn on SCAN engine clock on EXIT_SM command from firmware. To use this feature SACN NAP mode shall be enabled in LLH. | 0x0 |
| enable_adv_clk_ctrl | 4 | If this bit is set to ‘1’ then LLH will turn on ADV engine clock on EXIT_SM command from firmware. To use this feature ADV NAP mode shall be enabled in LLH. | 0x0 |
| enable_eng_clk_ctrl | 3 | Configuration bit for turning off all engine specific clocks after wakeup from SHUTDOWN (after register restore is complete). If this bit is set to ‘1’ then LLH will turn off all engine specific clocks after restore_done. To use this feature all NAP modes shall be enabled in LLH. | 0x1 |
| restore_done_bypass | 2 | Configuration bit for bypassing wait for restore_done signal to exit from RESTORE state. If this bit is set to ‘1’ then power control FSM will move to next state without waiting for restore_done from RESTORE state and if this bit is set to ‘0’ then power state machine will wait in RESTORE state till restore_done comes. This bit can be effectively used to reduce wakeup time if restore gets completed in one LP clock cycle. | 0x0 |
| store_done_bypass | 1 | Configuration bit for bypassing wait for store_done | 0x0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | signal to enter in to SHUTDOWN. If this bit is set to '1' then power control FSM will move to next state without waiting for store_done from STORE state and if this bit is set to '0' then power state machine will wait in STORE state till store_done comes. This bit can be effectively used to reduce SHUTDOWN entry time if store gets completed in one LP clock cycle. | |
| dsm_config | 0 | Configuration bit for DSM SHUTDOWN LLH store and restore control. If this bit is set to '1' then LLH will do the store and restore of its registers on its own during entry and after exit from shutdown. To use this feature DSM SHUTDOWN mode is to be supported and also LLH shall have access to RETENTION RAM. | 0x0 |

Note To enable dynamic power saving though selective clock gating of different engines LL firmware will set the configuration bits corresponding to BT procedures which are active. This will ensure that clocks to only active blocks are enabled during EXIT_SM command from LL firmware to LL hardware. User can over-ride this mode by setting the bits.

Nearest Instant register

| **Addr 16-bit** | **Addr 32-bit** | **Register Name** | **RW** | **Description** | **Reset** |
|---|---|---|---|---|---|
| 0X520 | 0XA40 | NEAREST_INST | RO | This register reports the next immediate instant in terms of bt_clock value (actual value being reported is INSTANT – 3) where hardware has any scheduled BT activity. Firmware reads this register along with the TIM_COUNTER_L register to calculate the possibility of power save mode entry. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| global_nearest_instant | 15:0 | This field holds the immediate future value of BT_CLOCK where a scheduled Bluetooth activity is supposed to happen.<br>When no Bluetooth activity is running the register will hold current BT_CLOCK value – 3. | 0x0000 |

## Auto DSM Configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X274 | 0X4E8 | AUTO_DSM_CONFIG | RW | Auto DSM configuration register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Not Used | 15:6 | Not Used | 0x0 |
| Auto_entry_fsm_store_val | 5:4 | Store/Restore of Auto entry FSM states. | 0x0 |
| Dms3_mode_en | 3 | Enable DSM3 mode for Auto DSM entry feature. 1 – enable, 0 – disable. | 0x0 |
| Dsm2_mode_en | 2 | Enable DSM2 mode for Auto DSM entry feature. 1 – enable, 0 – disable. | 0x0 |
| Fw_wakeup_en | 1 | Enable wakeup at every FW_WAKEUP_INSTANT by firmware when HW is in sleep mode. 1 – enable, 0 – disable. | 0x0 |
| Dsm_auto_sleep_en | 0 | Reserved | 0x0 |

## Firmware wakeup Instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X276 | 0X4EC | FW_WAKEUP_INSTANT | RW | Wakeup instant register. Program the 16-bit reference clock value for auto-wakeup from deep sleep mode. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Fw_wakeup_instant | 15:0 | This field holds the clock instant value when LLH need to exit auto dsm mode. | 0x0000 |

### 64.5.1.1.2 Advertising Channel Registers

Advertising Channel Register Descriptions for the Bluetooth Link Layer

##*Advertising parameters register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x0C | 0x018 | ADV_PARAMS | RW | Advertising parameters register. Firmware sets the necessary parameters for the advertising procedure into this register before issuing start advertise command. The fields in this register correspond to the fields in the LE_Set_Advertising_Parameters HCI command. | 0x00E0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| ##peer_rx_txaddr<br>(Read Only) | 15 | Transmit address field of the received packet indicating the peer address type as public/ random. This field is used by firmware to report peer_addr_type parameter in the connection complete event.<br>0 – addr type is public.<br>1 – addr type is random. | 0 |
| ##peer_addr_resolved_reg<br>(Read Only) | 14 | Address resolution status of the received packet if the received address was RPA.<br>0 – RPA resolution fail<br>1 – RPA resolution pass.<br>Else this bit is set to zero. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| #slv_index | 13:11 | Slave index is programmed by FW before initiation procedure for selecting connection engine in slave role connection.<br>Number of supported engines is 8. (Index from 0x0 to 0x7) | 0 |
| *adv_low_duty_cycle | 10 | This bit field is used to specify to the Controller the Low Duty Cycle connectable directed advertising variant being used.<br>1 – Low Duty Cycle Connectable Directed Advertising.<br>0 – High Duty Cycle Connectable Directed Advertising. | 0 |
| Force_scan_rsp<br>(Write Only) | 9 | Force scan response packet always. *Used only if TESTER build is enabled.*<br>• Override ADV packet type and send scan response packet type in the header field in all ADV packets<br>0 – no effect. | 0 |
| Rx_addr | 8 | Peer addresses type. This is the Direct_Address_type field programmed, only if ADV_DIRECT_IND type is sent.<br>• Rxaddr type is random.<br>0 - Rxaddr type is public. | 0 |
| Adv_channel_map | 7:5 | Advertising channel map indicates the advertising channels used for advertising. By setting the bit, corresponding channel is enabled for use. At least one channel bit should be set.<br>Bit 7- enable channel 39.<br>Bit 6 - enable channel 38.<br>Bit 5 - enable channel 37. | 7 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Adv_filt_policy | 4:3 | Advertising filter policy. The set of devices that the advertising procedure uses for device filtering is called the White List .<br>0x00 - Allow scan request from any device, allow connect request from any device.<br>0x01- Allow scan request from devices in white list only, allow connect request from any device.<br>0x10--Allow scan request from any device, allow connect request from devices in white list only.<br>0x11--Allow scan request from devices in white list only, allow connect request from devices in white list only. | 0 |
| Adv_type | 2:1 | The Advertising type is used to determine the packet type that is used for advertising when advertising is enabled.<br>00b- Connectable undirected advertising. (adv_ind)<br>01b- Connectable directed advertising (adv_direct_ind).<br>10b- Discoverable undirected advertising (adv_discover_ind)<br>11b- Non connectable undirected advertising (adv_nonconn_ind). | 0 |
| Reserved | 0 | Reserved for future use. | 0 |

## Advertising Interval register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x0E | 0x01C | ADV_INTERVAL_TIMEOUT | RW | Advertising interval register. It is the i nterval between two consecutive advertising events. For directed advertising, this register holds the timeout value.<br>Has a resolution of 0.625ms.<br>Time = N * 0.625 msec | 0x0020 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Time Range: 20 ms to 10.24 sec.<br>Firmware updates this value before issuing start advertise command. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Adv_interval | 15:0 | Range:<br>0x0020 to 0x4000 (For ADV_IND)<br>0x00A0 to 0x4000 (For ADV_SCAN_IND and NONCONN_IND)<br>For directed advertising, firmware programs the default value of 1.28seconds. | 0x0010 |

## Advertising interrupt clear register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x10 | 0x020 | ADV_INTR_CLEAR | WO | Clears the source of the interrupt. This register is written by firmware to clear interrupts set in the ADV_STATUS register. One or more interrupts can be cleared in a single write operation, by setting the bit field to 1 for corresponding interrupt. It is not required to write a follow-up write with bit 0, as the previous bit 1 is not actually stored. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Not used. | XX |
| adv_timeout | 7 | Clear adv_timeout interrupt. Applicable in ADV_DIRECT_IND advertising. Write to the register with this bit set to 1, clears the interrupt source. | 0 |
| slv_connected | 6 | Clear slave connected interrupt. Write to the register | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | with this bit set to 1, clears the interrupt source. | |
| conn_req_rx_intr | 5 | Clear connect request packet received interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| scan_req_rx_intr | 4 | Clear scan request packet received interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| scn_rsp_tx_intr | 3 | Clear scan response packet transmitted interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| adv_tx_intr | 2 | Clear adv packet transmitted interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| adv_close_intr | 1 | Clear advertising event stop interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| adv_strt_intr | 0 | Clear advertising event start interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |

## Advertising status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x10 | 0x020 | ADV_STATUS | RO | Advertising status register shows the status of the interrupts .Each of the status bits is set by the advertising procedure in hardware. The bits are set till they are cleared by firmware by writing to appropriate interrupt clear registers. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Not used. | XX |
| Adv_on | 8 | Advertiser procedure is ON in hardware. Indicates that advertiser procedure is ON in hardware.<br>1 – on | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | 0 – off | |
| adv_timeout | 7 | If this bit is set it indicates that the directed advertising event has timed out after 1.28 seconds. Applicable in adv_direct_ind advertising only. | 0 |
| slv_connected | 6 | If this bit is set it indicates that connection is created as slave. | 0 |
| conn_req_rx_intr | 5 | If this bit is set it indicates connect request packet is received. | 0 |
| scan_req_rx_intr | 4 | If this bit is set it indicates scan request packet received. | 0 |
| scn_rsp_tx_intr | 3 | If this bit is set it indicates scan response packet transmitted in response to previous scan request packet received. | 0 |
| adv_tx_intr | 2 | If this bit is set it indicates ADV packet is transmitted. | 0 |
| adv_close_intr | 1 | If this bit is set it indicates current advertising event is closed. | 0 |
| adv_strt_intr | 0 | If this bit is set it indicates a new advertising event started after interval expiry. | 0 |

## Advertising next instant register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x12 | 0x024 | ADV_NEXT_INSTANT | RO | Shows the instant at which the next advertising event begins. This is with reference to internal reference clock of resolution 625 us.. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| next_adv_instant | 15:0 | Shows the next start of advertising event with reference to the internal reference clock. | 0 |

## Scan Interval register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x14 | 0x028 | SCAN_INTERVAL | RW | Scan interval register. Interval between two consecutive scanning events. Firmware sets the scanning interval value to this register before issuing start scan command. | 0x0010 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| scan_interval | 15:0 | Interval between two consecutive scanning events.<br>Range: 0x0004 to 0x4000<br>Default: 0x0010 (10 ms)<br>Time = N * 0.625 msec<br>Time Range: 2.5 msec to 10.24 sec . | 0X0010 |

## Scan Window register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x16 | 0x02C | SCAN_WINDOW | RW | Scan window register. Duration of scan in a scanning event, which should be less than or equal to scan interval value . Firmware sets the scan window value to this register before issuing start scan command. | 0x0010 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| scan_window | 15:0 | Duration of scan in a scanning event, which should be less than or equal to scan interval value.<br>Range: 0x0004 to 0x4000<br>Default: 0x0010 (10 ms)<br>Time = N * 0.625 msec<br>Time Range: 2.5 msec to 10.24 sec . | 0X0010 |

## ##Scan parameters register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x18 | 0x030 | SCAN_PARAM | RW | Scanning parameters register. Firmware sets the necessary parameters for scanning procedure into this register before issuing start scan command. The fields are derived from the LE_Set_Scan_Parameters HCI command. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:7 | Not used. | XX |
| scan_tx_addr[1] | 6 | MSB of the device's Own address. This bit along with scan_tx_addr[0] decides the own address used for scanner. | 0 |
| Dup_filt_en | 5 | Filter duplicate packets.<br>1 -Duplicate packet filtering enabled.<br>0- Duplicate packet filtering not enabled.<br>This field is derived from the *LE_set_scan_enable* command. | 0 |
| scan_filt_policy | 4:3 | The scanner filter policy determines how the scanner processes advertising packets .<br>0x00 – Accept advertising packets from any device. A connectable Directed advertising packet not containing the scanner's device address is ignored.<br>0x01 - Accept advertising packets from only devices in the whitelist . A connectable Directed advertising packet not containing the scanner's device address is ignored.<br>0x10 - Accept advertising packets from any device. A connectable Directed advertising packet is not ignored if the InitA is a resolvable private address. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | 0x11 - Accept advertising packets from only devices in the whitelist. A connectable Directed advertising packet is not ignored if the InitA is a resolvable private address.<br>Adv_direct_ind packets which are not addressed to this device are ignored. | |
| scan_type | 2:1 | 0x00- passive scanning. (default)<br>0x01- active scanning.<br>0x10- RFU<br>0x11- RFU | 0 |
| scan_tx_addr[0] | 0 | Device's own address type.<br>1 - addr type is random.<br>0 - addr type is public.<br>Note :<br>Scan_tx_addr[1:0]<br>0x0 – use public address<br>0x1 – use random address<br>0x2 – use RPA if matching entry is available in Resolving list else use public address.<br>0x3 - use RPA if matching entry is available in Resolving list else use random address. | 0 |

## Scan interrupt clear register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1C | 0x038 | SCAN_INTR_CLEAR | WO | Clears the source of the interrupt. This register is written by firmware to clear interrupts set in the SCAN_STATUS register. One or more interrupts can be cleared in a single write operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:5 | Not used. | XX |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| scan_rsp_rx_intr | 4 | Clear scan_rsp packet received interrupt. Write to the register with this bit set to 1, clears the interrupt source. | 0 |
| adv_rx_intr | 3 | Clear adv packet received interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| scan_tx_intr | 2 | Clear scan request packet transmitted interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| scan_close_intr | 1 | Clear scan event close interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| scan_strt_intr | 0 | Clear scan event start interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |

## Scan status register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1C | 0x038 | SCAN_STATUS | RO | Shows the status of the interrupt. This register is read by firmware to learn the pending scan interrupts that are set and are to be served. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 and 7:6 | Not used. | XX |
| Scan_on | 8 | Scan procedure is active. 1 – scan procedure is active. 0 – scan procedure is not active. | |
| scan_rsp_adv_rx_intr | 5 | If this bit is set after both SCAN_RSP and ADV packets are received. This interrupt may be enabled, if firmware desires to be interrupted after complete reception, instead of interrupt for each packet – which are enabled by bits 4 and 3. | 0 |
| scan_rsp_rx_intr | 4 | If this bit is set it indicates SCAN_RSP packet is received. Firmware can read the content of the packet from | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | the INIT_SCN_ADV_RX_FIFO. | |
| adv_rx_intr | 3 | If this bit is set it indicates ADV packet received. Firmware can read the content of the packet from the INIT_SCN_ADV_RX_FIFO. | 0 |
| scan_tx_intr | 2 | If this bit is set it indicates scan request packet is transmitted. | 0 |
| scan_close_intr | 1 | If this bit is set it indicates scan window is closed. | 0 |
| scan_strt_intr | 0 | If this bit is set it indicates scan window is opened. | 0 |

*Note:*

*scan_rsp_adv_rx_intr—This interrupt is generated while active scanning ,after receiving both the adv and scan response packets, in case of receiving adv_ind and adv_discover_ind. Currently not in use.*

*scan_rsp_rx_intr-- This interrupt is generated while active scanning upon receiving scan response packet.*

*adv_rx_intr-- This interrupt is generated while active/passive scanning upon receiving adv packets.*

## Scan next instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E | 0x03C | SCAN_NEXT_INSTANT | RO | Shows the instant w.r.t internal reference clock of resolution 625us at which next scanning event begins. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| next_scan_instant | 15:0 | Shows the instant w.r.t internal reference clock at which next scanning window begins. | 0 |

## Initiator Interval register

| Addr | Addr | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | |
|---|---|---|---|---|---|
| 0x20 | 0x040 | INIT_INTERVAL | RW | Initiator interval register. Firmware sets the initiator's scanning interval value to this register before issuing create connection command. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_scan_interval | 15:0 | Interval between two consecutive scanning events.<br>Range: 0x0004 to 0x4000<br>Time = N * 0.625 msec<br>Time Range: 2.5 msec to 10.24 sec . | 0X0000 |

## Initiator window register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x22 | 0x044 | INIT_WINDOW | RW | Initiator window register. Firmware sets the scan window value to this register before issuing the create connection command. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_scan_window | 15:0 | Duration of scan in a scanning event, which should be less than or equal to scan interval value.<br>Range: 0x0004 to 0x4000<br>Default: 0x0010 (10 ms)<br>Time = N * 0.625 msec<br>Time Range: 2.5 msec to 10.24 sec . | 0X0000 |

## ##Initiator parameter register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x24 | 0x048 | INIT_PARAM | RW | Initiator parameters register. Firmware sets the necessary parameters for initiation procedure to this register before issuing the create connection command. The fields in this register are derived from the parameters in the LE_Create_Connection HCI command. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:10 | Not used. | XX |
| peer_rpa_resolved_capt<br>(Read only) | 9 | If received peer address is an RPA, this bit represents the RPA resolution status. ie.,<br>1 – RPA resolution pass.<br>0 – RPA resolution fail.<br>Else this bit is set to zero. | 0 |
| init_tx_addr[1] | 8 | MSB of the device's Own address type. This bit along with init_tx_addr[0] decides the own address of the device. | 0 |
| #Init_conn_index | 7:5 | Firmware programs the index of the conn engine for the master connection at init start. Index can be in the range 0x0 to 0x7 depending on the number of connections supported. | 0x0 |
| Reserved | 4 | Not used | x |
| init_filt_policy | 3 | The Initiator_Filter_Policy is used to determine whether the White List is used or not used.<br>0 -White list is not used to determine which advertiser to connect to. Instead the Peer_Address_Type and Peer Address fields are used to specify the address type and address of the advertising device to connect to.<br>1 - White list is used to determine the advertising device to connect to. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | Peer_Address_Type and Peer_Address fields are ignored when whitelist is used. | |
| init_rx_addr[1] | 2 | MSB of the device's peer address type (FW programmed). This bit along with init_rx_addr[0] decides the peer address of the device. | 0 |
| rx_addr/rx_tx_addr<br>init_rx_addr[0 ] | 1 | Peer address type.<br>The rx_addr field is updated by the receiver with the address type of the received connectable advertising packet.<br>1- addr type is random.<br>0-addr type is public.<br>LSB of the device's peer address type (FW programmed).<br>Init_rx_addr[1:0] –<br>0x0 - public address<br>0x1 - random address<br>0x2 - public identity address<br>0x3 - random(static) identity address | 0 |

| | | | |
|---|---|---|---|
| Tx_addr<br>init_tx_addr[0] | 0 | Own address type.<br>1- addr type is random.<br>0-addr type is public.<br>LSB of the device's Own address type.<br>Init_tx_addr[1:0] -<br>0x0 – use public address (addr type is zero)<br>0x1 – use public address (addr type is one)<br>0x2 – use RPA if matching entry is available in Resolving list else use public address.<br>0x3 - use RPA if matching entry is available in Resolving list else use random address. | 0 |

## Initiator interrupt clear register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x28 | 0x050 | INIT_INTR_CLEAR | WO | Clears the source of the interrupt. This register is written by firmware to clear interrupts set in the INIT_STATUS register. One or more interrupts can be cleared in a single write operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:5 | Not used. | XX |
| master_conn_created | 4 | Clear master connection created interrupt. Write to the register with this bit set to 1, clears the interrupt source. | 0 |
| Reserved | 3 | Not used. | X |
| Init_tx_start | 2 | Clear init transmission start interrupt. Write to the register with this bit set to 1, clears the interrupt source. | 0 |
| init_close_window | 1 | Clear Initiator scan window close interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |
| init_interval_expire | 0 | Clear Initiator scan window start interrupt. Write to the register with this bit set to 1, clears the interrupt source | 0 |

## Initiator status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x28 | 0x050 | INIT_STATUS | RO | Shows the status of the interrupt. This register is read by firmware to learn the pending initiator interrupts that are set and are to be served. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:5 | Not used. | XX |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| master_conn_created | 4 | If this bit is set it indicates connection is created as master. | 0 |
| Reserved | 3 | Not used. | X |
| Init_tx_start | 2 | If this bit is set it indicates initiator packet (CONREQ) transmission has started. | 0 |
| init_close_window | 1 | If this bit is set it indicates initiator scan window has finished. | 0 |
| init_interval_expire | 0 | If this bit is set it indicates initiator scan window has started. | 0 |

## #Initiator next instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x2A | 0x054 | INIT_NEXT_INSTANT | RO | Shows the instant w.r.t internal reference clock of 625us resolution at which next initiator scanning event begins. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| next_init_instant | 15:0 | Shows the instant w.r.t internal reference clock at which next initiator scanning event begins. | 0 |

## #Initiator Anchor Point register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x100 | 0x200 | INIT_ANCHOR_POINT | RW | Firmware programs this register (while programming INIT parameters) with the instant (value of bt_clock) at which the initiator should start the procedure for the first time. This will give firmware a better control in effectively scheduling the new connections in the multiple connection scenarios. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_anchor_point | 15:0 | The value of bt_clock (625us period) at which initiator should start the procedure. | 0 |

## Device Random address lower register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x2C | 0x058 | DEV_RAND_ADDR_L | RW | Lower 16 bit random address of the device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rand_addr | 15:0 | Lower 16 bit of 48-bit random address of the device. | 0 |

## Device Random address middle register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x2E | 0x05C | DEV_RAND_ADDR_M | RW | Middle 16 bit random address of the device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rand_addr | 15:0 | Middle 16 bit of 48-bit random address of the device. | 0 |

## Device Random address higher register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x30 | 0x060 | DEV_RAND_ADDR_H | RW | Higher 16 bit random address of the device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rand_addr | 15:0 | Higher 16 bit of 48-bit random address of the device. | 0 |

## Peer address lower register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x34 | 0x068 | PEER_ADDR_L | RW | Lower 16 bit address of the peer device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| peer_addr | 15:0 | Lower 16 bit of 48-bit address of the peer device. | 0 |

## Peer address middle register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x36 | 0x06C | PEER_ADDR_M | RW | Middle 16 bit address of the peer device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| peer_addr | 15:0 | Middle 16 bit of 48-bit address of the peer device. | 0 |

## Peer address higher register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x38 | 0x070 | PEER_ADDR_H | RW | Higher 16 bit address of the peer device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| peer_addr | 15:0 | Higher 16 bit of 48-bit of address of the peer device. | 0 |

## #Initiator Peer Address Lower Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x66 | 0x0CC | INIT_PEER_ADDR_L | RW | Lower 16 bit of init peer address. This register is used only if multiple connections are supported. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_peer_addr | 15:0 | Lower 16 bit of the 48 bit init peer address. | 0 |

## #Initiator Peer Address Middle Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x68 | 0x0D0 | INIT_PEER_ADDR_M | RW | Middle 16 bit of init peer address. This register is used only if multiple connections are supported. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_peer_addr | 15:0 | Middle 16 bit of the 48 bit init peer address. | 0 |

## #Initiator Peer Address Upper Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x6A | 0x0D4 | INIT_PEER_ADDR_H | RW | Lower 16 bit of init peer address. This register is used only if multiple connections are supported. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Init_peer_addr | 15:0 | Upper 16 bit of the 48 bit init peer address. | 0 |

The peer address registers are used for multiple purposes. The register is written by firmware to provide the peer address to be used for a hardware procedure. When firmware reads the register, it reads back peer address values updated by hardware.

While doing directed Advertising, the firmware writes the peer address of the device specified by the Direct_Address parameter of the LE_Set_Advertising_Parameters command.

While device is configured as an initiator without white list filtering, the peer address specified in the peer_address field of the create connection command is programmed into this register, which is used by hardware procedures.

While device is configured as an initiator and white list is enabled, firmware can read this register to get the address of the peer device from which connectable ADV packet was received and to which the connection is created.

When a connection is created as a slave, the firmware can read this register to get the address of the peer device to which connection is created.

*INIT_PEER_ADDR is used only for multiple connection support where peer address for initiation is kept at INIT_PEER_ADDR register and ADV peer address is kept/read from PEER_ADDR register.*

White List address type register lower word

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xA2 | 0x144 | WL_ADDR_TYPE_L | RW | Stores the address type of the device addresses stored in whitelist | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| type | 15:0 | 16 address type bits corresponding to first 16 device address stored in whitelist.<br>1-Address type is random.<br>0-Address type is public. | 0 |

White List address type register upper word

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xA4 | 0x148 | WL_ADDR_TYPE_U | RW | Stores the address type of the device addresses stored in whitelist | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| | 15:10 | Reserved for future use | |
| type | 9:0 | 10 address type bits corresponding to 17 to 26 device address stored in whitelist.<br>1-Address type is random.<br>0-Address type is public. | 0 |

## White list enable register lower word

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xA6 | 0x14C | WL_ENABLE_L | RW | Stores the valid entry bit corresponding to each of the device address stored in the whitelist. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| en | 15:0 | Stores the valid entry bit corresponding to each of the first 16 device addresses stored in the whitelist memory in the hardware.<br>1-White list entry is valid.<br>0-White list entry invalid. | 0 |

## White list enable register upper word

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xA8 | 0x150 | WL_ENABLE_U | RW | Stores the valid entry bit corresponding to each of the device address stored in the whitelist. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| | 15:10 | Reserved for future use | |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| en | 9:0 | Stores the valid entry bit corresponding to each of the 17 to 26 device addresses stored in the whitelist memory in the hardware.<br>1-White list entry is valid.<br>0-White list entry invalid. | 0 |

## Advertising data transmit FIFO

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x70 | 0x0E0 | ADV_TX_DATA_FIFO | WO | IO mapped FIFO of depth 16 (2 byte wide), to store ADV data of maximum length 31 bytes for transmitting. Firmware writes consecutive words by writing to the same address location. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| data | 15:0 | Advertising data for transmission. | 0 |

## Adv scan response data transmit FIFO

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x74 | 0x0E8 | ADV_SCN_RSP_TX_FIFO | WO | IO mapped FIFO of depth 16 (2 byte wide), to store scan response data of maximum length 31 bytes for transmitting. Firmware writes consecutive words by writing to the same location. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| data | 15:0 | scan response data for transmission. | 0 |

Note: ADV_TX_DATA_FIFO and ADV_SCN_RSP_TX_FIFO shares same physical FIFO of depth 32. 16 locations for each FIFO are allocated.

| | | |
|---|---|---|
| ADV_TX_DATA_FIFO<br>address 16'h70 | Data byte 1 | Length of adv host data |
| | Data byte 3 | Data byte 2 |
| | Data byte 31 | Data byte 30 |
| ADV_SCN_RSP_TX_FIFO<br>address 16'h74 | Data byte 1 | Length of scan response host data |
| | Data byte 3 | Data byte 2 |
| | Data byte 31 | Data byte 30 |

The length of the payload combined with first payload data and loaded to the advertise channel data transmit FIFO followed by rest of the host data.

Example: Structure of advertising channel transmit FIFO.

bit15 bit8 bit7 bit0

| | |
|---|---|
| Data byte 1 | Length of the payload stored in FIFO |
| Data byte 3 | Data byte 2 |
| Data byte 5 | Data byte 4 |
| Data byte 7 | Data byte 6 |

## Conn request data Transmit FIFO

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x78 | 0x0F0 | CONN_REQ_TX_FIFO | WO | IO mapped FIFO of depth 48, to store connection request data of maximum length 34 bytes for transmitting. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data | 15:0 | Connection request data during transmit operation. | 0 |

## Adv scan response data receive FIFO

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x7C | 0x0F8 | INIT_SCN_ADV_RX_FIFO | RO | IO mapped FIFO of depth 64, to store ADV and SCAN_RSP header and payload received | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | by the scanner. The RSSI value at the time of reception of this packet is also stored. Firmware reads from the same address to read out consecutive words of data. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data | 15:0 | adv, scan response data during receive operation. | 0 |

Note: The 16 bit header is first loaded to the advertise channel data receive FIFO followed by the payload data and then 16 bit RSSI.

Example: Structure of advertising channel receive FIFO with payload with even number of bytes.

bit15 bit8 bit7 bit0

| | |
|---|---|
| Header 2 | Header 1 |
| Data byte 2 | Data byte 1 |
| Data byte 4 | Data byte 3 |
| Data byte 6 | Data byte 5 |
| RSSI | RSSI |

Example: Structure of advertising channel receive FIFO with payload with odd number of bytes.

bit15 bit8 bit7 bit0

| | |
|---|---|
| Header 2 | Header 1 |
| Data byte 2 | Data byte 1 |
| Data byte 4 | Data byte 3 |
| | Data byte 5 |
| RSSI | RSSI |

Device public address lower register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0XE0 | 0X1C0 | DEV_PUB_ADDR_L | RW | Lower 16 bit public address of the device. | 0x3412 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| public_addr | 15:0 | Lower 16 bit of 48-bit public address of the device. | 0x3412 |

## Device public address middle register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0XE2 | 0X1C4 | DEV_PUB_ADDR_M | RW | Middle 16 bit public address of the device. | 0x0056 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| public_addr | 15:0 | Middle 16 bit of 48-bit public address of the device. | 0x0056 |

## Device public address higher register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0XE4 | 0X1C8 | DEV_PUB_ADDR_H | RW | Higher 16 bit public address of the device. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| public_addr | 15:0 | Higher 16 bit of 48-bit public address of the peer device. | 0 |

## Advertising channel transmit power register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0XE6 | 0X1CC | ADV_CH_TX_POWER | RW | The advertising channel transmit power register sets the transmit power level used | 0x874F |

| | | | | | |
|---|---|---|---|---|---|
| | | | | for LE advertising channel packets and for DTM mode transmissions.<br><br>The same register is used for setting Transmit power level of all non-connection channels. This includes: Advertising, scanning, Initiating, and Direct test mode (DTM Transmitter tests). | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| adv_transmit_power | 15:0 | Size: 1 Octet (signed integer)<br><br>Range: -20 ≤ N ≤ 10<br><br>Units: dBm<br><br>Accuracy: +/- 4 dBm in general.<br><br>In implementation this is a radio specific value. | 0x874F |

## Offset to first instant register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0XE8 | 0X1D0 | OFFSET_TO_FIRST_INSTANT | RW | Offset to the first instant register. The first event instant is determined by firmware based on other procedures which may be on with various intervals, so as to not overlap on the existing procedure instants. For this, firmware determines the offset to the first instant from the current clock and programs the offset in OFFSET_TO_FIRST_INSTANT register.<br><br>Unit is in time slots of 625us<br><br>ex: if current clock value is 0004,and offset is 0008,then first event will begin when clock value becomes 000c. | 0x0006 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| offset_to_first_event | 15:0 | The offset w.r.t the internal reference clock at which instant the first event occurs.<br><br>This register will give flexibility to the firmware to position the connection at a desired point with respect to the internal free running clock. It is optional to be updated by firmware. This is not updated in the current firmware. This is for future use. | 0x0006 |

## Advertiser configuration register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xEA | 0x1D4 | ADV_CONFIG | RW | Advertiser procedure configuration register. Firmware sets the configuration parameters to this register before issuing start adv command. | 0x20FF |

| Field | Bit | Description | Reset |
|---|---|---|---|
| adv_pkt_interval | 15:11 | Time between the beginnings of two consecutive advertising PDU's.<br>Time = N * 0.625 msec<br>Time Range: <=10msec. | 00100b |
| Reserved | 10:9 | Not used. | 0 |
| Adv_rand_disable | 8 | Disable randomization of adv interval. When disabled, interval is same as programmed in adv_interval register. | 0 |
| adv_timeout_en | 7 | Enable adv_timeout interrupt. Applicable in adv_direct_ind advertising. | 1 |
| slv_connected_en | 6 | Enable slave connected interrupt. | 1 |
| adv_conn_req_rx_en | 5 | Enable connect request packet received interrupt. | 1 |
| adv_scn_req_rx_en | 4 | Enable scan request packet received interrupt. | 1 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| scn_tx_en | 3 | Enable scan response packet transmitted interrupt. | 1 |
| adv_tx_en | 2 | Enable adv packet transmitted interrupt. | 1 |
| adv_cls_en | 1 | Enable advertising event stop interrupt. | 1 |
| adv_strt_en | 0 | Enable advertising event start interrupt. | 1 |

## Scan configuration register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xEC | 0x1D8 | SCAN_CONFIG | RW | Scanner procedure configuration register. Firmware sets the configuration parameters to this register before issuing start scan command | 0xE07F |

| Field | Bit | Description | Reset |
|---|---|---|---|
| scan_channel_map | 15:13 | Advertising channels that are enabled for scanning operation.<br>15- Enables channel 39 for use.<br>14- Enables channel 38 for use.<br>13- Enables channel 37 for use. | 111 |
| Reserved | 12 | Not used. | X |
| backoff_enable | 11 | Enable random backoff feature in scanner.<br>1-enable.<br>0-disable. | 0 |
| Reserved | 10:5 | Not used. | XX |
| scn_rsp_rx_en | 4 | Enable scan_rsp packet received interrupt. | 1 |
| adv_rx_en | 3 | Enable adv packet received interrupt. | 1 |
| scn_tx_en | 2 | Enable scan request packet transmitted interrupt. | 1 |
| scn_close_en | 1 | Enable scan event close interrupt. | 1 |
| scn_strt_en | 0 | Enable scan event start interrupt. | 1 |

## #Initiator configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xEE | 0x1DC | INIT_CONFIG | RW | Initiator procedure configuration register. Firmware sets the configuration parameters to this register before issuing create connection command | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| init_channel_map | 15:13 | Advertising channels that are enabled for initiator scanning operation.<br>15- Enables channel 39 for use.<br>14- Enables channel 38 for use.<br>13- Enables channel 37 for use. | 0 |
| Reserved | 12:8 | Not used. | XX |
| conn_created | 4 | Enable master connection created interrupt | 0 |
| Reserved | 3 | Reserved | |
| conn_req_tx_en | 2 | Enables connection request packet transmission start interrupt. | 0 |
| init_close_en | 1 | Enable Initiator scan window close interrupt. | 0 |
| init_strt_en | 0 | Enable Initiator scan window start interrupt. | 0 |

## Whitelist base address register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X462 | 0x8C4 | WHITELIST_BASE_ADDR | RW | It is the starting address of white list memory which holds the white listed device address.<br>For a 48 bit device address, three writes of 16 bits is required at the appropriate offset from this base address. | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | The whitelist device addresses are stored as group of 3-words at offset of N*3, where N=0 to 7, from this base address.<br>While writing the device address, the firmware writes the address in the following order for storage.<br>1st write - [15:0],<br>2nd write - [31:16],<br>3rd write - [47:32] bits of the device address. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Device_addr | 15:0 | Device address values written to white list memory are written as 16-bit wide address. | 0 |

## Whitelist end address register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X4FE | 0x9FC | WHITELIST_END_ADDR | RW | It is the last address of white list memory which holds the white list device address. It holds last [47:32] bits of 26th white list device address. It is not accessed by firmware, only used for hardware reference. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Device_addr | 15:0 | Device address values written to white list memory are written as 16-bit wide address. | 0 |

## Advertiser Tx memory base address register – Reserved for future use

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0X160 | 0X2C0 | ADV_TX_MEM_BASE_ADDR | RW | It is the starting address of ADV Tx memory which holds the data to be transmitted during Advertising operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data | 15:0 | Data values written to Tx memory are written as 16-bit wide data. | 0 |

## Connection Tx memory base address register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0X700 | 0XE00 | CONN_TXMEM_BASE_ADDR | WO | It is the starting address of Connection Transmit data memory which holds the data to be transmitted during connection.<br>The connection Transmit memory is individually addressable location for firmware. So firmware writes to consecutive even address values to write the next word (2-byte) of data. Hardware accesses this memory as a FIFO. The hardware buffer index is managed in hardware. No buffer index needs to be maintained for firmware access. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data | 15:0 | Data values written to Tx memory are written as 16-bit wide data. | 0 |

## Connection Rx memory base address register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0X400 | 0X800 | CONN_RXMEM_BASE_ADDR | RO | It is the starting address of Connection Receive data memory/FIFO which holds the data to be received during connection.<br>The connection receive memory/FIFO is used as a FIFO by both hardware and firmware. Firmware needs to read from the same address to read out the consecutive words in the FIFO. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data | 15:0 | Data values read from Rx memory are read as 16-bit wide data | 0 |

## Conn_req_word0 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E0 | 0x3C0 | CONN_REQ_WORD0 | RW | The connect request word0 register must be programmed with the access address value of the connect request packet, before initiating connection. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Access_addr[15:0] | 15:0 | This field defines the lower 16 bits of the access address that is to be sent in the connect request packet of the initiator. | 0x0000 |

## Conn_req_word1 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E2 | 0x3C4 | CONN_REQ_WORD1 | RW | The connect request word1 register must be programmed | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | with the access address value of the connect request packet, before initiating connection. After slave connection this value can be obtained by reading this register on the advertiser side. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Access_Address[31:16] | 15:0 | This field defines the upper16 bits of the access address that is to be sent in the connect request packet of the initiator. | 0x0000 |

## Conn_req_word2 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E4 | 0x3C8 | CONN_REQ_WORD2 | RW | This field defines the lower byte [7:0] of the CRC initialization value, and tx_window_size [7:0], to be sent in the connect request packet of the initiator. After slave connection these values can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| crc_init[7:0] | 15:8 | This field defines the lower byte [7:0] of the CRC initialization value. | 0 |
| Tx_window_size[7:0] | 7:0 | window_size along with the window_offset is used to calculate the first connection point anchor point for the master.<br>This shall be a multiple of 1.25 ms in the range of 1.25 ms to the lesser of 10 ms and (connInterval – 1.25 ms).<br>Values range from 0 to 10 ms. | 0 |

## Conn_req_word3 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E6 | 0x3CC | CONN_REQ_WORD3 | RW | This field must be programmed with the upper byte [23:8] of the CRC initialization value.After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| crc_init[23:8] | 15:0 | This field defines the upper byte [23:8] of the CRC initialization value that is to be sent in the connect request packet of the initiator. | 0x0000 |

## Conn_req_word4 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1E8 | 0x3D0 | CONN_REQ_WORD4 | RW | This field defines the 16 bits of the transmit window offset that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Tx_window_offset | 15:0 | This is used to determine the anchor point for the master transmission.<br>Range: This shall be a multiple of 1.25 ms in the range of 0 ms to connInterval value. | 0 |

## Conn_req_word5 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x1EA | 0x3D4 | CONN_REQ_WORD5 | RW | This field defines the 16 bits of the connection interval value that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_interval_val | 15:0 | The value configured in this register determines the spacing between the connection events.<br>This shall be a multiple of 1.25 ms in the range of 7.5 ms to 4.0 s. | 0x0000 |

## Conn_req_word6 register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1EC | 0x3D8 | CONN_REQ_WORD6 | RW | This field defines the 16 bits of the slave latency value that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Slv_latency_val | 15:0 | The value configured in this field defines the number of consecutive connection events that the slave device is not required to listen for master. The value of connSlaveLatency should not cause a Supervision Timeout. This shall be an integer in the range of 0 to ((connSupervision Timeout/ connInterval)-1). connSlaveLatency shall also be less than 500. | 0x0000 |

## Conn_req_word7 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1EE | 0x3DC | CONN_REQ_WORD7 | RW | This field defines the 16 bits of the supervision timeout value that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sup_to_val | 15:0 | This field defines the maximum time between two received Data packet PDUs before the connection is considered lost.<br>This shall be a multiple of 10 ms in the range of 100 ms to 32.0 s and it shall be larger than (1+connSlaveLatency)*connInterval. | 0x0000 |

## Conn_req_word8 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1F0 | 0x3E0 | CONN_REQ_WORD8 | RW | This field defines the channel map for channels [15:0], that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels[15:0] | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the lower 16 (15:0) data channel indices. | Data_channels |

| | | | |
|---|---|---|---|
| | | '1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | |

## Conn_req_word9 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1F2 | 0x3E4 | CONN_REQ_WORD9 | RW | This field defines the channel map for channels [31:16], that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels[31:16] | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the middle 16 (31:16) data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | Data_channels |

## Conn_req_word10 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1F4 | 0x3E8 | CONN_REQ_WORD10 | RW | This field defines the channel map for channels [36:32], that is to be sent in the connect request packet of the initiator. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Data_channels[36:32] | 4:0 | This register field indicates which of the data channels are in use. This stores the information for the upper 5 (36:32) data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | Data_channels |

## Conn_req_word11 register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1F6 | 0x3EC | CONN_REQ_WORD11 | RW | The connect request word0 register must be programmed with Connection parameters sca, hop_increment value of the connect request packet, before initiating connection. After slave connection this value can be obtained by reading this register on the advertiser side. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sca[2:0] | 7:5 | This field defines the sleep clock accuracies given in ppm. | 0 |
| hop_increment[4:0] | 4:0 | This field is used for the data channel selection process. | 0 |

## ##RPA timer interval register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x582 | 0xB04 | RPA_TIMER_INTERVAL | RW | The RPA timer interval register stores the RPA refresh time interval. (FW programmed). | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| rpa_timer_interval_val | 15:0 | This field holds the value of the RPA refresh time interval in seconds. | 0 |

## ##Privacy Configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x108 | 0x210 | LL_PRIVACY_CONFIG | RW | This register fields can be set to enable/disable Privacy feature in LLH. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:2 | Reserved for future use. | 0 |
| allow_rpa_in_whitelist | 1 | Allow initiator to receive advertising packets even if peer address resolution fails by adding the received advertiser RPA to the whitelist and to the resolving list as per Errata 6984. | 0 |
| privacy_config | 0 | Whenever this bit is set LLH will try to resolve the address for all ADVCH procedures. | 0 |

## ##RPA selection index register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x10A | 0x214 | CURRENT_LOCAL_IRK_INDEX | RW | This register is programmed by firmware before advertising and Initiation procedures. This bit is used to indicate the local IRK to be used by HW during advertising and Initiation procedures. | 0xFF |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Reserved for future use. | 0 |
| init_local_irk_index | 7:4 | This field holds the resolving list index corresponding to the IRK to be used for the INIT procedure. This field is invalid when white list is enabled. | 0xF |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | Value 0xF – valid IRK for INIT procedure is not set in resolving list. | |
| adv_local_irk_index | 3:0 | This field holds the resolving list index corresponding to the IRK to be used for the ADV procedure. This field is also valid when white list is enabled.<br>Value 0xF – valid IRK for ADV procedure is not set in resolving list. | 0xF |

## ##Peer identity address type register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x10C | 0x218 | PEER_IDENTITY_ADDR_TYPE | RW | This register stores the peer identity address type (Public or Random) corresponding to the IRK pair populated in the resolving list. The value is valid if the resolving list entry is valid. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:4 | Reserved for future use. | 0 |
| peer_id_addr_type | 3:0 | Peer_id_addr_type [i] = 0/1 => ith resolving list entry corresponds to a peer whose address is public/random. | 0 |

## ##Resolving list entry valid register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x10E | 0x21C | RES_LIST_VALID | RW | This register is used to indicate to HW which resolving list entries are valid/ invalid. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:4 | Reserved for future use. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| device_valid_entry | 3:0 | Device_valid_entry [i] = 0/1 => ith resolving list entry is invalid/valid. | 0 |

## ##Advertising parameters 2 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x0A | 0x14 | ADV_PARAMS_2 | RW | This register stores the own address type for the Advertising procedure. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:2 | Reserved for future use. | 0 |
| adv_tx_addr | 1:0 | Device Own_address_type field.<br>0x0 - Own address type is public address.<br>0x1 - Own address type is random address.<br>0x2 - Own address type is RPA. If resolving list contains no matching entry, use public address.<br>0x3 - Own address type is RPA. If resolving list contains no matching entry, use random address. | 0 |

## ##Local IRK n registers

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1100-0x11FE | - | L_IRK_n_x | RW | These registers store the local IRK value for nth resolving list entry. IRK value is of 128 bits and stored in Little Endian format (lowest address register has the LSW). | 0x00000000000000000000000000000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dev_localirk[n] | 127:0 | This field has the 128 bit value of local IRK for nth resolving list entry. | 0 |

| | | | |
|---|---|---|---|
| | | Note: Refer to appendix 6 for full addressing with respect to each resolving entry. | |

## ##Peer IRK n registers

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1200-0x12FE | - | P_IRK_n_x | RW | These registers store the peer IRK value for nth resolving list entry. IRK value is of 128 bits and stored in Little Endian format (lowest address register has the LSW). | 0x00000000000000000000000000000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dev_peerirk[n] | 127:0 | This field has the 128 bit value of peer IRK for nth resolving list entry.<br>Note: Refer to appendix 6 for full addressing with respect to each resolving entry. | 0 |

## ##Peer identity address n registers

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1300-0x135E | - | PEER_ID_ADDR_n_x | RW | These registers store the peer identity address for nth resolving list entry. Identity address is of 48 bits and stored in Little Endian format (lowest address register has the LSW). | 0x000000000000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dev_identity_addr[n] | 47:0 | This field has the 48 bit value of peer ID address for nth resolving list entry.<br>Note: Refer to appendix 6 for full addressing with respect to each resolving entry. | 0 |

## ##Local RPA n registers

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1360-0x13BE | - | L_RPA_n_x | RO | These registers store the local RPA generated by the HW using the local IRK of the nth resolving list entry. RPA is of 48 bits and stored in Little Endian format (lowest address register has the LSW). | 0x0000000000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| self_rpa_list_n | 47:0 | This field has the 48 bit value of local RPA for nth resolving list entry.<br>Note: Refer to appendix 6 for full addressing with respect to each resolving entry. | 0 |

## ##Peer RPA n registers

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x13C0-0x141E | 0x1CC0 – 0x1CC8 | P_RPA_n_x | RO | These registers store the peer RPA generated by the HW using the peer IRK of the nth resolving list entry. RPA is of 48 bits and stored in Little Endian format (lowest address register has the LSW). | 0x0000000000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| peer_rpa_list_n | 47:0 | This field has the 48 bit value of peer RPA for nth resolving list entry.<br>Note: Refer to appendix 6 for full addressing with respect to each resolving entry. | 0 |

## ##RPA timer wrap count register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x586 | 0xB0C | RPA_TIMER_WRAP_COUNT | RO | The RPA timer wrap count register stores the wrap count value. A wrap is a complete bt_clock roll from 0x0000 to 0xFFFF. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| **Reserved** | **15:10** | | **0** |
| rpa_timer_wrap_count | 9:0 | This field holds the bt_clock wrap count value. | 0 |

## ##RPA timer current wrap register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x584 | 0xB08 | RPA_TIMER_CURRENT_WRAP | RO | The RPA timer current wrap register stores the current wrap count value. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:12 | | 0 |
| rpa_timer_current_wrap | 11:2 | This field holds RPA timer's current wrap count value. | 0 |
| rpa_timer_en | 1 | Indicates whether RPA timer is ON in hardware.<br>1 – on<br>0 – off | 0 |
| wrap_valid | 0 | Indicates whether RPA timer's warp is valid.<br>1 – wrap is valid<br>0 – wrap is invalid | 0 |

## ##RPA timer next instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x588 | 0xB10 | RPA_TIMER_NEXT_INSTANT | RO | Holds the next RPA timer instant value. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| **Reserved** | **15:10** | | **0** |
| rpa_timer_next_instant | 9:0 | Holds the next RPA timer instant value. This instant is valid when wrap valid = 1. | 0 |

## ##RPA LFSR seed lower register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x9C | 0x138 | RPA_LFSR_L | RW | Lower seed value of the LFSR for RPA generation. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rpa_lsfr_l | 15:0 | Firmware can writes to this register to update the seed value of the LFSR for random RPA generation after the reset. | 0 |

## ##RPA LFSR seed middle register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x9E | 0x13C | RPA_LFSR_M | RW | Middle seed value of the LFSR for RPA generation. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rpa_lsfr_m | 15:0 | Firmware can writes to this register to update the seed value of the LFSR for random RPA generation after the reset. | 0 |

## ##RPA LFSR seed higher register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xA0 | 0x140 | RPA_LFSR_H | RW | Upper seed value of the LFSR for RPA generation. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:12 | RFU | **0** |
| Rpa_lsfr_u | 11:0 | Firmware can writes to this register to update the seed value of the LFSR for random RPA generation after the reset. | 0 |

## ##Privacy Mode register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x280 – 0x28E | 0x500 – 0x51C | PRIVACY_MODE_0 – PRIVACY_MODE_7 | RW | Mode of the privacy address added for each entry in the resolving list. | 0x0 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:1 | RFU | **0** |
| Privacy_mode | 0 | This bit defines the mode of the device is network privacy or device privacy.<br>0 – Network Privacy Mode<br>1 – Device Privacy Mode | 0 |

### 64.5.1.1.3 Data Channel Registers

Data Channel Register Descriptions for the Bluetooth Link Layer

Transmit window offset register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x40 | 0x080 | TRANSMIT_WINDOW_OFFSET | RW | Stores the transmit window offset parameter used during connection setup and connection update procedures.<br>The register is updated by hardware when device is slave. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| window_offset | 15:0 | This is used to determine the first anchor point for the master transmission, from the time of connection creation.<br>Range: This shall be a multiple of 1.25 ms in the range of 0 ms to connInterval value. | 0 |

## Transmit window size register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x42 | 0x084 | TRANSMIT_WINDOW_SIZE | RW | Stores the transmit window size parameter used during connection setup and connection update procedures. The register is updated by hardware when device is slave. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| window_size | 7:0 | window_size along with the window_offset is used to calculate the first connection point anchor point for the master.<br>This shall be a multiple of 1.25 ms in the range of 1.25 ms to the lesser of 10 ms and (connInterval – 1.25 ms).<br>Values range from 0 to 10 ms. | 0 |

## Data channel map 0 (lower word) register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x44 | 0x088 | DATA_CHANNELS_L0 | RW | Stores the channel map for channels [15:0]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the lower 16 (15:0) data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x0000 |

## Data channel map 0(middle word) register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x46 | 0x08C | DATA_CHANNELS_M0 | RW | Stores the channel map for channels[31:16]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the middle 16 (31:16) data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x0000 |

## Data channel map 0(upper word) register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x48 | 0x090 | DATA_CHANNELS_H0 | RW | Stores the channel map for channels[36:32]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:5 | Unused | 0 |
| Data_channels | 4:0 | This register field indicates which of the data channels are in use. This stores the information for the upper 5 (36:32) data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x00 |

## Data channel map 1 (lower word) register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x4C | 0x098 | DATA_CHANNELS_L1 | RW | Stores the channel map for channels[15:0]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the lower 16 data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x0000 |

## Data channel map 1 (middle word) register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x4E | 0x09C | DATA_CHANNELS_M1 | RW | Stores the channel map for channels[31:16]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels | 15:0 | This register field indicates which of the data channels are in use. This stores the information for the middle 16 data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x0000 |

## Data channel map 1 (upper word) register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x50 | 0x0A0 | DATA_CHANNELS_H1 | RW | Stores the channel map for channels[36:32]. The register is updated by firmware whenever a new/updated channel map is available for the connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Data_channels | 15:5 | Unused | 0 |
| Data_channels | 4:0 | This register field indicates which of the data channels are in use. This stores the information for the upper 5 data channel indices.<br>'1' indicates the corresponding data channel is used and '0' indicates the channel is unused. | 0x00 |

*Note: The Data channel map 0 and data channel map 1 are two sets of channel maps stored, common for all the connections. At any given time, only two maps can be maintained and the connections will use one of the two sets as indicated by the channel map index field in the CE_CNFG_STS registers specific to the link. Firmware must also manage to update this field along with the map.*

## Connection channel Status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x56 | 0x0AC | CONN_STATUS | RO | Indicates the status of the connection channel data path and other common connection channel operations. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rx_packet_counter | 15:12 | This field stores the count for the number of receive packets in the receive FIFO that are still not ready by firmware.<br>The counter value is incremented by hardware for every good packet it stores in the FIFO.<br>After firmware reads a packet, it decrements the counter by issuing the PACKET_RECEIVED command from the commander. | 0 |
| Reserved | 11:0 | Reserved for future use | 0 |

## Connection configuration register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xF0 | 0x1E0 | CONN_CONFIG | RW | This register fields can be set to configure the LLH in data transfer scenarios. | 0x631F |

| Field | Bit | Description | Reset |
|---|---|---|---|
| pdu_index_auto_updt | **15** | This bit enable the LL hardware to update the index of the PDU being queued in the data buffer.<br>0x0- LL Hardware to update the index of the PDU being queued.<br>0x1- LL firmware to update the index when PDU is queued. | 0x0 |
| mask_suto_at_update | 14 | This bit is used to enable/ disable masking of internal hardware supervision timeout | 0x01 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | trigger when switching from old connection parameters to new parameters.<br>1 - Enable<br>0 - Disable | |
| extend_cu_tx_win | 13 | This bit is used to enable/ disable extending the additional rx window on slave side during connection update in event of packet miss at the update instant.<br>1 - Enable<br>0 - Disable | 0x01 |
| slv_md_config | 12 | Configuration is provided to send last data packet with MD bit set/zero or one (only for slave mode).<br>SLV_MD_CONFIG bit has effect only when md_bit_ctr bit is not set<br>0 – The MD bit is set based on the transmit buffer empty condition. The last packet goes with a MD bit set since the FIFO would not be empty as we have to retain the data in the buffer till we realize the ACK from the remote side.<br>1 - MD bit will be controlled based on the availability of next data in the FIFO. So send last data packet with MD bit zero or one. If the data transmitted is the last packet in the FIFO the MD bit will be '0' in that packet. Only if the next location holds data then MD bit will be '1'. | 0x0 |
| aww_en | 11 | This field indicates whether window widen optimization is enable or not.<br>0 – It indicates AWW is enable<br>1 – It indicates AWW is disable | 0x0 |
| sl_dsm_en | 10 | This resiter field indicates whether slave latency dsm is enable or not.<br>0 – It indicates SL_DSM is enable<br>1 – It indicates SL_DSM is disable | 0x0 |

| | | | |
|---|---|---|---|
| index_not_in_addr | 9 | This register field indicates whether connection index is present in address or not.<br>1 – Index is not in address<br>0 – Index is in address | 0x1 |
| Sw_cntrl_md | 8 | This register field indicates whether the MD (More Data) bit needs to be controlled by 'software' or, 'hardware and software logic combined'.<br>1 - MD bit is exclusively controlled by software, ie based on status of *CE_CNFG_STS_REGISTER[6] - md bit.*<br>0 - MD Bit in the transmitted pdu is controlled by software and hardware logic. MD bit is set in transmitted packet, only if the software has set the md bit in CE_CNFG_STS_REGISTER[6] and either of the following conditions is true,<br>1. If there are packets queued for transmission.<br>2. If there is an acknowledgement awaited from the remote side for the packet transmitted. | 0x1 |

| | | | |
|---|---|---|---|
| rx_intr_threshold | 7:4 | This register field allows setting a threshold for the packet received interrupt to the firmware.<br>For example if the value programmed is<br>0x2 – then LLH will generate interrupt only on receiving the second packet.<br>In any case if the received number of packets in a conn event is less than the threshold or there are still packets (less than threshold) | 0x1 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | pending in the Rx FIFO, LLH will generate the interrupt at the ce_close.<br>Min value possible is 1. Max value depends on the Rx FIFO capacity. | |
| rx_pkt_limit | 3:0 | Defines a limit for the number of Rx packets that can be received by the LLH. Default maximum value is 0xF.Minimum value shall be '1' or no packet will be stored in the Rx FIFO. | 0xF |

*Note:*

*The DUT should not send empty packets with MD bit set to 1. This will extend the connection event and increase the power consumption which is unnecessary. However there could be scenarios where maximum throughput is the target. In that case we would need to extend the connection event and allow additional time for the software to queue additional data. So both the behaviors are kept in the implementation and can be selected using* md_bit_ctr *(ie CONN_CONFIG[8]).*

md_bit_ctr *= 1 - MD bit is exclusively controlled by software.*

*If this bit is set, the MD bit in the transmitted packets is exclusively based on the status of md bit (CE_CNFG_STS_REGISTER[6]).*

*In this mode, empty packets with MD bit set would be transmitted by us during the time an acknowledgement is being processed in the other end. This feature will extend the connection event since the remote end host will get see the MD bit and so stays in the same connection event, and this will allow us more time for our software to process the acknowledgement and queue additional data from the host to hardware. This is useful when we need to maximize the data transmitted in a connection interval.*

*sw_cntrl_md = 0 - MD bit is controlled by software and hardware logic.*

*Note that MD bit is not set in the transmitted packet if software has not set the md bit in CE_CNFG_STS_REGISTER[6] as 0b.*

*In this mode of operation an empty packet will still be sent since the send status clearing and more data check is at same time.*

Connection channel transmit power register

| Addr | Addr | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | |
|---|---|---|---|---|---|
| 0xF2 | 0x1E4 | CONN_CH_TX_POWER | RW | Connection channel transmit power. This register controls transmit power on all connection channel transmissions. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_tx_power[15:0] | 15:0 | Transmit power to be used for all packets transmitted on the connection channel. | 0x0000 |

## *Connection Interrupt mask register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xF8 | 0x1F0 | CONN_INTR_MASK | RW | Connection Interrupt enable register. This register controls enabling of interrupts and other enables common for all connections. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| *ping_nearly_expired_intr | 15 | If this bit is set ping timer nearly expired interrupt is enabled. | 0 |
| *ping_timer_expired_intr | 14 | If this bit is set ping timer expired interrupt is enabled. | 0 |
| Reserved | 13:10 | Unused. | 0 |
| rx_bad_pdu_int_en | 9 | If this bit is set packet receive bad pdu interrupt is enabled. Effective only when bit 6 is set. | 0 |
| rx_good_pdu_int_en | 8 | If this bit is set packet receive good pdu interrupt is enabled. Effective only when bit 6 is set. | 0 |
| conn_updt_intr_en | 7 | If this bit is set connection update interrupt is enabled. | 0 |
| ce_rx_int_en | 6 | If this bit is set interrupt is enabled for reception of packet in a connection event. Bit 8 and 9 are sub-mask bits below this mask. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| ce_tx_ack_int_en | 5 | If this bit is set transmission acknowledgement interrupt is enabled:<br>This interrupt is generated to indicate to the firmware that a non-empty packet transmitted is successfully acknowledged by the remote device.<br>For negative acknowledgements from remote device, this interrupt indication is not generated. | 0 |
| close_ce_int_en | 4 | If this bit is set connection event closed interrupt is enabled. | 0 |
| start_ce_int_en | 3 | If this bit is set connection event start interrupt is enabled | 0 |
| map_updt_int_en | 2 | If this bit is set, channel map update interrupt is enabled. | 0 |
| conn_estb_int_en | 1 | If this bit is set connection establishment interrupt is enabled. | 0 |
| conn_cl_int_en | 0 | If this bit is set connection closed interrupt is enabled. | 0 |

## Slave timing control register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xFA | 0x1F4 | SLAVE_TIMING_CONTROL | RW | Slave timing control register. This register controls slave related timing. | 0xBE96 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Slave_time_adj_val | 15:8 | Timing adjust value. The internal micro second counter is adjusted to this value whenever slave receives a good access address match at connection anchor point. This will ensure the slave gets synchronized to master timing. | 0xBE |
| Slave_time_set_val | 7:0 | Programmable adjust value to the clock counter when slave is connected | 0x96 |

## Window widen for offset register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xB2 | 0x164 | WINDOW_WIDEN_WINOFF | RW | Window widen value corresponding to transmitwindowoffset. Firmware calculates the possible drift due to transmitwindowoffset to the first packet after connection/ connection update and programs the value into this register. The value is in microseconds. | 0x000A |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:12 | Unused | |
| Window_widen | 11:0 | This field stores the additional number of microseconds the slave must conn_config its listening window to listen for a master packet for receiving the first packet after connection creation. This value is calculated based on the window offset value to the first anchor point. This is used at connection setup directly. During connection setup, this value is added with window_widen_intvl register value to calculate the window widening size. | 00A |

## Connection Index register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x58 | 0x0B0 | CONN_INDEX | RW | Index of the connection to which the connection-specific parameter is being written to or read from. Firmware shall update this register with proper index before writing/ reading the connection-specific registers (refer to register summary before for the connection specific register set). | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | This register is relevant only for multiple connection support. If multiple connection is supported but number of connections is made one then the register value needs to be kept 0x0. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:3 | Reserved | 0 |
| fw_conn_index | 2:0 | This field is used to index the multiple connections existing. Range is 0 to one less than maximum number of connections supported (limited to 4 as of now).<br>0x0 – Connection Engine 0 in LLH<br>0x1 – Connection Engine 1 in LLH | 0 |

*Note – An alternative for programming of the conn index register for every access to the connection registers is to pass the conn index along with the LLH address itself which requires address space to be available for this mapping.*

*Connection Interrupt clear register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x54 | 0x0A8 | CONN_INTR_CLEAR | WO | Clear connection interrupts. Write to the register to clear one more connection interrupts. This register is implemented per connection. To clear interrupt for a specific connection, connection index register must be programmed before writing to this register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| *conn_ping_timer_nearly_expire | 15 | If this bit is written with 1, it clears the conn_ping_timer_nearly_expire interrupt. | 0 |
| *conn_ping_timer_expire | 14 | If this bit is written with 1, it clears the | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | conn_ping_timer_expire interrupt. | |
| Reserved | 13:8 | Unused | 0 |
| con_updt_done | 7 | If this bit is written with 1, it clears the connection updated interrupt. | 0 |
| ce_rx | 6 | If this bit is written with 1, it clears the connection event received interrupt. | 0 |
| ce_tx_ack | 5 | If this bit is written with 1, it clears the ce transmission acknowledgement interrupt. | 0 |
| close_ce | 4 | If this bit is written with 1, it clears the connection event closed interrupt. | 0 |
| start_ce | 3 | If this bit is written with 1, it clears the connection event started interrupt. | 0 |
| map_updt_done | 2 | If this bit is written with 1, it clears the map update done interrupt. | 0 |
| conn_estb | 1 | If this bit is written with 1, it clears the connection established interrupt. | 0 |
| conn_closed | 0 | If this bit is written with 1, it clears the connection updated interrupt. | 0 |

## *Connection Interrupt status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x54 | 0x0A8 | CONN_INTR_STATUS | RO | Connection Interrupt status register. To read interrupt for a specific connection, connection index register must be programmed before reading from this register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| *ping_nearly_expird_intr | 15 | If this is set, it indicates that ping timer has nearly expired. | 0 |
| *ping_timer_expird_intr | 14 | If this is set, it indicates that ping timer has expired. | 0 |
| rx_pdu_status | 13:11 | Status of PDU received. This information is valid along with receive interrupt. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | Xx1 – Bad Packet (packet with CRC error)<br>000 – empty PDU<br>010 - new data (non-empty) PDU<br>110 – Duplicate Packet | |
| discon_status | 10:8 | Reason for disconnect – indicates the reason the link is disconnected by hardware.<br>001 – connection failed to be established<br>010 - supervision timeout<br>011 – kill connection by host<br>100 – kill connection after ACK transmitted<br>101 – PDU response timer expired | 0 |
| con_updt_done | 7 | This bit is set when the last connection event with previous connection parameters is reached. The bit is set immediately after the receive operation at the anchor point of the last connection event. | 0 |
| ce_rx | 6 | If this bit is set it indicates that a packet is received in the connection event. | 0 |
| ce_tx_ack | 5 | If this bit is set it indicates that the connection event transmission acknowledgement is received for the previous non-empty packet transmitted. | 0 |
| close_ce | 4 | If this bit is set it indicates that the connection event closed interrupt has happened. | 0 |
| start_ce | 3 | If this bit is set it indicates that the connection event started interrupt has happened. | 0 |
| map_updt_done | 2 | If this bit is set it indicates that the channel map update is completed at the instant specified by the firmware. | 0 |
| conn_estb_updt | 1 | If this bit is set it indicates that the connection has been established. The bit is also set when a connection update procedure is completed, at the start of the first anchor point with the updated parameters. | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| conn_closed | 0 | If this bit is set it indicates that the link is disconnected. | 0 |

## Connection Interval register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x80 | 0x100 | CONN_INTERVAL | RW | Connection Interval registers. Firmware writes the connection interval specific to the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| connection Interval | 15:0 | The value configured in this register determines the spacing between the connection events.<br>This shall be a multiple of 1.25 ms in the range of 7.5 ms to 4.0 s. | 0x0000 |

## Supervision timeout register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x82 | 0x104 | SUP_TIMEOUT | RW | Supervision timeout for the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| supervision_timeout | 15:0 | This field defines the maximum time between two received Data packet PDUs before the connection is considered lost. | 0x0000 |

| | | | |
|---|---|---|---|
| | | This shall be a multiple of 10 ms in the range of 100 ms to 32.0 s and it shall be larger than (1+connSlaveLatency)*connInterval. | |

## Slave Latency register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x84 | 0x108 | SLAVE_LATENCY | RW | Slave latency for the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| slave_latency | 15:0 | The value configured in this field defines the number of consecutive connection events that the slave device is not required to listen for master.<br>The value of connSlaveLatency should not cause a Supervision Timeout.<br>This shall be an integer in the range of 0 to ((connSupervision Timeout/ connInterval)-1). connSlaveLatency shall also be less than 500. | 0x0000 |

## Connection event length register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x86 | 0x10C | CE_LENGTH | RW | Connection event length for the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x00<br>00 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Connection event Length | 15:0 | This field defines the length of Connection event. This value is derived from the CE length HCI parameters received from the host. This determines the number of master transmit slots in a connection event, subject to either of the MD bits being set. If both MD bits are set to 0, this has no effect.<br>Units : 625 us.<br>Note: The connection event length as specified by the CE_LENGTH shall not exceed CONN_INTERVAL – 1.25 ms. | 0x0000 |

The CE-length parameter, according to the Bluetooth specification, is the length of the connection event.

Take an example to illustrate this scenario:

Assume a connection with interval = 100ms. that the application has put allowed 20ms of CE-length.

Here, the CE-length can be up to 100ms (100ms - 150us to be exact).

If the connection is maintained for 5 minutes, there could be 10*60*5 = 3000 connection-intervals.

The CE-length need not maintained constant during all the 3000 connection events.

Here are the typical cases that determine the value of CE-length:

(1) No data packets exchanged. we are just maintaining time and frequency synchronization. In this case, only a packet pair will be exchanged every connection interval. Here, CE-length = 1.

(2) Average of 10 packets to be sent per connection event.

We can pump data in multiple ways here:

2.1: Send data at uniform rate : In this case, the CE-length will be enough to accommodate 10 packets, which will take about 7ms. As this is less than application enforced limit of 20ms, we can comfortably push all the 10 data packets in this connection interval. So data will be pumped to the other BT device at the same rate as is received from my application.

2.2: Can send data in bursts. Assume that we accumulate data for 1 second and pump out at the end of 1 second(this is not done by our Bluetooth stack, the application needs to buffer the data). So, at 10th connection interval, we have 100 packets accumulated. We are now ready to pump this data. 100 packets take about 70 ms. This is above the application enforced 20ms. So, the hardware can pump data that can fill up 20ms. The remaining data will be deferred to the next connection interval.

So, in this case, you would see a CE-length spread over time like this (Per connection interval):

0,0,0,0,0,0,0,0,0,0, 20,20,20,10,0,0,0,0,0,0, 20,20,20,10,0,0,0,0,0,0,
20,20,20,10,0,0,0,0,0,0,

and so on.

(3) We are receiving data at the same rate as in (2). This case is to honor data sent by the other BT-device by giving it more time in the current connection interval.

In (2) and (3) you will see non-empty packets either transmitted or received. We can also utilize the CE-length for different reasons:

(4) A transaction is in progress, and we are expecting a response packet very soon. In this case, we may be exchanging only empty packets now, and in the next few packet-pairs.

In this case, you will the CE-length to be large, and a non-empty packet may not be exchanged in all the slots.

Access address (lower) register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x88 | 0x110 | PDU_ACCESS_ADDR_L_REGISTER | RW | Access address bits 15:0 for the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| PDU Access Address Lower bits | 15:0 | This field defines the lower 16 bits of the access address for each Link layer connection between any two devices. | 0x0000 |

Access address (upper) register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x8A | 0x114 | PDU_ACCESS_ADDR_H_REGISTER | RW | Access address bits 32:16 for the connection. The connection index register must be programmed with index of the connection, before programming the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| PDU Access Address Lower bits | 15:0 | This field defines the higher 16 bits of the access address for each Link layer connection between any two devices. | 0x0000 |

## Connect Event Instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x8C | 0x118 | CONN_CE_INSTANT | RW | This is the instant used for connection update procedure and channel map update procedure. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Ce_instant | 15:0 | This is the value of the free running Connection Event counter when the new parameters of ‘connection update’ and/or ‘Channel map update’ will be effective.<br>Range : 0x0000 to 0xFFFF | 0x0000 |

## Connect Event Counter register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x92 | 0x124 | CONN_CE_COUNTER | RO | This is the free running counter, connEventCounter as defined by Bluetooth spec. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| connectionEventCounter | 15:0 | Firmware will read the instantaneous Event counter from this register, during connection update and channel map update procedure. Firmware will use this value to calculate the instant from which the new parameters (for connection update and channel map update) will be effective. | 0x0000 |

## Connection configuration & status register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x8E | 0x11C | CE_CNFG_STS_REGISTER | RW | Connection specific configuration and status information register.<br>This register facilitates the pause/resume mechanism of data PDUs by LL , typically used during “enable encryption” procedure.<br>In case multiple connections are supported, “connection index register” has to be written by the f/w before programming/reading this register | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| current_pdu_index | 15:12 | Read Only field. The index of the transmit packet buffer that is currently in transmission/ waiting for transmission. | 0 |
| Reserved | 11 | Reserved for future use. | 0 |
| conn_active | 10 | Read Only field . This bit is ‘1’ whenever the connection is active. | 0 |
| force_nesn0 | 9 | not used | 0 |
| pause_data | 8 | Pause data. 1 – pause data, 0 – do not pause.<br>Pause the data transfer on the connection. The current_pdu_index in | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | hardware does not move to next index until pause_data is cleared. | |
| map_index/curr_index | 7 | Mixed info field. Written by firmware to select the channel map register set to be used by hardware for this connection. 1 – use channel map register set 1. 0 – use channel map register set 0.<br>When firmware reads this field, it returns the current map index being used in hardware. | 0 |
| md | 6 | MD bit set to '1' indicates device has more data to be sent. | 0 |
| mas_slv | 5 | mas_slv bit set to '1' indicates that device is configured as a master or a slave.<br>1 – master, 0 – slave. | 0 |
| data_list_head_up | 4 | Update the first packet buffer index ready for transmission to start/resume data transfer after a pause.<br>The bit must be toggled every time the firmware needs to indicate the start/resume. This requires a read modify write operation. | 0 |
| Data_list_index/ last_ack_index | 3:0 | Data list index for start/ resume. This field must be valid along with data_list_head_up and indicate the transmit packet buffer index at which the data is loaded.<br>The default number of buffers in the IP is 5,but may be customized for a customer. The buffers are indexed 0 to 4.<br>Hardware will start the next data transmission from the index indicated by this field. | 0 |

## Next CE instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x90 | 0x120 | NEXT_CE_INSTANT | RO | 16-bit internal reference clock value at which the next connection event will occur on a connection. The connection index register must be programmed with index of the connection, before reading the register. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Instant | 15:0 | 16-bit internal reference clock value at which the next connection event will occur on a connection. The connection index register must be programmed with index of the connection, before reading the register. | 0x0000 |

## Connection parameter 1 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xF4 | 0x1E8 | CONN_PARAM1 | RW | Connection parameters like sca, hop_increment and crc_init exchanged in connect_request of this connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| crc_init[7:0] | 15:8 | This field defines the lower byte (7:0) of the CRC initialization vector. | 0 |
| hop_increment[4:0] | 7:3 | Hop increment for connection channel. | |
| sca[2:0] | 2:0 | Sleep Clock accuracy value. | |

## Connection parameter 2 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xF6 | 0x1EC | CONN_PARAM2 | RW | Connection parameter crc_init bits 24:7 exchanged in connect_request of this | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | connection. The connection index register must be programmed with index of the connection, before reading the register. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| crc_init[23:8] | 15:0 | This field defines the upper two bytes (23:8) of the CRC initialization vector. | 0x0000 |

## Connection Update New Interval

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1D2 | 0x3A4 | CONN_UPDATE_NEW_INTERVAL | RW | The connection interval that will be effective after the connection update instant. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_interval_new[15:0] | 15:0 | This register will have the new connection interval that the hardware will use after the connection update instant. Before the instant, the connection interval in the register CONN_INTERVAL will be used by hardware. | 0x0000 |

- Connection Update New Latency

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1D4 | 0x3A8 | CONN_UPDATE_NEW_LATENCY | RW | The slave latency that will be effective after the connection update instant. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Slave_latency_newl[15:0] | 15:0 | This register will have the new slave latency parameter that the hardware will use after the connection update instant. | 0x0000 |

| | | | |
|---|---|---|---|
| | | Before the instant, the connection interval in the register SLAVE_LATENCY will be used by hardware. | |

- Connection Update New Su To

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1D6 | 0x3AC | CONN_UPDATE_NEW_SU_TO | RW | The Supervision timeout that will be effective after the connection update instant. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_so_to_new[15:0] | 15:0 | This register will have the new supervision timeout that the hardware will use after the connection update instant. Before the instant, the connection interval in the register SUP_TIMEOUT will be used by hardware. | 0x0000 |

- Connection Update New slaveLatency x connInterval value

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1D8 | 0x3B0 | CONN_UPDATE_NEW_SL_INTERVAL | RW | The (slaveLatency * connInterval) value that will be effective after the connection update instant. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| sl_conn_interval_new[15:0] | 15:0 | This register will have the new SL*CI value that the hardware will use after the connection update instant. Before the instant, the connection interval in the register SL_CONN_INTERVAL will be used by hardware. | 0x0000 |

## Window Widen for Interval register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xB0 | 0x160 | WINDOW_WIDEN_INTVL | RW | Window widening value based on connection interval of the connection. The connection index register must be programmed with index of the connection, before reading the register. | 0x000A |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:12 | Not used | |
| Window_widen | 11:0 | This value defines the increased listening time for the slave.<br>The windowWidening shall be smaller than ((connInterval/2)-T_IFS us)<br>This value is calculated by firmware based on the drift, the connection interval value. The value is the unit widening value for one connection interval duration. In case of slave latency, this value is accumulated till the next anchor point at which the slave will listen. | 0x000A |

## PDU response timer register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x502 | 0xA04 | PDU_RESP_TIMER | RW | PDU response timer register. This timer is used to monitor the time to get the response for the control procedures for which timeout rules are specified in the specification. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Pdu_resp_time_val | 15:0 | This register is loaded with the count value to monitor the time to get a response for the control PDU sent to a peer device.<br>Firmware starts the timer by issuing the command, RESP_TIMER_ON, after it has queued a control PDU for transmission that requires a response.<br>If a response is received, firmware stops and clears the timer by issuing the command RESP_TIMER_OFF.<br>If this timer expires, it results in hardware closing the connection and triggering a conn_closed interrupt.<br>The *discon_status* field in the Connection status register is set with the appropriate reason.<br>Units : Milliseconds.<br>Resolution : 1.25 ms | 0 |

## Next response timeout instant register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x504 | 0xA08 | NEXT_RESP_TIMER_EXP | RO | 16-bit internal reference clock value at which the next PDU response timeout event will occur on a connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Next_resp_timer_expire | 15:0 | This field defines the clock instant at which the next PDU response timeout event will occur on a connection.<br>This is with reference to the 16-bit internal reference clock. | 0x0000 |

## Next Supervision timeout instant register

| Addr | Addr | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | |
|---|---|---|---|---|---|
| 0x506 | 0xA0C | NEXT_SUP_TO | RO | 16-bit internal reference clock value at which the next supervision timeout event will occur on a connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Next_timeout_instant | 15:0 | This field defines the clock instant at which the next connection supervision timeout event will occur on a connection.<br>This is with reference to the 16-bit internal reference clock. | 0x0000 |

## Data list SENT Status register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x94 | 0x128 | DATA_LIST_SENT_STATUS | WO | The register is used by firmware to indicate that a packet buffer is queued (loaded) with data for transmission. Firmware sets a SENT bit in hardware corresponding to the packet buffer queued with a packet for transmission. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Unused | 0 |
| Set/Clear | 8 | Used to set the SENT bit in hardware for the selected packet buffer.<br>1 – packet queued<br>When firmware has a packet to send, firmware first loads the next available packet buffer. Then the hardware SENT bit is set by writing 1 to this bit field along with the list_index field that identified the buffer index. This indicates that a packet has been queued in the data | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | buffer for sending. This packet is now ready to be transmitted.<br>The SENT bit in hardware is cleared by hardware only when it has received an acknowledgement from the remote device.<br>Firmware typically does not clear the bit. However, It only clears the bit on its own if it needs to 'flush' a packet from the buffer, without waiting to receive acknowledgement from the remote device, firmware clears BIT7 along with the list_index specified.<br>Note: This register has a different meaning in the Read-path | |
| List_index | 7:0 | Indicates the buffer index for which the SENT bit is being updated by firmware.<br>The default number of buffers in the IP is 5. The index range is 0-4. | 0 |

## Data list ACK update register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x96 | 0x12C | DATA_LIST_ACK_STATUS | WO | Clear ACK indication for the packet, as reported by link layer hardware. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Unused | 0x0000 |
| Set/clear | 8 | Firmware uses the field to clear the ACK bit in the hardware to indicate that the acknowledgement for the transmit packet has been received and processed by firmware.<br>Firmware clears the ACK bit in the hardware by writing in this register only after the | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | acknowledgement is processed successfully by firmware.<br>For clearing ack for a packet transmitted in fifo-index : '3', firmware will write '3' in the 'list-index' field and set this bit (BIT7) to 0.<br>This is the indication that the corresponding packet buffer identified by List-Index is cleared of previous transmission and can be re-used for another packet from now on.<br>The ACK bit in hardware is set by hardware when it has successfully transmitted a packet.<br>Note: This register has a different meaning in the Read-path | |
| List_index | 7:0 | Indicates the buffer index for which the ACK bit is being updated by firmware.<br>The default number of buffers in the IP is 5. The index range is 0-4. | 0 |

## Data list SENT status register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x94 | 0x128 | DATA_LIST_SENT_STATUS | RO | Status of SENT bit of all the transmit buffers available in the hardware. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Unused | 0x0000 |
| tx_sent | 7:0 | The bits in this field indicate the status of the SENT bit in the hardware for each packet buffer. The bit values are<br>1 – queued<br>0 – no packet / packet ack received by hardware | 0 |

| | | | |
|---|---|---|---|
| | | Example1: If the read value is : 0x03, then packets in buffer 0 and buffer 1 are in the queue to be transmitted. All the other FIFOs are empty or hardware has cleared them after receiving acknowledgement. | |

## Data list ACK update register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x96 | 0x12C | DATA_LIST_ACK_STATUS | RO | Status of ACK bit of all the transmit buffers available in the hardware. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Unused | 0x0000 |
| tx_ack | 7:0 | If a particular bit is set, then the packet in the selected buffer has been transmitted (at least once) by the hardware and hardware is waiting for acknowledgement.<br>Example1 : If the read value is : 0x03, then packets in FIFO-0 and FIFO-1 are acknowledged by the remote device. These acknowledgements are pending to be processed by firmware.<br>Example2 : If the read value is : 0x02, then packet FIFO-1 is acknowledged by the remote device. This acknowledgement is pending to be processed by firmware.<br>Note : This register has a different meaning in the Write-path. | 0 |

The SENT bit and ACK bit have to be taken together to understand the meaning of packet status. The table below describes how the two bits are sequentially updated by either hardware/firmware to complete one data transmission.

| SENT | ACK | Description |
|---|---|---|
| 0 | 0 | Buffer is empty. No packet is queued in the buffer |
| 1 | 0 | Packet is queued by firmware. |
| 1 | 1 | Packet is transmitted by hardware. Hardware is waiting for acknowledgement. |
| 0 | 1 | Hardware has received ACK. Firmware has not yet processed the ACK. |
| 0 | 0 | Firmware has processed the ack. The buffer is again empty. |

## Device Data List Status Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x290 | 0x520 | DEVICE_DATA_LIST_STATUS | RO | Status of data packets queued and sent. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Unused | 0x0000 |
| device_data_list_status | 7:0 | These bits the status of data pecked queued and sent at device level. If the particular bit is set to ‘1’ indicates that there either pending PDU in the queue or pending ACK to be cleared. Based on the status of this register FW will queue next PDU.<br><br>Note : This register is used only in shared connection tx_fifo. | 0 |

### Data list Index 0 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x98 | 0x130 | LIST_INDEX0 | WO | Reserved for future use. | 0x0000 |

### Data list Index 1 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x9A | 0x134 | LIST_INDEX1 | WO | Reserved for future use. | 0x0000 |

## Data buffer descriptor 0-7 register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x200 - 0x21E | 0x400- 0x43C | DATA_MEM_DESCRIPTOR0 - DATA_MEM_DESCRIPTOR15 | RW | Descriptor for packet stored in the each of the transmit buffer which includes the packet specific information like length and LLID. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| conn_handle | 15:8 | Unused. Reserved for future use. | 0x0 |
| Enc | 7 | Unused. Reserved for future use. | |
| Data_length | 6:2 | This field indicates the length of the data packet.<br>Range: 0x0 to 0x1F. | 0 |
| LLID | 1:0 | The LLID indicates whether the packet is an LL Data PDU or an LL Control PDU.<br>00b= Reserved.<br>01b=LL Data PDU: Continuation fragment of an L2CAP message or an Empty PDU.<br>10b=LL Data PDU: Start of an L@CAP message or a complete L2CAP message with no fragmentation.<br>11b=LL Control PDU. | |

## Feature Configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x508 | 0xA10 | LLH_FEATURE_CONFIG | RW | Enabling/Disabling of different features implemented in connection. | 0x000A |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 7:2 | RFU | 0 |
| llf_sl_dsm_ww_ctrl | 3 | Control bit to enable or disable the wakeup delay when slave latency is enabled. | 1 |
| slave_latency_en_sts | 2 | Read Only . Firmware reads this bit to know the status of slave latency enable. | 0 |
| llf_sl_dsm_en | 1 | LLF assisted slave latency dsm feature is enabled by setting this bit.<br>When slave latency is enabled, this feature enables the slave to awaken out of DSM when it is necessary for the slave to receive a packet. | 1 |
| Quick_transmit | 0 | Quick transmit feature in slave latency is enabled by setting this bit.<br>When slave latency is enabled, this feature enables the slave to transmit in the immediate connection interval, in case required, instead of waiting till the end of slave latency | 0 |

## Adaptive Window Widening Config

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x50A | 0xA14 | AWW_CONFIG | RW | Enabling/Disabling of various AWW specific functions and AWW configurable parameters. | 0xE244 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| aww_error_tolerence_en | 15 | This bit enables/disables the access code bit error tolerance functionality.<br>1 – enable<br>0 - disable | 1 |
| aww_tolerence_threshold | 14:10 | This value sets the lower bound on tolerance of bit errors in access code to | 0x18 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | overlook AWW history recapture. Eg., 0x18 implies 32-24 = 8 bit errors are tolerable. | |
| aww_alg_auto_switch_en | 9:9 | This bit is used to enable/ disable the functionality of automatic switching between the AWW algorithms i.e The Conservative Mode and Greedy Mode.<br>1 – Enable<br>0 – Disable | 1 |
| aww_alg_sel | 8:8 | IF the AWW algorithm auto switch functionality is disabled, this bit is used to manually select the AWW algorithm to be used.<br>1 – Greedy Mode<br>0 – Conservative Mode | 0 |
| constancy_cnt_init_val | 7:4 | This value defines the number of connection events for which the adaptive window module checks if master anchor points variations fall within set threshold before switching from conservative mode to greedy mode.<br>Valid Range : (0x1,0xf) | 0x04 |
| Obs_ce_cnt_init_val | 3:0 | This value defines the number of connection events for which the adaptive window module performs measurements of master anchor point swing before it applies the optimized window in accordance with conservative mode aww algorithm.<br>Valid Range : (0x1,0xf) | 0x04 |

## Slave Window Adjustment

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x50C | 0xA18 | SLV_WIN_ADJ | RW | Programmable adjust value to add to the calculated window widen value. This allows the firmware to add flexibility to calculated value of drift - to compensate for any underestimated drift in the | 0x0010 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | manufacturer specification of crystal drift. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:11 | Unused | 0x00 |
| Slv_win_adj | 10:0 | Window Adjust value. This value is added to the calculated slave window widening value to be used as final window widen value. | 0x10 |

## slaveLatency x connInterval value

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x50E | 0xA1C | SL_CONN_INTERVAL | RW | Programmable Register which holds the (slaveLatency * connInterval) value for the connection. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| sl_conn_interval_val | 15:0 | This field defines the (SL * CI) product for the ongoing connection. This value is used in calculation of next connection instant during slave latency.<br>Unit is 1.25ms | 0x0000 |

## Cumulative hop increment register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x25A | 0x4B4 | SL_CONN_HOP_INCREMENT | RW | This register holds the value of cumulative hop increment corresponding to the Slave latency. | 0x00 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Sl_conn_hop_increment | 5:0 | Firmware writes the cumulative hop increment value corresponding to the slave latency upon connection creation complete before enabling Slave latency.<br>Range of the value is 0 to 36. | 0x00 |

## Cumulative window widen register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x25C | 0x4B8 | SL_CONN_WINDOW_WIDEN | RW | This register holds the value of cumulative window widen value corresponding to the slave latency. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sl_conn_window_widen | 15:0 | Firmware writes the cumulative window widen value corresponding to the slave latency upon connection creation complete before enabling Slave latency.<br>Unit is in microseconds. | 0x0000 |

## Connection Update New slave latency hop increment register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x25E | 0x4BC | CONN_UPDATE_NEW_SL_HOP_INCREMENT | RW | The (slaveLatency * hopIncrement) value that will be effective after the connection update instant. | 0x00 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sl_hop_increment_new[5:0] | 5:0 | This register will have the new SL_New * hopIncrement value that the hardware will use after the connection update instant. Before the instant, the hop increment in | 0x00 |

| | | | | |
|---|---|---|---|---|
| | | the register SL_CONN_HOP_INCREMENT will be used by hardware.<br>Range of the value is 0 to 36. | | |

## Connection Update New slave latency window widen register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x260 | 0x4C0 | CONN_UPDATE_NEW_SL_WIN_WIDEN | RW | The cumulative window widen value corresponding to the new slave latency after the update instant. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Sl_window_widen_new[15:0] | 15:0 | The cumulative window widen value after the connection update instant. Before the instant, the window widen in the register SL_CONN_WINDOW_WIDEN will be used by hardware.<br>Unit is in microseconds. | 0x0000 |

## Previous connection instant register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x262 | 0x4C4 | PREVIOUS_CONN_INSTANT | RO | Reference BT clock value of the previous anchor point. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Previous_conn_instant | 15:0 | Read only register.<br>Holds the value of the reference BT clock value at the previous connection event that LLH participated. | 0x0000 |

## Previous Connection instant event count

| Addr | Addr | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | |
|---|---|---|---|---|---|
| 0x26E | 0x4DC | PREVIOUS_CONN_CE_COUNT | RO | Connection event count of the previous anchor point. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| previous_conn_ce_count | 15:0 | Read only register.<br>Holds the value of the event counter at the previous connection event that LLH participated. | 0x0000 |

## Previous Connection channel

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x270 | 0x4E0 | PREVIOUS_CONN_CH | RO | Connection channel of the previous anchor point. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| previous_conn_ch | 15:0 | Read only register.<br>Holds the value of the channel at the previous connection event that LLH participated. | 0x0000 |

## Slave latency connection event counter increment register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x264 | 0x4C8 | CE_COUNT_INCREMENT_N | RW | This register holds the value of connection event count N. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Ce_count_increment_n | 15:0 | Firmware programs the value of connection event count N corresponding to the elapsed time (from the previous anchor point) and expected next anchor point (anchor | 0x0000 |

| | | | |
|---|---|---|---|
| | | point when it wants to exit slave latency). | |

## Slave latency connection interval increment register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x266 | 0x4CC | CONN_INTERVAL_INCREMENT_N | RW | This register holds the value of cumulative connection interval for the connection event count N. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_interval_increment_n | 15:0 | Firmware writes the cumulative connection interval for the connection event count N corresponding to the elapsed time (from the previous anchor point) and expected next anchor point (anchor point when it wants to exit slave latency).<br>Unit is 1.25ms | 0x0000 |

## Slave latency hop increment register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x268 | 0x4D0 | CONN_HOP_INCREMENT_N | RW | This register holds the value of cumulative hop increment for the connection event count N. | 0x00 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_hop_increment_n | 5:0 | Firmware writes the cumulative hop increment for the connection event count N corresponding to the elapsed time (from the previous anchor point) and expected next anchor point (anchor point when it wants to exit slave latency).<br>Range of the value is 0 to 36. | 0x00 |

## Slave latency window widen increment register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x26A | 0x4D4 | CONN_WINDOW_WIDEN_N | RW | This register holds the value of cumulative window widen for the connection event count N. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Conn_window_widen_n | 15:0 | Firmware writes the cumulative window widen for the connection event count N corresponding to the elapsed time (from the previous anchor point) and expected next anchor point (anchor point when it wants to exit slave latency).<br>Unit is in microseconds. | 0x0000 |

## Slave latency new connection instant register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x26C | 0x4D8 | ANCHOR_POINT_INCREMENT_N | RW | This register holds the value of the distance for connection event count N corresponding to the elapsed time. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Anchor_point_increment_n | 15:0 | Firmware write the value of the distance for the connection event count N corresponding to the elapsed time from the previous anchor point.<br>It calculate the distance as<br>(N * CI) – (((SL_CONN_WINDOW_WIDEN_N) slots) + 1)).<br>Unit is 0.625ms | 0x0000 |

| | | | | |
|---|---|---|---|---|
| | | LLH use this value to check the validity the instant passed for the new derived anchor point with reference to the PREVIOUS_INSTANT. | | |

## **LE Ping timer address register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x510 | 0xA20 | CONN_PING_TIMER_ADDR | RW | The register used to configure the LE Authenticated payload Timeout (LE APTO) which is the Maximum amount of time specified between packets authenticated by a MIC.<br>This value of ping timer is in the order of 10ms, valid range 0x1 ~ 0xFFFF | 0x0000 |

## **LE Ping connection timer offset

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x512 | 0xA24 | CONN_PING_TIMER_OFFSET | RW | The value of ping timer nearly expired offset in the order of 10ms, valid range 0x0 ~ 0xFFFF. This is the time period after which the ping timer nearly expired interrupt is generated. | 0x0000 |

## **LE Ping timer next expiry instant

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x514 | 0xA28 | CONN_PING_TIMER_NEXT_EXP | RO | The value of ping timer next expiry instant in the terms of native clock value (least 16 bit value of the 17 bit ping counter).<br>This together with CONN_PING_TIMER_NEXT_EXP_WRAP will provide the correct status of ping timer duration. | 0x0000 |

## **LE Ping timer next expiry wrap count

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x516 | 0xA2C | CONN_SEC_CURRENT_WRAP | RO | This register holds the current position of the Ping timer. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RESERVED | 15:9 | RFU | 0x0 |
| CONN_SEC_NEAR_WRAP | 8:5 | This field provides the time offset of the nearly expired event from the authentication payload timeout event. This offset is in the order of 40959.375 ms and specifies the time offset after starting the ping timer the nearly expired event will be generated.<br>Time= N*40959.375 ms. | 0x0 |
| CONN_SEC_CURRENT_WRAP | 4:1 | This field provides the current position of the ping timer and the value is in the order of 40959.375 ms.<br>Time= N*40959.375 ms<br>For Example if the APTO configured in 655,350 ms and this field returns 10, it means another 6 more units are remaining for the APTO event to be generated. | 0x0 |
| WRAP_VALID | 0 | This field will be ‘1’ from nearly expired event to the authenticated payload timeout or till the next reload of the ping timer. | 0x0 |

## #Connection Arbiter Parameter register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x102 | 0x204 | CONN_ARB_PARAMS | RO | Register holds the highest priority request to the connection arbiter from the connection engines. | 0xF |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:4 | Not used | -- |
| priority_pointer | 3:0 | Pointer for the highest priority request to the connection arbiter. | 0xF |

## Conn_SUTO_CI_Ratio register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x104 | 0x208 | CONN_SUTO_CI_RATIO | RW | Firmware programs this register with connection supervision timeout to connection interval ratio at the time of connection creation/ connection update. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:13 | Not used | 0x0 |
| conn_suto_ci_ratio[12:0] | 12:0 | This register will have the ratio of the connection supervision timeout to the connection interval as programmed by the firmware before a connection creation or a connection update. The value is used by hardware to update the priority of a connection link that is nearing connection timeout due to packet miss. | 0x0000 |

## Connection Priority Config register

| Addr | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x106 | 0x20C | CONN_PRIORITY_CNFG | RW | This register is used by the firmware to override the hardware priority and set the priority of each connection link in the connection arbiter. The value of Zero has the least priority. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| priority_sel | 15 | If this bit is set to '1' connection arbiter will use the firmware programmed priority set in this register for each connection engine. | 0 |
| RFU | 14:12 | Not used | 0x0 |
| fw_priority3 | 11:9 | Firmware programmed priority for connection engine 3. | 0x0 |
| fw_priority2 | 8:6 | Firmware programmed priority for connection engine 2. | 0x0 |
| fw_priority1 | 5:3 | Firmware programmed priority for connection engine 1. | 0x0 |
| fw_priority0 | 2:0 | Firmware programmed priority for connection engine 0. | 0x0 |

## Connection Update New SUP_TO to CI ratio

| Addr | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1DA | 0x3B4 | CONN_UPDT_NEW_SU_CI_RATIO | RW | The Supervision Timeout to conn interval ratio that will be effective after the connection update instant. Firmware programs this register along with other connection update parameter registers. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:13 | Not used. | 0x0 |
| conn_suto_ci_ratio_new | 12:0 | This register will have the new supervision timeout to connection interval ratio that the hardware will use after the connection update instant. Before the instant, the value in the register CONN_SUTO_CI_RATIO will be used by hardware. | 0x0000 |

## #Slave Window Offset Full register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x1DC | 0x3B8 | SL_WIN_OFF_FULL | RW | This register is programmed | 0x0000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | with the maximum possible window widen value by the firmware when dsm entry during slave latency feature is enabled. This value is used by the hardware to calculate the next ce instant being reported to firmware. | | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:6 | Not used. | 0x0 |
| sl_window_off_full | 5:0 | Value of maximum possible window widen programmed by the firmware when slave latency dsm feature is enabled. | 0x0000 |

# # Connection RSSI register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset | |
|---|---|---|---|---|---|---|
| 0x518 | 0xA30 | CONN_RX_RSSI | RO | This connection register holds the RSSI value of the last good (empty/non-empty) packet received by the specific connection. | 0x0000 | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| conn_rx_rssi | 15:0 | RSSI during the last good packet received in the specific connection. | 0x0000 |

# Connection Rx Memory read enable register

| Addr | Addr | Register Name | RW | Description | Reset | |
|---|---|---|---|---|---|---|

*Table continues on the next page...*

| 16-bit | 32-bit | | | | | |
|---|---|---|---|---|---|---|
| 0x522 | 0xA44 | LLH_MEM_READ_ENABLE_CONTROL_REG | WO | This register is used by the firmware to control read access to the connection Rx memory. | 0x0000 | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:14 | Not Used | 0x0 |
| conn_rxmem_rd_ctrl | 0 | When this bit is '1', connection Rx memory is enabled. | 0 |

## Tx memory configuration register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x27A | 0x4F4 | TXMEM_CONFIG | RW | This register fields can be set to configure the size of the data PDUs. | 0x700 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| RFU | 15:12 | Reserved | 0x0 |
| Number_of_packets_minus_1 | 11:9 | Maximum Number of Data PDUs in Tx FIFO. | 0x03 |
| Max_data_pdu_size | 8:0 | Maximum data packet length. | 0x100 |

## Tx memory configuration register 2

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x27C | 0x4F8 | TXMEM_CONFIG_2 | RW | This register fields can be set to configure the base address of the Control PDU. | 0x300 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| RFU | 15:11 | Reserved | 0x0 |
| Max_total_data_size | 10:0 | Total size of Data PDUs.<br>i.e. Maximum Number of Data PDUs * Maximum Data Packet Length as configured in TXMEM_CONFIG register.<br>It is base address of first Control PDU. | 0x300 |

### 64.5.1.1.4 Test and Debug Registers

Test and Debug Register Descriptions for the Bluetooth Link Layer

DTM control register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xB8 | 0x170 | LE_RF_TEST_MODE | RW | LE Direct Test Mode (DTM) configuration and control register. Used to control the direct test mode (DTM) operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:10 | Reserved for future use | 0 |
| dtm_conttx_en | 13 | Write:<br>1 – To enable continuous transmit mode<br>0 – To disable continuous transmit mode | 0 |
| Reserved | 12:10 | Not Used | |
| Pkt_payload[2:0] | 9:7 | Payload type as per the HCI parameter.<br>0x00 Pseudo-Random bit sequence 9<br>0x01 Pattern of alternating bits ‘11110000<br>0x02 Pattern of alternating bits ‘10101010’<br>0x03 Pseudo-Random bit sequence 15<br>0x04 Pattern of All ‘1’ bits<br>0x05 Pattern of All ‘0’ bits | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | 0x06 Pattern of alternating bits '00001111<br>0x07 Pattern of alternating bits '0101'<br>0x08-0xFF Reserved for future use | |
| Test_type | 6 | Mixed Info Field.<br>Read:<br>1 – Indicates DTM test ON<br>0 – Indicates DTM test OFF<br>Write:<br>1 – To enable continuous receive mode<br>0 – To disable continuous receive mode | 0 |
| Test_frequency[5:0] | 5:0 | N = (F – 2402) / 2<br>Range: 0x00 – 0x27. Frequency Range : 2402 MHz to 2480 MHz | 0 |

## DTM receive packet count register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xBA | 0x174 | DTM_RX_PKT_COUNT | RO | Count of the number of LE packets received when device is configured in receive test mode. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rx_packet_count[15:0] | 15:0 | Number of packets received in receive test mode. | 0 |

## Connection channel test control register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xBC | 0x178 | CONN_TEST_CONTROL | RW | Connection test control register. To introduce test behavior in the operation of connection. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Reserved for future use | 0 |
| Nonempty_pdu_rxnack | 7 | NACK the received packet, if rx packet is a NON-EMPTY PDU.<br>• Always NACK (no acknowledgement)<br>0 - No change from normal behavior | 0 |
| Empty_pdu_rxnack | 6 | NACK received packet, if rx packet is an EMPTY PDU.<br>• Always NACK<br>0 - No change from normal behavior | 0 |
| Nonempty_pdu_retx | 5 | Retransmit previous transmitted non-empty PDU irrespective of received NESN (whether acknowledged or not).<br>1 – Always retransmit<br>0 – No change from normal behavior | 0 |
| empty_pdu_retx | 4 | Retransmit previous transmitted empty PDU irrespective of received NESN (whether acknowledged or not).<br>1 – Always retransmit<br>0 – No change from normal behavior | 0 |
| Nonempty_crc_err | 3 | Cause CRC field error in transmitted non-empty PDU (only).<br>1 – Causes CRC error<br>0 – no change from normal behavior | 0 |
| empty_crc_err | 2 | Cause CRC field error in transmitted empty PDU (only).<br>1 – Causes CRC error<br>0 – no change from normal behavior | 0 |
| Nonempty_acc_err | 1 | Causes Access address error in transmitted non-empty PDU.<br>1 – Causes access address error<br>0 – no change from normal behavior | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| empty_acc_err | 0 | Causes Access address error in transmitted empty PDU.<br>1 – Causes access address error<br>0 – no change from normal behavior | 0 |

## Advertising channel test control register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xBE | 0x17C | ADVCH_TEST_CONTROL | RW | Advertising channel test control register. To introduce advertising channel test control operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:7 | Reserved for future use | 0 |
| Scan_tx_hdr | 6 | Corrupt the tx header of SCAN_REQ packet<br>1 – corrupt the address<br>0 – no change | |
| Peer_addr_err | 5 | Corrupt the peer address field in the SCAN_REQ packet<br>1 – corrupt the address<br>0 – no change | 0 |
| Rcv_txaddr_err | 4 | Corrupt the received transmit address type indication<br>1 - corrupt the tx address type (invert the bit)<br>0 – no change | 0 |
| Scnrsp_tx_err | 3 | Introduce CRC error in scan response packet.<br>1 – corrupt CRC<br>0 – no change | 0 |
| Rx_crc_err | 2 | Receive packet CRC error indication.<br>1 – Indicate CRC error of received packet, irrespective of good/bad CRC<br>0 – No change from normal behavior | 0 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Tx_crc_err | 1 | Corrupt transmit packet CRC, irrespective of packet type.<br>• Corrupt CRC<br>0 – No change in normal behavior<br>This is done by corrupting the crc_init value. | 0 |
| Tx_acc_err | 0 | Corrupt transmit packet access address.<br>• Corrupt access address<br>0 – No change in normal behavior | 0 |

## DTM Error Count

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xC0 | 0x180 | DTM_CRC_ERR_COUNT | RO | Indicates number of packets received with CRC error in the DTM mode | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dtm_crc_err_pkt_count | [15:0] | DTM CRC error packet count. Used for Debug purpose only. | 0 |

## DTM transmit packet count register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xC2 | 0x184 | DTM_TX_PKT_COUNT | RO | Count of the number of LE packets transmitted when device is configured in transmit test mode. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| tx_packet_count[15:0] | 15:0 | Number of packets transmitted in transmit test mode. Used for Debug purpose only. | 0 |

## Channel Address register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xC4 | 0x188 | CH_ADDR/ TXRX_HOP | RO | Contains value of the transmit and receive hop frequency | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| reserved | 15 | Reserved for future use | 0 |
| hop_ch_rx | [14:8] | Receive channel index. Channel index on which previous packet is received. | 0 |
| reserved | 7 | Reserved for future use | 0 |
| hop_ch_tx | [6:0] | Transmit channel index. Channel index on which previous packet is transmitted. | 0 |

## DTM Length register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xCE | 0x19C | LE_DTM_TEST_LEN | RW | LE Direct Test Mode (DTM) length register. Used to configure the payload length of direct test mode (DTM) operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:8 | Reserved for future use | 0 |
| dtm_len | 7:0 | 0x00-0xFF (except 0x3F) Length in bytes of payload data in each packet<br>0x3F – Continuous Transmit mode. | 0 |

## Divider Value Register for SCA

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0x500 | 0xA00 | DIV_VAL_ADDR | RW | This value will go to divide by N module to derive various sleep clock frequencies | 0x1F3C |

| Reference Clock | Divider Value | Output Sleep Frequency(KHz) | PPM | SCA | |
|---|---|---|---|---|---|
| | Decimal | | | | Hex |
| 16.384 KHz | 7808 | 1E80 | 16.39351 | 580.59674 | |
| 7809 | 1E81 | 16.39135 | 448.73375 | 0 | |
| 7810 | 1E82 | 16.38920 | 317.56129 | 0 | |
| 7811 | 1E83 | 16.38722 | 196.25739 | 1 | |
| 7812 | 1E84 | 16.38501 | 61.54619 | 4 | |
| 7813 | 1E85 | 16.38287 | -69.19713 | 4 | |
| 7814 | 1E86 | 16.38097 | -185.16635 | 1 | |
| 7815 | 1E87 | 16.37870 | -323.70446 | 0 | |
| 7816 | 1E88 | 16.37668 | -446.48984 | 0 | |
| 16 KHz | 7996 | 1F3C | 16.00811 | 507.13706 | |
| 7997 | 1F3D | 16.00610 | 380.94506 | 0 | |
| 7998 | 1F3E | 16.00414 | 258.94704 | 0 | |
| 7999 | 1F3F | 16.00221 | 137.93902 | 2 | |
| 8000 | 1F40 | 15.99992 | -4.79998 | 7 | |
| 8001 | 1F41 | 15.99786 | -133.74211 | 2 | |
| 8002 | 1F42 | 15.99585 | -259.13283 | 0 | |
| 8003 | 1F43 | 15.99392 | -380.01553 | 0 | |
| | 8004 | 1F44 | 15.99192 | -505.02482 | |

A macro ‘SCA_DRIFT “is used to select the clock source for the Sleep Clock.

When the SCA_DRIFT macro is enabled then the sleep clock source is derived from the 64MHz based on the register DIV_VAL_ADDR setting. Otherwise sleep clock source is the on board 32.768 KHz

From 64 MHz of RF clock either 16 KHz or 16.384 KHz clock is derived using DCM (Digital Clock Manager, clock multiplier by 2) and a clock divider (divide by N). 64 MHz clock is given to DCM and this multiplied clock of 128 MHz is given to divide by N module. With 64 MHz clock only, it was not possible to cover lower, middle and upper range of required SCA (0 to 7) that led to use a DCM to get a higher clock and desired SCA values.

Clock divider value is given into a Programmable Read/Write register DIV_VAL_ADDR, whose address is 0x500 for 16 bit address bus.

Config.xml file is having all power up configuration parameters for BlueLitE IP. This will be loaded into Hardware after Reset.

For the validation, In order to introduce SCA other than '0' which is by default configuration of BlueLitE IP, one has to change corresponding "sca" field in config.xml file also. E.g. if one has selected SCA of 4 from table 2 given, then DIV_VAL_ADDR has to be programmed with 0x1E84 for positive ~61 PPM or 0x1E85 for negative ~69 PPM and "sca" field in config.xml has to be updated with 0x04.

Whenever SCA value is chosen from sleep clock frequency of 16 KHz, at that time internal adjustment for sleep clock frequency is not required. For that "clock_config" field in config.xml has to be updated with 0xA020.

## DTM 2 wire UART *Baud rate configuration*

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x580 | 0xB00 | DTM_2WIRE_CONFIG | RW | This value will go to different baud rate generator module for DTM 2 wire UART. | 0x006F |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dtm_2wire_baud_config | 15:0 | This value will be given to baud clock generator module. For different input clock frequency to LLH, this value will be different to generate corresponding baud clock as mentioned in below table. | 0x006F |

| Radio clock | Baud rate | Configuration Value |
|---|---|---|
| 26 MHz | 1200 | 0x2A3B |
| | 2400 | 0X151E |
| | 9600 | 0X0546 |
| | 14400 | 0X0384 |
| | 19200 | 0X02A1 |
| | 38400 | 0X0152 |
| | 57600 | 0X00E1 |
| | 115200 | 0X006F |
| 64 MHz | 1200 | 0X19FA |

*Table continues on the next page...*

| | | |
|---|---|---|
| | 2400 | 0X0CFE |
| | 9600 | 0X033E |
| | 14400 | 0X0229 |
| | 19200 | 0X019E |
| | 38400 | 0X00D0 |
| | 57600 | 0X008A |
| | 115200 | 0X0042 |

Note: For lec_top (LLH + PHY) environment input frequency is 64 MHz. Value mentioned on above table for 64 MHz is w.r.t 16 MHz because we divide 64 MHz value by 4 internally. This 16 MHz derived value is given as an input to LLH and used for further calculation.

## DTM Offset Configuration Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x118 | 0x230 | DTM_OFFSET_CONFIG | RW | Contains configurable parameters for DTM function enhancement request. | 0x012D |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:11 | Reserved for future use. | |
| dtm_offset_us_buffer | 10:1 | Represents TIFS value ‘150’ in the calculation of dtm offset. | 0x96 |
| dtm_offset_en | 0 | If set to 1, DTM offset enhancement is enabled.<br>If set to 0, DTM offset enhancement is disabled. | 0x1 |

- 

## DTM Transmit Packet Configuration Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x11A | 0x234 | DTM_TX_PKT_CONFIG | RW | Configures number of DTM packets to be transmitted when DTM Tx operation is triggered and automatically | 0x0000 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | stops the DTM Tx operation after that many number of packets are transmitted. | |

| Field | Bit | Description | Reset |
|---|---|---|---|
| dtm_tx_pkt_config[15:0] | 15:0 | If this field value is non-zero, it represents number of DTM packets to be transmitted when dtm_start command is received. DTM Tx operation is automatically stopped when DTM Tx packet count equals to the configured value.<br>If this field value is zero, DTM continues to transmit packets till dtm_stop command is issued. | 0 |

## DTM Expected Receive Packet Length Register

- 

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x11C | 0x238 | DTM_EXP_RX_LEN | RW | Contains configurable parameters for DTM timeout feature. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Reserved for future use. | |
| dtm_exp_rx_length | 8:1 | Configured length value for DTM receiver. | 0x00 |
| dtm_exp_rx_len_en | 0 | If set to 1, DTM timeout feature is enabled.<br>If set to 0, DTM timeout feature is disabled. | 0x0 |

## Packet Payload Status Register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x11E | - | PKT_PAYLOAD_STATUS | RO | Contains CRC status and Payload length value of received packet. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:9 | Reserved for future use. | |
| crc_status | 8 | Contains the value of CRC status of the received packet. Updates each time when packet reception is over. | 0x0 |
| payload_length | 7:0 | Contains the value of payload length field. Updated each time when packet reception is over. | 0x00 |

## Packet RSSI Status Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x120 | - | PKT_RSSI_STATUS | RO | Contains RSSI value for the received packet. | 0x000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| pkt_rssi_status | 15:0 | Contains the RSSI value of the received packet. Updated each time when packet reception is over. | 0x0000 |

### 64.5.1.1.5 Interface Registers

Interface Register Descriptions for the Bluetooth Link Layer

Receive trigger control register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0XFC | 0X1F8 | RECEIVE_TRIG_CTRL | RW | Receivers trigger control register. The threshold value for access address match, the access match trigger timeout is programmed to this register by firmware. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| Acc_trigger_timeout | 15:8 | If access address match does not occur then within this time from the start of receive operation, the receive operation times out and stops. An internal counter value of 1usec resolution is continuously compared with the value programmed.<br>Max value :0Xff | 0 |
| tim_adj_trig | 7 | Not used –RFU | 0 |
| | 6 | RFU | |
| Acc_trigger_threshold | 5:0 | Access address match threshold value. Number of bits of access address that should match with the expected access address to trigger an access code match.<br>Max value : 32 (for 32-bit access address)<br>Lower values may be programmed for bad radios or channels but care must be taken to ensure there are no 'false' matches due to reduced number of bits required to match.<br>Note : BQB spec mandates this to be 32. So ensure that the standard versions have 32. For debugging or RF tuning, we can experiment with smaller values. | 0 |

## Transmit/Receive data delay Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xC8 | 0x190 | TX_RX_ON_DELAY | RW | Controls the delay in link layer from internal reference point, to start the transmit and receive operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Txon_delay[7:0] | 15:8 | Transmit delay – Delay from internal trigger of transmit to transmission of first bit on air. | 0x00 |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| | | It is used to control the T_IFS. The delay is in resolution of 1 microsecond. | |
| Rxon_delay[7:0] | 7:0 | Receive delay – Delay from start of receive to expected first bit of receive packet at the controller. Used to control the turn on time of radio to optimize on power. The delay is in resolution of 1 microsecond. | 0x00 |

## Transmit/receive synthesizer delay register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0xCC | 0x198 | TX_RX_SYNTH_DELAY | RW | Controls the link layer behavior after waiting for RF synthesizer settling time. The delay relates to the PLL settling time, from the trigger to start transmit or receive operation. The action to be taken at the expiry of this delay is specific to each radio (e.g. we may set an external control pin to the radio). | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Tx_synth_delay[7:0] | 15:8 | Transmit synthesizer delay – Delay from start of transmit to stabilization of PLL. This may be used if any radio specific operation needs to be performed after PLL stabilization before bit transmission. | 0x00 |
| Rx_synth_delay[7:0] | 7:0 | Receive synthesizer delay – Delay from start of receive to receiver synthesizer stability. | 0x00 |

## RSSI Register

| Addr 16-bit | Addr 32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|

*Table continues on the next page...*

| | | | | | |
|---|---|---|---|---|---|
| 0xD8 | 0x1B0 | RADIO_RSSI_READ | RO | Indicates the RSSI value read from the radio for the last packet received.<br>Mixed-info-field/register | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Rssi_data | 15:0 | Indicates the RSSI value read from the radio for the last packet received. The meaning is specific to the RF IC used.<br>Mixed-info-field/register | 0x0 |

## RF_ACTIVE_PERIOD register

| Addr<br>16-bit | Addr<br>32-bit | Register Name | RW | Description | Reset |
|---|---|---|---|---|---|
| 0x5A | 0x0B4 | rf_active_period | RW | Register to specify the time offset before the start of a transceiver operation (i.e., a Tx/Rx) at which "RF_ACTIVE" output signal shall be asserted by the LLH. The purpose of the RF_ACTIVE signal is to indicate to the host of any upcoming transceiver activity. The host may schedule not to do any power-intensive operations during this time to reduce the system peak power. The time offset is specified in the units of BT slots (625us). Note: RF_ACTIVE signal will be asserted at the specified time offset before a transceiver operation and remain asserted till the end of the operation. | 0x0000 |

| Field | Bit | Description | Reset |
|---|---|---|---|
| Reserved | 15:7 | RFU | xx |
| rf_active_polarity | 6 | Polarity bit for rf_active<br>0: active high<br>1: active low. | |

*Table continues on the next page...*

| | | | |
|---|---|---|---|
| rf_active_slots | 5:0 | Number of BT slots (625 us) in advance of the actual start of the slot (tx/Rx) to assert rf_active. ”rf_active” is always de-asserted when the value is 6’h0 (all zeros) and always asserted when the value is 6’h3F(all ones) Therefore the Min. value is .625ms<br>and the Max value is 62*. 625ms. | 0 |

## 64.5.1.1.6 Instruction Set

Instruction Set Description for the Bluetooth Link Layer

| Command | Opcode | Description |
|---|---|---|
| START_ADV | 0x40 | Start Advertiser operation.<br>The associated Advertiser configuration registers are programmed before the command is issued. |
| STOP_ADV | 0x41 | Stop advertiser operation. |
| START_SCAN | 0x42 | Start scanner operation.<br>The associated configuration registers must be programmed before the command is issued. |
| STOP_SCAN | 0x43 | Stop the scanner operation. |
| START_INIT | 0x44 | Start connection creation operation.<br>The associated configuration registers must be programmed before the command is issued. |
| STOP_INIT | 0x45 | Cancel connection creation operation. |
| DTM_TX_START | 0x46 | Start Direct Test Mode Transmit Test.<br>The associated configuration registers must be programmed before the command is issued. |
| DTM_RX_START | 0x47 | Start Direct Test Mode Receive Test.<br>The associated configuration registers must be programmed before the command is issued. |
| DTM_STOP | 0x48 | Stop Direct Test Mode. |
| UPDATE_CHAN_MAP | 0x4B | Update channel map for the connection. |
| UPDATE_CONN_INSTANT | 0x4C | Start connection update procedure for the connection. |
| PACKET_RECEIVED | 0x4D | Indicates a received connection packet is read by firmware from connection receive FIFO. |
| ENTER_DSM | 0x50 | Enter deep sleep mode. |

*Table continues on the next page...*

| | | |
|---|---|---|
| ENTER_SM | 0x51 | Enter sleep mode. |
| EXIT_SM | 0x52 | Exit sleep mode |
| ENC_CLK_ON | 0x53 | Turn on clock to encryption block |
| ENC_CLK_OFF | 0x54 | Turn off clock to encryption block |
| ADV_CLK_ON | 0x55 | Turn on clock to advertiser block in NAP mode. |
| ADV_CLK_OFF | 0x56 | Turn off clock to advertiser block in NAP mode. |
| SCAN_CLK_ON | 0x57 | Turn on clock to scanner block in NAP mode. |
| SCAN_CLK_OFF | 0x58 | Turn off clock to scanner block in NAP mode. |
| INIT_CLK_ON | 0x59 | Turn on clock to initiator block in NAP mode. |
| INIT_CLK_OFF | 0x5a | Turn off clock to initiator block in NAP mode. |
| CONN_CLK_ON | 0x5b | Turn on clock to connection block in NAP mode. |
| CONN_CLK_OFF | 0x5c | Turn off clock to connection block in NAP mode. |
| UPDATE_CONN | 0x68 | Update connection parameters. Deprecated. |
| RC_ANCHOR_POINT | 0x69 | Instruct LLH to check the validity of the set and recalculate the new connection anchor point parameter if valid. |
| ENTER_AUTO_DSM | 0x6A | Indicates CPU is idle. |
| KILL_CONN | 0x70 | Kill connection immediately. |
| KILL_CONN_AFTER_TX | 0x71 | Kill connection after a transmit operation is over. |
| RESET_US_COUNTER | 0xc3 | Reset microsecond counter |
| RESP_TIMER_ON | 0x72 | Start PDU response timer.<br><br>The PDU_RESP_TIMER register must be programmed with timeout value before issuing this command. |
| RESP_TIMER_OFF | 0x73 | Stop PDU response timer. |
| RESET_READ_PTR | 0x74 | Reset the white list memory read pointer to 0. |
| *CONN_PING_TIMER_ON | 0x75 | Start connection ping timer |
| *CONN_PING_TIMER_OFF | 0x76 | Stop connection ping timer |
| ENTER_DSM_SHUTDOWN | 0x77 | Enter deep sleep mode with shutdown |
| STORE_START | 0x78 | Firmware triggers the store process |
| RESTORE_START | 0x79 | Firmware triggers the restore process |
| DATA_RESTORE_ON | 0x7A | Firmware data restore enable |
| DATA_RESTORE_OFF | 0x7B | Firmware data restore disable |
| START_RPA_TIMER | 0x7C | Start Privacy RPA timer |
| STOP_RPA_TIMER | 0x7D | Stop Privacy RPA timer |
| START_RPA_GEN | 0x7E | Start RPA address generation |
| SOFT_RESET | 0x80 | Software reset. Resets all the hardware registers (except a few registers related to radio initialization). |
| LOAD_RAND_NUM | 0x8C | Load random seed to LFSR for RPA generation |

## 64.5.1.2 BLE changes

| |
|---|
| [EXAMPLE LABEL: d_ip_ble_mt_syn_spec.001] |

## 64.5.1.3 BTLE_RF register descriptions

### 64.5.1.3.1 BTLE_RF Memory map

BTLE_RF base address: 4103_2000h

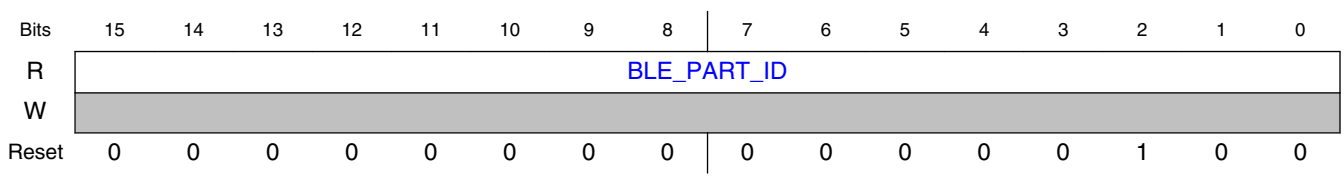
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 600h | BLUETOOTH LOW ENERGY PART ID (BLE_PART_ID) | 16 | RO | 0004h |
| 604h | BLE DSM STATUS (DSM_STATUS) | 16 | RO | See description. |
| 608h | BLE MISCELLANEOUS CONTROL (MISC_CTRL) | 16 | RW | 0000h |
| 60Ch | BLE STATE MACHINE STATUS (BLE_FSM) | 16 | RO | 0000h |

### 64.5.1.3.2 BLUETOOTH LOW ENERGY PART ID (BLE_PART_ID)

#### 64.5.1.3.2.1 Offset

| Register | Offset |
|---|---|
| BLE_PART_ID | 600h |

#### 64.5.1.3.2.2 Diagram




#### 64.5.1.3.2.3 Fields

| Field | Function |
|---|---|
| 15-0<br>BLE_PART_ID | BLE Part ID<br>0000000000000000b - Pre-production<br>0000000000000001b - Pre-production<br>0000000000000010b - KW40<br>0000000000000011b - KW41<br>0000000000000100b - K3S |

| Field | Function |
|---|---|
| | 0000000000000101b - KW35/KW36 |

## 64.5.1.3.3 BLE DSM STATUS (DSM_STATUS)

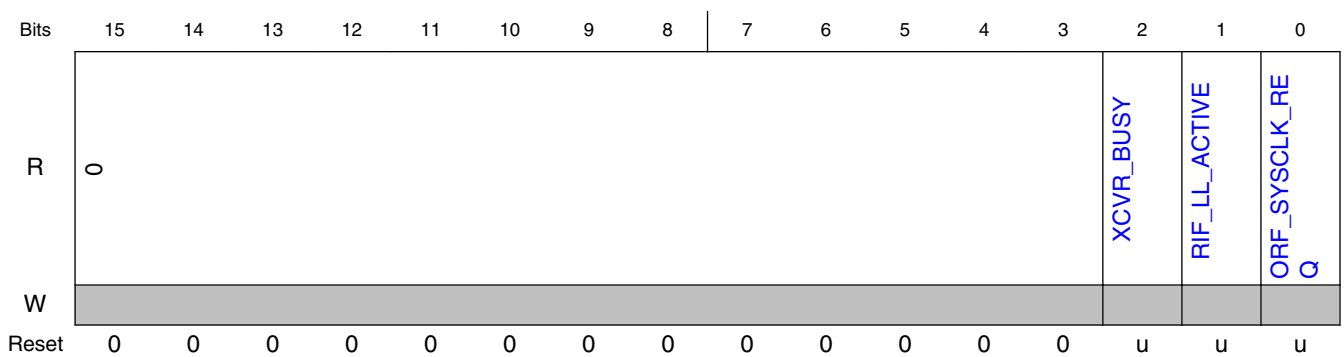
### 64.5.1.3.3.1 Offset

| Register | Offset |
|---|---|
| DSM_STATUS | 604h |

### 64.5.1.3.3.2 Function

BLE Deep Sleep Mode Status Register

### 64.5.1.3.3.3 Diagram




### 64.5.1.3.3.4 Fields

| Field | Function |
|---|---|
| 15-3<br>— | Reserved. |
| 2<br>XCVR_BUSY | Transceiver Busy Status Bit<br>0b - RF Channel in available (TSM is idle)<br>1b - RF Channel in use (TSM is busy) |
| 1<br>RIF_LL_ACTIVE | Link Layer Active<br>Reflects the state of the BLE LL output of the same name, the signal to be used by the host as an 'early' indication to prevent host to do any operations while the BLE block is doing transceiver operations, so as to reduce the peak power and noise. |
| 0 | RF Oscillator Requested |

| Field | Function |
|---|---|
| ORF_SYSCLK_REQ | Reflects the state of the BLE LL output of the same name, the control signal used to enable/disable the RF Oscillator for entry and exit from DSM (deep sleep mode). |

## 64.5.1.3.4 BLE MISCELLANEOUS CONTROL (MISC_CTRL)

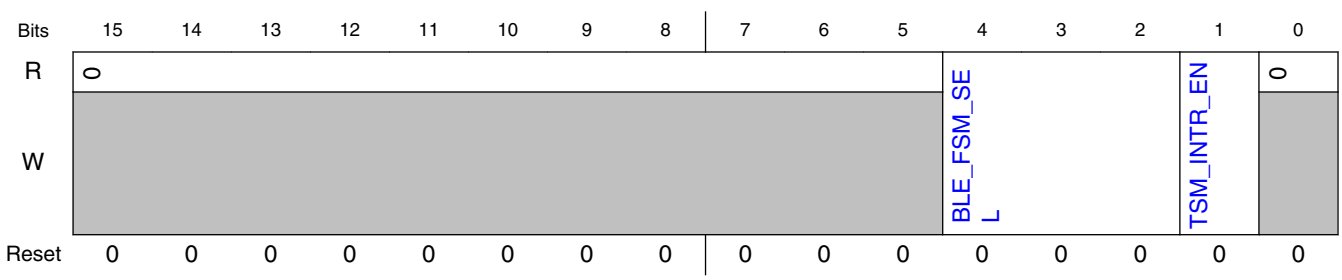
### 64.5.1.3.4.1 Offset

| Register | Offset |
|---|---|
| MISC_CTRL | 608h |

### 64.5.1.3.4.2 Function

BLE Miscellaneous Control Register

### 64.5.1.3.4.3 Diagram




### 64.5.1.3.4.4 Fields

| Field | Function |
|---|---|
| 15-5<br>— | Reserved. |
| 4-2<br>BLE_FSM_SEL | BLE FSM Selector<br>BLE_FSM[2:0] selects which BLE state machine appears in the VAR_CS field of the BLE_FSM register, according to the following table:<br>0: VAR_CS = {2'b00, dtm_cs[2:0]} (DTM Current State)<br>1: VAR_CS = { adv_cs[4:0]} (ADV Current State)<br>2: VAR_CS = { scan_cs[4:0]} (SCAN Curernt State)<br>3: VAR_CS = { init_cs[4:0]} (INIT Current State)<br>4: VAR_CS = {1'b0, connection_cs[3:0]} (CONN Current State, using Connection Engine 0)<br>5: Reserved<br>6: Reserved<br>7: Reserved |

*Table continues on the next page...*

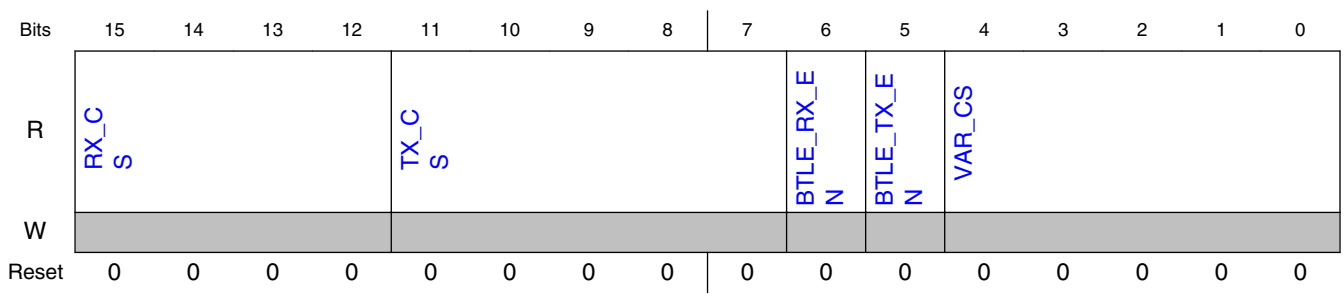
| Field | Function |
|---|---|
| 1<br>TSM_INTR_EN | TSM Interrupt Enable<br>This control bit enables the TSM Interrupt for BLE. If TSM_INTR_EN=1 and a TSM Interrupt occurs during a BLE TX or RX operation, bit [9] of the BLE EVENT_STATUS register will become set. |
| 0<br>— | Reserved. |

## 64.5.1.3.5 BLE STATE MACHINE STATUS (BLE_FSM)

### 64.5.1.3.5.1 Offset

| Register | Offset |
|---|---|
| BLE_FSM | 60Ch |

### 64.5.1.3.5.2 Diagram




### 64.5.1.3.5.3 Fields

| Field | Function |
|---|---|
| 15-12<br>RX_CS | BLE RX State Machine Current State<br>Current State of *rx_cs[3:0]* (BLE RX state machine) |
| 11-7<br>TX_CS | BLE TX State Machine Current State<br>Current State of *tx_cs[4:0]* (BLE TX state machine) |
| 6<br>BTLE_RX_EN | BLE RX Enable Control to TSM<br>Current State of *btle_rx_en* (BLE TSM RX request input) |
| 5<br>BTLE_TX_EN | BLE TX Enable Control to TSM<br>Current State of *btle_tx_en* (BLE TSM TX request input) |
| 4-0<br>VAR_CS | Variable State Machine Current State<br>One of several BLE state machines can be mapped to VAR_CS[4:0]. See description of BLE_FSM_SEL[2:0] in the MISC_CTRL register |

## 64.5.2 802.15.4 Link Layer Controller

### 64.5.2.1 Introduction

The 802.15.4 Link Layer Controller implements all hardware functionality of the 802.15.4 Link Layer.

### 64.5.2.2 Overview

The 802.15.4 Link Layer implementation is divided into software and hardware components. The 802.15.4 Link Layer Controller represents the hardware component of the link layer. Link Layer hardware functionality is further sub-divided into functional blocks. These blocks include:

- Packet Processor block.
- Sequence Manager block.
- Packet Storage block.
- Event Timer block.
- Interrupts block.
- Clear Channel Assessment/Energy Detect/Link Quality Indication block.
- Bit Streaming Mode block.

Together, these blocks implement and accelerate critical functions of the 802.15.4 Link Layer. The non-timing critical functions are implemented in software on a MCU.

### 64.5.2.3 Block Diagram

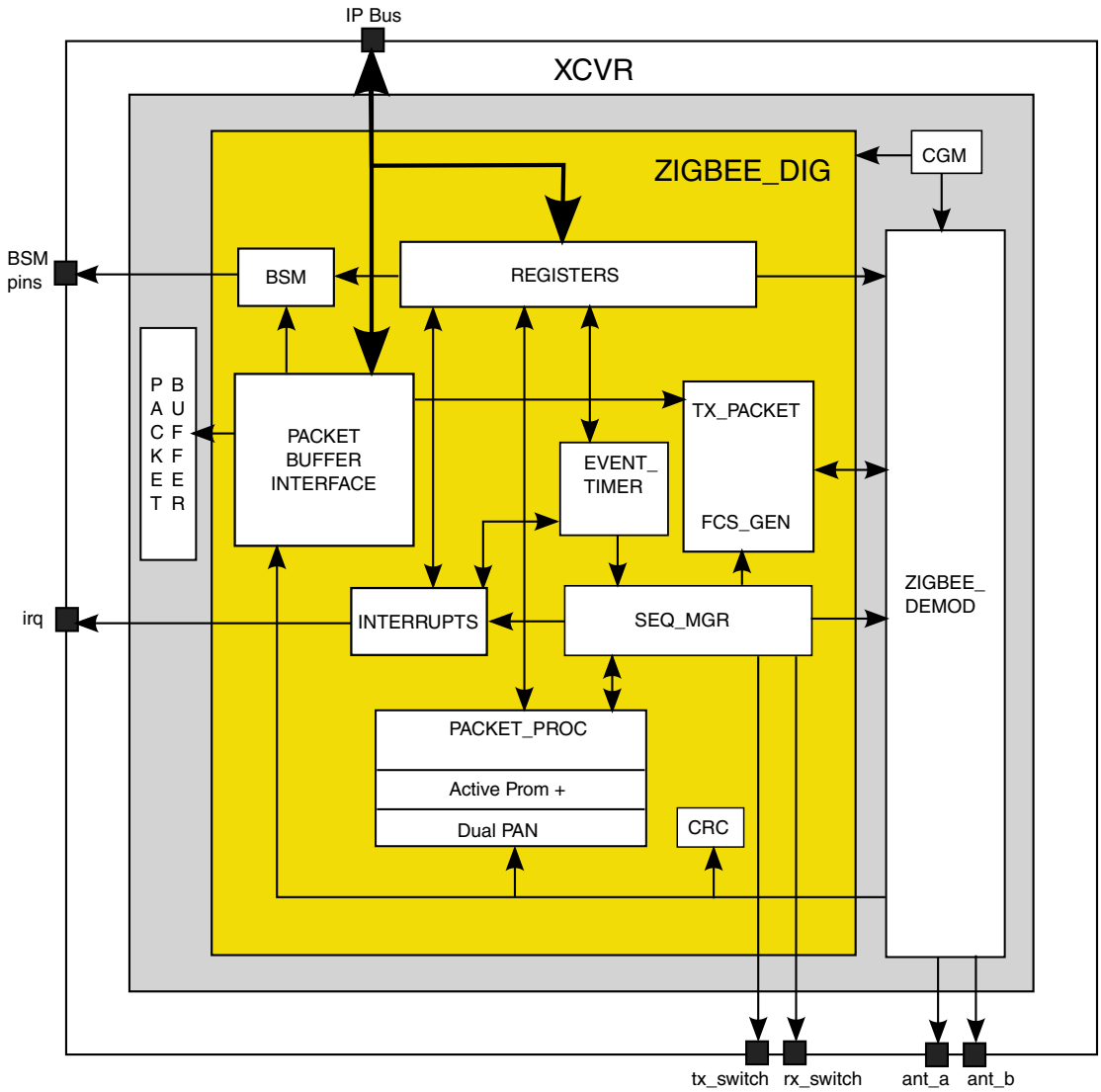



**Figure 64-138. 802.15.4 Link Layer Controller Block Diagram**

### 64.5.2.4 Functional Description

#### 64.5.2.4.1 Packet Processor

##### 64.5.2.4.1.1 Introduction

The 802.15.4 Packet Processor performs sophisticated hardware filtering of the incoming received packet, to determine whether the packet is both PHY- and MAC-compliant, whether the packet is addressed to this device, and if the device is a PAN Coordinator, whether a message is pending for the sending device. The packet processor greatly reduces the packet filtering burden on software, allowing software to tend to higher-layer tasks with a lower latency and smaller software footprint.

**Table 64-42. References**

| Revision | Title |
|---|---|
| IEEE Std 802.15.4 -2003 | Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) |
| IEEE Std 802.15.4 -2006 | 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) |
| IEEE Std 802.15.4 -2011 | Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) |
| IEEE Std 802.15.4e™-2012 | Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) (Amendment to IEEE Std 802.15.4™-2011) |
| IEEE Std 802.15.4 -2015 | Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) (Amendment to IEEE Std 802.15.4-2011) |

#### *64.5.2.4.1.1.1 Overview*

The 802.15.4 Packet Processor performs aggressive hardware filtering of the incoming packet as it is being received, to increase battery life by not awakening the MCU to process packets which are non-compliant or not addressed to the device, or that violate the 802.15.4 MAC Frame structure. The advanced frame parsing enables the sequence manager to handle complex sequences, by, for example, determining whether a receive frame requires an acknowledge, and generating the Sequence Number for that acknowledge, so that the transmit/receive combination sequence can be executed without any MCU intervention. Finally, the packet processor includes a hardware accelerator that allows the device to handle a critical timing requirement that had been problematic for 8-bit MCU's in the past: the quick-turnaround requirement for an indirect queue-lookup to determine the correct response to an end device sending a data polling request. The packet processor contains a register-based table of end devices which have messages pending, allowing a single-cycle hardware table lookup to be performed, and a minimum-turnaround-time acknowledge packet to be sent to the end device without MCU intervention. The packet processor also includes support for Dual PAN mode, which allows the device to simultaneously reside on 2 networks, with 2 separate IEEE addesses. And the packet processor also supports Active Promiscuous mode, which allows address filtering rules to be bypassed, while still enabling the device to automatically acknowledge packets which are addressed to the device.

#### 64.5.2.4.1.1.2 Features

- Aggressive packet filtering to enable long, uninterrupted MCU sleep periods
- Fully compliant with both 2003 and 2006 versions of the 802.15.4 wireless standard
- Supports all Frame Types, including reserved types
- Supports all valid 802.15.4 Frame Lengths
- Enables auto-Tx Acknowledge frames (no MCU intervention) by parsing of Frame Control Field and Sequence Number
- Supports all Source and Destination Address modes, and also PAN ID Compression
- Supports Broadcast address for PAN ID and short address mode
- Supports “Promiscuous” mode, to receive all packets regardless of address- and rules-checking
- Allows Frame Type-specific filtering (e.g., reject all but Beacon frames)
- Supports SLOTTED and non-SLOTTED modes
- Includes special filtering rules for PAN Coordinator devices
- Enables minimum-turnaround TX-Acknowledge frames for data-polling requests by automatically determining message-pending status
- Assists MCU in locating pending messages in its indirect queue for data-polling end devices
- Assists MCU in determining if a poll request originated from a valid child end device
- Makes available to MCU detailed status of frames which fail address- or rules-checking
- Supports Dual PAN mode, allowing the device to exist on 2 PAN’s simultaneously
- Supports 2 IEEE addresses for the device
- Supports Active Promiscuous Mode
- Includes provisions to enable software support of 802.15.4e and 802.15.4-2015 frame types and versions

#### 64.5.2.4.1.1.3 Block diagram

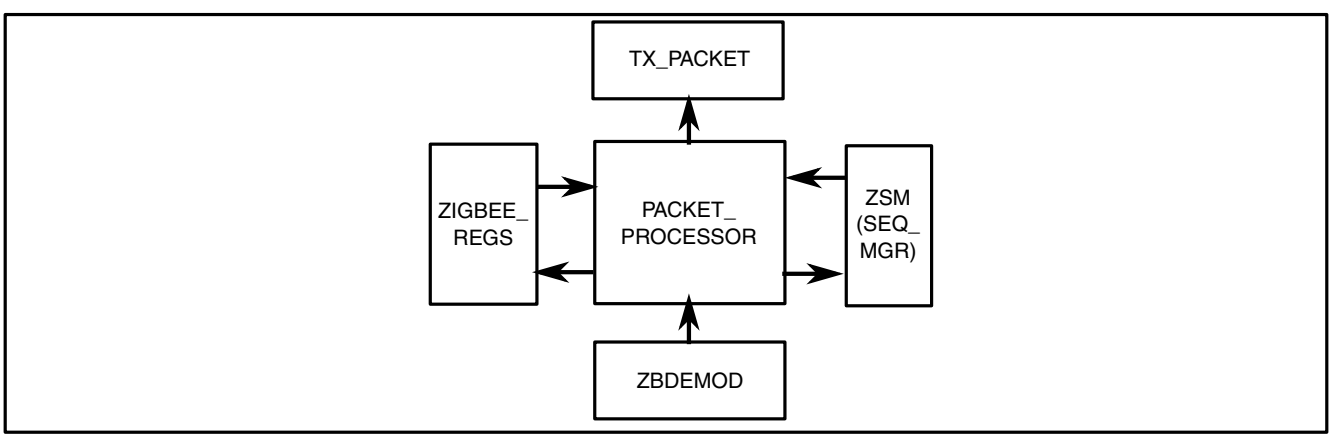



**Figure 64-139. Block diagram**

#### 64.5.2.4.1.2 Memory Map and register definition

Numerous registers exist to provide configuration and status for the 802.15.4 Packet Processor.

The memory-mapped registers which affect the Packet Processor are described in the table below.

| Field | R/W | Description |
|---|---|---|
| PI | r | Poll Indication:<br>1: the received packet was a data request, regardless of whether a source-address table-match occured, and regardless of whether source-adress matching is enabled (see SRCADDR_EN bit);<br>0: the received packet was not a data request<br>PI is a read-only bit. |
| SRCADDR | r | If Source Address Management is engaged, meaning at least one of the following bits is set:<br>• SAP0_EN<br>• SAA0_EN<br>• SAP1_EN<br>• SAA1_EN<br>then SRCADDR will be set to 1 if the packet just received is a poll request (PI=1), and at least one of the following conditions is met:<br>• SAP0_EN && SAP0_ADDR_PRESENT<br>• SAA0_EN && SAA0_ADDR_ABSENT<br>• SAP1_EN && SAP1_ADDR_PRESENT<br>• SAA1_EN && SAA1_ADDR_ABSENT<br>If SRCADDR=1, this indicates to SW that the Packet Processor has determined that an auto-TxACK frame must be transmitted with the FramePending subfield of the FrameControlField set to 1. HW will assemble and transmit this Ack packet.<br>If the above conditions are not met, SRCADDR will be cleared to 0. |
| SAP0_EN | rw | 1: Enables SAP0 Partition<br>0: Disables SAP0 Partition |
| SAA0_EN | rw | 1: Enables SAA0 Partition<br>0: Disables SAA0 Partition |
| SAP1_EN | rw | 1: Enables SAP1 Partition<br>0: Disables SAP1 Partition |
| SAA1_EN | rw | 1: Enables SAA1 Partition<br>0: Disables SAA1 Partition |
| SAA0_START[6:0] | rw | First Index of SAA0 partition. |
| SAP1_START[6:0] | rw | First Index of SAP1 partition. |
| SAA1_START[6:0] | rw | First Index of SAA1 partition. |
| SAM_CHECKSUM[15:0] | | Software-computed source address checksum, to be installed into a table index |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| SAM_INDEX[6:0] | | Contains the table index to be enabled or invalidated. Software must ensure that the index is within the range of the desired partition. |
| SAM_INDEX_WR | w | For 32-bit writes, SAM_INDEX_WR must be set to indicate that the table entry specified by SAM_INDEX[6:0] is to be written; if SAM_INDEX_WR=0, the table entry is not written, but the SAM_INDEX[6:0] register is updated. For 8-bit writes, this bit is ignored. |
| SAM_INDEX_EN | w | Enable the index selected by SAM_INDEX[6:0] for searching. |
| SAM_INDEX_INV | w | Invalidate the index selected by SAM_INDEX[6:0] |
| FIND_FREE_IDX | w | After modifying Valid bits (enabling or invalidating), write this bit to 1 to force HW to update the “First Free Index” registers to account for the changed Valid bits. This HW update process takes 4us. SW can poll SAM_BUSY to determine when the table update is complete. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| SAM_BUSY | r | HW is in the process of updating the Source Address table, either in response to a poll indication from the packet processor, or due to SW setting FIND_FREE_IDX=1. In the latter case, SW should poll SAM_BUSY until low before accessing the “First Free Index” registers. Read-only bit. |
| INVALIDATE_ALL | w | Writing a 1 to this bit clears all 128 Valid bits. Invalidates the entire table. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| SAP0_1ST_FREE_IDX[6:0] | r | First non-enabled (invalid) index in the SAP0 partition. Read-only. |
| SAA0_1ST_FREE_IDX[6:0] | r | First non-enabled (invalid) index in the SAA0 partition. Read-only. |
| SAP1_1ST_FREE_IDX[6:0] | r | First non-enabled (invalid) index in the SAP1 partition. Read-only. |
| SAA1_1ST_FREE_IDX[6:0] | r | First non-enabled (invalid) index in the SAA1 partition. Read-only |
| PROMISCUOUS | rw | Bypasses most packet filtering.<br>1: all packet filtering except frame length checking (FrameLength>=5 and FrameLength<=127) is bypassed.<br>0: normal mode<br>**Note:** Promiscuous Mode is not intended for Sequence TR with RXACKRQD=1 (auto-RXACK mode), because the contents of auto-RXACK packets are not stored in the Packet Buffer. If auto-RXACK packets are to be stored in the PB, use ACTIVE_PROMISCUOUS instead. |
| PANCORDNTR0 | rw | Device is a PAN Coordinator on PAN0. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| MACPANID0[15:0] | rw | MAC PAN ID for PAN0. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID against this register. Also, if PANCORDNTR0=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| MACSHORTADDRS0[15:0] | rw | MAC Short Address for PAN0, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| MACLONGADDRS0[63:0] | rw | MAC Long Address for PAN0, for 64-bit destination addression mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| BEACON_FT | rw | 1: Beacon frame type enabled.<br>0: reject all Beacon frames |
| ACK_FT | rw | 1: Acknowledge frame type enabled.<br>0: reject all Acknowledge frames |
| DATA_FT | rw | 1: Data frame type enabled.<br>0: reject all Data frames |
| CMD_FT | rw | 1: MAC Command frame type enabled.<br>0: reject all MAC Command frames |
| LLDN_FT | rw | 1: LLDN frame type enabled (Frame Type 4).<br>0: reject all MAC Command frames |
| MULTIPURPOSE_FT | rw | 1: MULTIPURPOSE frame type enabled (Frame Type 5).<br>0: reject all MAC Command frames |
| EXTENDED_FT | rw | 1: EXTENDED frame type enabled (Frame Type 7).<br>0: reject all MAC Command frames |
| NS_FT | rw | 1: Not-specified (reserved) frame type enabled. No packet filtering is performed, except for frame length checking (FrameLength>=5 and FrameLength<=127).<br>0: reject all reserved frame types |
| EXTENDED_FCS_CHK | rw | Verify FCS on Frame Type Extended<br>1: Packet Processor will check FCS at end-of-packet based on packet length derived from PHR, for Frame Type EXTENDED<br>0: Packet Processor will not check FCS for Frame Type EXTENDED (default) |
| ACTIVE_PROMISCUOUS | rw | 1: Provide Data Indication on all received packets under the same rules which apply in PROMISCUOUS mode, however acknowledge those packets under rules which apply in non-PROMISCUOUS mode<br>0: normal operation (Default) |
| FRM_VER_FILTER[3:0] | rw | The incoming packet's Frame Control Field is parsed to obtain the FrameVersion subfield, and that value is compared against this register, in accordance with the following:<br>xxx1: Accept received packets with FrameVersion=00<br>xx1x: Accept received packets with FrameVersion=01<br>x1xx: Accept received packets with FrameVersion=10<br>1xxx: Accept received packets with FrameVersion=11<br>These filtering rules apply to Beacon, Acknowledge, Data, and MAC Command Frame Types, since these frame types require a 2-octet Frame Control Field which embeds a 2-bit FrameVersion subfield. |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | Later frame types introduced in the 802.15.4e addendum (LLDN, Multipurpose) don't guarantee a FrameVersion subfield with the original meaning, so these filtering rules do not apply to these frame types. See registers LLDN_FT and MULTIPURPOSE_FT.<br>For Acknowledge frames, FrameVersion bits are ignored by the Packet Processor, irrespective of FRM_VER_FILTER. |
| MACPANID1[15:0] | rw | MAC PAN ID for PAN1. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID against this register. Also, if PANCORDNTR1=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |
| MACSHORTADDRS1[15:0] | rw | MAC Short Address for PAN1, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| MACLONGADDRS1[63:0] | rw | MAC Long Address for PAN1, for 64-bit destination addression mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| ACTIVE_NETWORK | rw | Selects the PAN on which to transceive, by activating a PAN parameter set (PAN0 or PAN1). In Manual Dual PAN mode (or Single PAN mode), this bit selects the active PAN parameter set (channel and addressing parameters) which governs all autosequences. In Auto Dual PAN mode, this bit selects the PAN on which to begin transceiving, latched at the point at which DUAL_PAN_DWELL register is written.<br>1: Select PAN1<br>0: Select PAN0 (Default) |
| DUAL_PAN_AUTO | rw | Activates automatic Dual PAN operating mode. In this mode, PAN-switching is controlled by hardware at a pre-programmed rate, determined by DUAL_PAN_DWELL.<br>1: Auto Dual PAN Mode<br>0: Manual Dual PAN mode (or Single PAN mode). Default.<br>Whenever DUAL_PAN_AUTO=0, CURRENT_NETWORK=ACTIVE_NETWORK at all times. In other words, software directly controls which PAN is selected.<br>Whenever DUAL_PAN_AUTO=1, CURRENT_NETWORK is controlled by hardware |
| PANCORDNTR1 | rw | Device is a PAN Coordinator on PAN1. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| CURRENT_NETWORK | r | This read-only bit indicates which PAN is currently selected by hardware in automatic Dual PAN mode<br>1: PAN1 is selected<br>0: PAN0 is selected |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>R/W</th><th>Description</th></tr>
<tr><td>DUAL_PAN_DWELL[7:0]</td><td>rw</td><td>Channel Frequency Dwell Time. In Auto Dual PAN mode, hardware will toggle the PAN, after dwelling on the current PAN for the interval described below (assuming Preamble/SFD not detected). A write to DUAL_PAN_DWELL, always re-initilizes the DWELL TIMER to the programmed value. If a write to DUAL_PAN_DWELL occurs during an autosequence, the DWELL TIMER will begin counting down immediately. If a write to DUAL_PAN_DWELL occurs when there is no autosequence underway, the DWELL TIMER will not begin counting until the next autosequence begins; it will begin counting at the start of the sequence warmup.<br><br>
<table>
<tr><th>PRESCALER (bits [1:0])</th><th>TIMEBASE (bits [7:2])</th><th>RANGE (min) - (max)</th></tr>
<tr><td>00</td><td>0.5ms</td><td>0.5 - 32ms</td></tr>
<tr><td>01</td><td>2.5ms</td><td>2.5 - 160ms</td></tr>
<tr><td>10</td><td>10ms</td><td>10 - 640ms</td></tr>
<tr><td>11</td><td>50ms</td><td>50ms - 3.2seconds</td></tr>
</table>
<br>A write to DUAL_PAN_DWELL also causes the value of ACTIVE_NETWORK to get latched into the hardware. This latched value will be the starting point for the automatic dual-pan mode (i.e., start on PAN0 or on PAN1). The starting value takes effect immediately (if sequence is underway and DUAL_PAN_AUTO=1), or is otherwise delayed until sequence starts and DUAL_PAN_AUTO=1.</td></tr>
<tr><td>DUAL_PAN_REMAIN[5:0]</td><td></td><td>This read-only register indicates time remaining before next PAN switch in auto Dual PAN mode. The units for this register, depend on the PRESCALER setting (bits [1:0]) in the DUAL_PAN_DWELL register, according to the following table:<br><br>
<table>
<tr><th>DUAL_PAN_DWELL PRESCALER</th><th>DUAL_PAN_REMAIN UNITS</th></tr>
<tr><td>00</td><td>0.5ms</td></tr>
<tr><td>01</td><td>2.5ms</td></tr>
<tr><td>10</td><td>10ms</td></tr>
<tr><td>11</td><td>50ms</td></tr>
</table>
<br>The readback value indicates that between N-1 and N timebase units remain until the next PAN switch. For example, a DUAL_PAN_REMAIN readback value of 3, with a DUAL_PAN_DWELL PRESCALER setting of 2 (10ms), indicates that between 20ms (2*10ms) and 30ms (3*10ms), remain until the next automatic PAN switch.</td></tr>
<tr><td>RECD_ON_PAN0</td><td>r</td><td>Indicates the packet which was just received, was received on PAN0.<br><br>In Dual PAN mode operating on 2 different channels, RECD_ON_PAN0 will be set if CURRENT_NETWORK=0 when the packet was received, regardless of FILTERFAIL status.</td></tr>
</table>

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN0 will be set only if a valid packet was received on PAN0 (PAN0's FILTERFAIL_FLAG is deasserted). |
| RECD_ON_PAN1 | r | Indicates the packet which was just received, was received on PAN1.<br>In Dual PAN mode operating on 2 different channels, RECD_ON_PAN1 will be set if CURRENT_NETWORK=1 when the packet was received, regardless of FILTERFAIL status.<br>In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN1 will be set only if a valid packet was received on PAN1 (PAN1's FILTERFAIL_FLAG is deasserted). |
| FILTERFAIL_CODE[9:0] | r | Status of packet processor filtering. For incoming packets which have been rejected by the packet processor, FILTERFAIL_CODE will indicate which violated packet-filtering rule(s) led to the rejection. This is intended for debug only. The contents of this register are valid at the FILTERFAIL_IRQ interrupt. The packet processor maintains 2 independent copies of FILTERFAIL_CODE[9:0], one for each PAN. When reading this register, software can select for which PAN the FILTERFAIL_CODE applies, using the FILTERFAIL_PAN_SEL bit. See Appendix A. |
| FILTERFAIL_PAN_SEL | rw | 1: FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN1<br>0: FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN0 |
| LENIENCY[39:0] | rw | The LENIENCY register allows individual packet filtering rules to be overriden, on a rule-by-rule basis. In other words, a specific packet filtering rule, such as an address-mode check, which would result in an incoming packet being rejected, can be overridden by setting the corresponding bit in the LENIENCY register, which would then result in the packet being accepted. This is similar to PROMISCUOUS mode, except that it allows checks to be overridden individually. This will facilitate debug, and potentially allow the device to communicate with devices which may not be 100% MAC-compliant. The mapping of LENIENCY register bits to rules is shown in Appendix B of this document. |

#### 64.5.2.4.1.3 Functional Description

The packet processor is responsible for receiving symbols from the demodulator, combining them into octets, assembling the octets into packets, filtering the packets in compliance with the 802.15.4 wireless MAC standard, and storing the packets into the Packet Buffer (Packet RAM).

The packet processor parses the incoming packets on-the-fly to determine:

1. whether the packet is MAC-compliant, and if it is,
2. whether the packet is addressed to the end device.

The packet processor performs aggressive filtering on incoming packets, in order to reduce unnecessary MCU wake-ups, to allow longer MCU sleep durations, and to provide sufficient hardware automation so as to enable complex sequences to be executed by the sequence manager. The packet processor also includes hardware support for Dual PAN operation (allowing the device to reside on two networks simultaneously), and for Active Promiscuous mode (allowing the device to receive all packets, but acknowledging only those addressed to the device).

The 802.15.4 packet processor also includes functionality to allow a critical-timing requirement to be met, which in the past has been problematic for low-MIPS 802.15.4 receiving devices: the ability of a coordinator device to respond to a MAC Command data request from an end device, with an Acknowledge frame containing a FramePending bit accurately reflecting the presence of a message for the end device in the coordinator's indirect queue, or the absence of the end device in the coordinator's neighbor table, within the prescribed 192us window. This functionality is described in more detail in the Source Address Matching section.

#### *64.5.2.4.1.3.1 Packet Filtering Background*

The packet processor parses packets to verify compliance with the 802.15.4 MAC frame format. The basic MAC frame format is shown in the figure below.

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/ 14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

**Figure 64-140. MAC Frame Format**

The packet processor parses the incoming octets on the fly, as they are received. The Frame Control Field, two octets in length, contains subfields which encode "instructions" on how to parse the remainder of the MHR (MAC Header). The structure of the Frame Control Field is shown in the figure below.

| Bits: 0–2 | 3 | 4 | 5 | 6 | 7–9 | 10–11 | 12–13 | 14–15 |
|---|---|---|---|---|---|---|---|---|
| Frame Type | Security Enabled | Frame Pending | Ack. Request | PAN ID Compression | Reserved | Dest. Addressing Mode | Frame Version | Source Addressing Mode |

**Figure 64-141. Frame Control Field**

The packet processor utilizes the Frame Control Fields subfields in the following manner:

| FCF Subfield | Utilization by Packet Processor |
|---|---|
| FrameType | Interprets the remaining MHR as specific to Beacon, Ack, Data, Command, or Reserved frame types. Each frame type has a unique MHR structure, and so different parsing rules apply. |
| SecurityEnabled | If SecurityEnabled=1 and FrameVersion>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR and will need to be further parsed by the packet processor if this is a MAC Command frame (see Section Source Address Management below). There is no other use of SecurityEnabled by the packet processor. |
| FramePending | ignored by packet processor |
| Ack. Request | will be copied into the **pp_ack_request** signal to the sequence manager. An auto-TxAck frame will follow the incoming receive frame if necessary conditions are met |
| PAN ID Compression | used for addressing mode rules-checking and addressing field parsing. |
| Reserved | ignored by packet processor |
| Destination Addressing Mode | used for addressing mode rules-checking and addressing field parsing |
| FrameVersion | For Beacon, Data, and MAC Command frames, FrameVersion is checked against the allowed frame versions according to FRM_VER_FILTER<br>For Acknowledge frames, FrameVersion is ignored.<br>If SecurityEnabled=1 and FrameVersion>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR and the MHR will need to be further parsed by the packet processor if this is a MAC Command frame (see Section Source Address Management). |
| Source Addressing Mode | used for addressing mode rules-checking and addressing field parsing |

The Packet Processor parses the FrameType field of the Frame Control Field per the following table:

| Frame type value $b_2$ $b_1$ $b_0$ | Description |
|---|---|
| 000 | Beacon |
| 001 | Data |
| 010 | Acknowledgment |
| 011 | MAC command |
| 100–111 | Reserved |

**Figure 64-142. Frame Type Field**

Directly following the Frame Control Field is the Sequence Number field. The Sequence Number field of the MHR is captured by the packet processor. If an auto-TxAck frame follows the incoming receive frame, the captured Sequence Number will be copied to the transmitted Acknowledge packet, and the Packet Processor will insert FrameVersion=00 into the Frame Control Field of the transmitted frame, regardless of the received FrameVersion subfield of the original frame.

The Addressing Fields follow the Sequence Number. The format of the Addressing Fields depends upon the Source and Destination Addressing Mode subfields of the Frame Control Field. The Addressing Modes are defined in the table below:

| Addressing mode value $b_1$ $b_0$ | Description |
|---|---|
| 00 | PAN identifier and address fields are not present. |
| 01 | Reserved. |
| 10 | Address field contains a 16-bit short address. |
| 11 | Address field contains a 64-bit extended address. |

**Figure 64-143. Addressing Mode Fields**

The packet processor uses these Source and Destination Address Modes shown above, to extract the Source PAN ID and Address (if present), and the Destination PAN ID and Address (if present), from the MHR, according to the table below:

| FrameType | Destination Addressing Mode | Source Addressing Mode | PanID-Compression | Addressing Fields |
|---|---|---|---|---|
| Acknowledge only<br>(reject all other frame types) | 0 | 0 | 0 | None |
| Data or Command<br>(reject Ack and Beacon frames) | 2 or 3 | 0 | 0 | DstPanID + DstAddr |
| Beacon (all devices), or<br>Data or Command (PAN Coord Only)<br>(reject Ack frames) | 0 | 2 or 3 | 0 | SrcPanID + SrcAddr |
| Data or Command<br>(reject Ack and Beacon frames) | 2 or 3 | 2 or 3 | 0 | DstPanID + DstAddr + SrcPanId + SrcAddr |
| Reject all frames<br>(Illegal as per Section 7.2.2.1.5) | 0 | 0 | 1 | - |
| Reject all frames<br>(Illegal as per Section 7.2.2.1.5) | 2 or 3 | 0 | 1 | - |
| Reject all frames<br>(Illegal as per Section 7.2.2.1.5) | 0 | 2 or 3 | 1 | - |
| Data or Command<br>(reject Ack or Beacon frames) | 2 or 3 | 2 or 3 | 1 | DstPanID + DstAddr + SrcAddr |
| Reject all frames<br>(Illegal as per Section 7.2.2.1.6) | 1 | - | - | - |
| Reject all frames<br>(Illegal as per Section 7.2.2.1.8) | - | 1 | - | - |

The Source PAN ID and Address, and Destination PAN ID and Address, extracted by the packet processor, are then checked against the MACPANID, MACSHORTADDRS, and MACLONGADDRS register settings, depending on the FrameType of the incoming packet, to determine:

1. Is the addressing mode combination valid for this frame type?
2. Do the address fields indicate that the packet is indeed addressed to the end device?

The details on how the packet processor answers those 2 questions, are described in the following sections.

#### *64.5.2.4.1.3.2 Packet Filtering Detail*

The figure below is a high-level representation of the packet filtering performed by the 802.15.4 packet processor. The sections which follow provide more detail on the packet filter implementation.

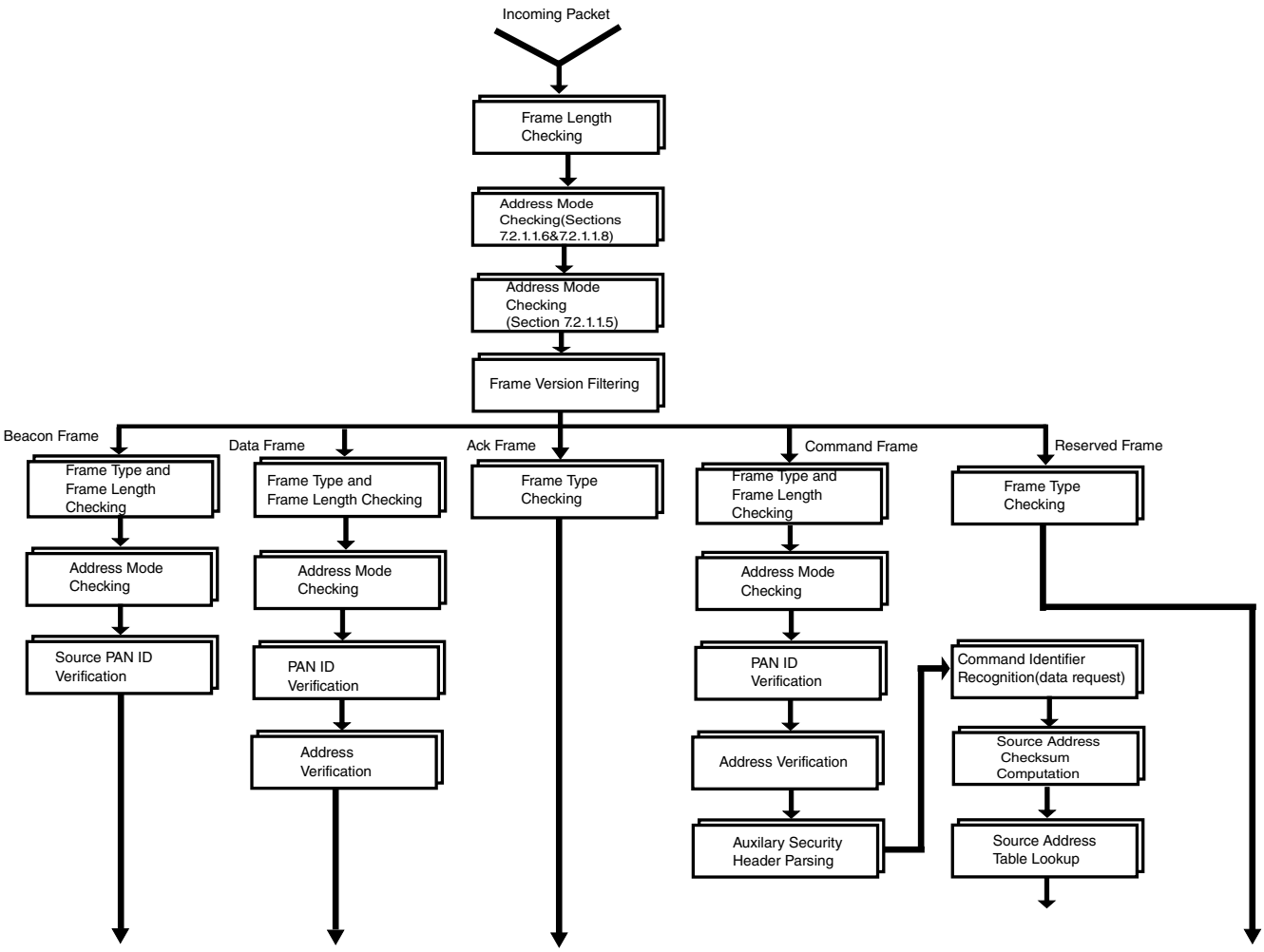



**Figure 64-144. Packet Filtering Flow Diagram**

## First-Stage Packet Filtering

The first stage of packet filtering examines the incoming packet, and implements basic rule-checking, based on the 2006 version of the 802.15.4 standard, and is applied to *all* frame types. First stage rule-checking is described in the table below:

| | Filtering Step | Description |
|---|---|---|
| 1 | Frame Length Checking | Check that all frames are at least 5 octets long (minimum frame length required by the standard), and no longer than 127 octets long (maximum frame length allowed by the standard). |
| 2 | Address Mode Checking (Sections 7.2.1.1.6 and 7.2.1.1.8) | Check that neither Source nor Destination Address Mode set to 1 (illegal addressing mode) |
| 3 | Address Mode Checking (Section 7.2.1.1.5) | Check for PAN ID Compression enabled, with either Source or Destination Address Mode set to 0 (illegal addressing mode) |
| 4 | FrameVersion Checking | Check that incoming packet's Frame Version is allowed based on FRM_VER_FILTER setting. (FRM_VER_FILTER is a programmable register, see Register Descriptions Section). |

Packets which pass first stage filtering, proceed to the second stage of packet filtering, which is *frame type-specific* filtering. Packets which fail initial rule checking are marked as rejected.

## Second-Stage Packet Filtering

The second stage of packet filtering is *frame type-specific*. Different variations of the MAC Header (MHR) apply to different frame types, so different parsing mechanisms apply to each.

**Note:** In the following section, the following references are to register bits; see Register Description Section for register descriptions: BEACON_FT, ACK_FT, DATA_FT, CMD_FT, NS_FT, PANCORDNTR, MACPANID, MACSHORTADDRS, and MACLONGADDRS.

1) For Beacon frames, these are the basic filtering steps:

- Check that Beacon frames are enabled by software (BEACON_FT)
- Check that FrameLength >= 9
- Check for illegal combinations of PAN ID Compression and Source/Destination Address Modes
- Verify the Source PAN ID matches MACPANID, or that MACPANID = 0xffff.

2) For Acknowledge frames, these are the basic filtering steps:

- Check that Ack frames are enabled by software (ACK_FT)
- If Sequence TR is active and RXACKRQD = 1, check the received Sequence Number matches the transmitted Sequence Number; FrameVersion bits are ignored, as per section 7.2.2.3.1 of the 802.15.4 standard.

**Note**: It is recommended to use Sequence TR to receive Acknowledge frames, not Sequence R. See Chapter ZSM Sequence Manager, Section Sequence TR (Transmit/ Receive).

3) For Data frames, these are the basic filtering steps:

- Check that Data frames are enabled by software (DATA_FT)
- Check that FrameLength >= 9
- Check for illegal combinations of Source and Destination Address Modes
- If PANCORDNTR=0, check for missing Destination Address field or present Source Address field
- If no Destination Address field (PANCORDNTR=1 only), check Source PAN ID matches MACPANID
- If Destination Address present, check Destination PAN ID matches MACPANID or broadcast (0xffff)
- If Short Destination Address field, check address matches MACSHORTADDRS or broadcast (0xffff)
- If Long Destination Address field, check address matches MACLONGADDRS.

4) For MAC Command frames, these are the basic filtering steps:

- Check that Command frames are enabled by software (CMD_FT)
- Check that FrameLength >= 9
- Check for illegal combinations of Source and Destination Address Modes
- If PANCORDNTR=0, check for missing Destination Address field or present Source Address field
- If no Destination Address field (PANCORDNTR=1 only), check Source PAN ID matches MACPANID
- If no Destination Address field (PANCORDNTR=1 only), capture Source PAN ID and Source Address for later use
- If Destination Address present, check Destination PAN ID matches MACPANID or broadcast (0xffff)
- If Short Destination Address field, check address matches MACSHORTADDRS or broadcast (0xffff)
- If Long Destination Address field, check address matches MACLONGADDRS.
- At end of the addressing field, if incoming packet's FrameVersion=0 or SecurityEnabled=0, capture the next octet (Command Identifier)
- At end of the addressing field, if incoming packet's FrameVersion!=0 and SecurityEnabled=1, parse the Auxiliary Security Header, then capture the next octet (Command Identifier)
- If Command Identifier indicates the MAC Command is a data request, compute a "Source Address Checksum" to uniquely identify the sending device.

#### *64.5.2.4.1.3.3 Source Address Management*

The 802.15.4 wireless MAC standard envisions a scenario whereby an end device may interrogate a coordinator as to whether the coordinator is storing data (i.e., a pending message) for the end device. The situation arises in a beacon-enabled network, when a coordinator includes in its transmitted beacon frame the MAC address of the end device in its “Pending Address Fields”. The “Pending Address Fields” are part of the required MAC payload of the beacon frame. The “Pending Address Fields” contain a list of end device addresses for which messages are pending. In this scenario, an end device which finds its address included in the “Pending Address Fields” of the received beacon frame, must respond to the beacon (coordinator) with a MAC Command of type “data request”. Alternatively, in non-beacon-enabled networks, an end device may periodically wake up and “poll” a coordinator, to determine if a message is pending for the end device. In either case, the coordinator stores messages for its end devices in its “indirect queue”. The coordinator must respond to an incoming MAC Command data request (which *must* have AckRequest=1 in its Frame Control Field), with an Acknowledge frame containing a FramePending subfield indicating the presence (or absence) of a message for the requesting end device, in the coordinator’s indirect queue. For a coordinator built around a low-MIPS MCU, a significant amount of MCU time is typically required, simply to ascertain whether a given end device has a message pending or not; the indirect queue can be quite large, and for 8-bit MCU’s, the queue must be searched byte-by-byte. This makes it difficult, or impossible, for an underpowered MCU to establish frame-pending status within the 192us RX-to-TX turnaround time required by the standard.

Also, coordinators may include, in software, a “Neighbor Table”, the contents of which consist of all the end devices with which the coordinator is in communication. The neighbor table requires periodic maintenance by coordinator (parent) software, as end devices (children) come and go within the network, join different parents, or simply never return. Child tables have limited capacity and parents need to create space for new devices that may show up. A timeout mechanism is the only way that can guarantee that child tables will eventually be cleaned up. Coordinator software must “age out” end devices with which there has not been recent communication. In order for an end device to refresh its status within its parent’s Neighbor Table, the end device may periodically send a “keepalive” message in the form of a MAC Data Poll Request. Upon receipt of such a poll request, the parent must search its neighbor table to determine if the source address of the end device is a valid table entry. If the end device exists in the table, an Acknowledgment packet must be transmitted to the end device, with the FramePending subfield set to 0, and coordinator software resets that end device’s timeout counter; if the child is not present in the table, an Acknowledgment packet must be transmitted to the end device, with the FramePending subfield set to 1, and the parent device will follow up by transmitting a “Leave Packet” to the end device, instructing the end device to sever its child status with the parent. The end device may opt to rejoin at a later time. Due to the

size of the parent's Neighbor table, the search a child's source address can be time consuming, and may not always be possible within the 192us RX-to-TX turnaround time required by the standard.

To alleviate both of these problems, the 802.15.4 packet processor includes hardware acceleration to expedite the source address table search, greatly relieving the software bottleneck during the critical 192us turnaround time. The table search acceleration consists of 2 components: 1) a search for the *presence* of a specific end device's source address, which expedites the process of determining whether a message is pending for the device; and 2) a search for the *absence* of a specific end device's source address, which expedites the process of determining whether the end device is a valid child of the parent.

The packet processor maintains a register table (128 deep, 16-bit wide). The table is divided into multiple partitions, one to perform the "Source Address Present" (SAP) search, and another to perform the "Source Address Absent" (SAA) search. In the SAP partition of the table, coordinator software will store the address of end devices for which it has a message pending; in the SAA partition of the table, coordinator software will store the address of end devices for which it is the parent. To reduce area, the address information is stored in a "compressed" format. Software compresses the address information, and stores it into the table partitions, during a non-critical time. Then, when the coordinator's packet processor receives a MAC Command data request, it extracts the source address information from the incoming packet, and compresses it using the *same* compression algorithm used by software to populate the source address table. The packet processor performs 2 steps simultaneously:

1. it compares the just-received, compressed source address information, or "checksum", against all of the enabled entries in the SAP partition of the table. If a match is found in the SAP partition, the packet processor *asserts* the **rx_frame_pending** signal to the transmit block (TX_PACKET), so that an auto-TxAck packet can be sent to the requesting end device, with the FramePending subfield set to 1, indicating that the end device has a message pending in the coordinator's indirect queue. A message packet from the coordinator to the end device will follow.
2. it compares the just-received, compressed source address information, or "checksum", against all of the enabled entries in the SAA partition of the table. If *no* match is found in the SAA partition, the packet processor *asserts* the **rx_frame_pending** signal to the transmit block (TX_PACKET), so that an auto-TxAck packet can be sent to the requesting end device, with the FramePending subfield set to 1, indicating that the end device is not a valid child of this parent, and a "Leave Packet" from the coordinator to the end device, will follow.

If neither condition is met, i.e, there is no address match in the SAP partition, and a positive match in the SAA partition, the packet processor *deasserts* the **rx_frame_pending** signal, so that an auto-TxAck packet can be sent indicating to the end device that it has no data pending, and that it remains a valid child of this parent; the end device may then deactivate its receiver immediately. For the packet processor, the entire table lookup occurs in 128 clock cycles, once the last source address octet has been received. The SAP and SAA lookups are performed simultaneously. The checksum is also computed on-the-fly by hardware as the source address octets are received, taking no more than 1 clock cycle per octet. Finally, table-search status is reported to the MCU:

1. after the packet processor completes its SAP table search, the status bit SAP0_ADDR_PRESENT will be set to 1 if a match is found in the SAP partition, and the table index which matched will be indicated in the SAP0_MATCH[6:0] register, to further assist software in locating the relevant message within its indirect queue; if no match is found in the SAP partition, the status bit SAP0_ADDR_PRESENT will be set to 0.
2. after the packet processor completes its SAA table search, the status bit SAA0_ADDR_ABSENT will be set to 1 if *no* match is found in the SAA partition; if a positive match is found in the SAA partition, the status bit SAA0_ADDR_ABSENT will be set to 0, and the table index which matched will be indicated in the SAA0_MATCH[6:0] register, to further assist software in locating the child device within its neighbor table.

Both features, SAP and SAA acceleration, are optional, and functionally independent. Coordinator software can elect to enable both, either, or neither feature. The control bit SAP0_EN enables SAP acceleration. The control bit SAA0_EN enables SAA acceleration. If either feature is disabled, that table partition (SAP or SAA) is not searched, and so is not a factor in determining FramePending status for the transmitted Acknowledge frame. If *both* features are disabled, FramePending status defaults to software control, and merely tracks the state of the ACK_FRM_PND register bit; in this case software must manage FramePending status for all auto-TxAck frames.

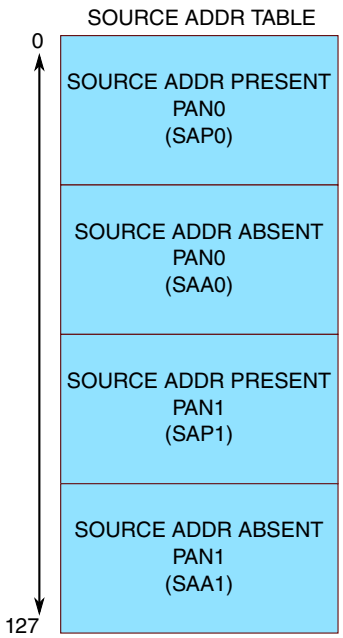
Software can also take direct control of FramePending status for the next auto-TxAck frame at any time (regardless of whether SAP or SAA acceleration is enabled), by setting the ACK_FRM_PND_CTRL bit; in this case, hardware FramePending status will track the state of ACK_FRM_PND.

In order to support Dual PAN mode, a device could theoretically be a PAN Coordinator on both networks. Thus, a coordinator device may need to maintain distinct message pending lists and/or neighbor tables for the 2 PANs. Accordingly, the packet processor further subdivides the Source Address Management table into 4, instead of 2, partitions:

1. SAP0: Source Address Present Partition for PAN0
2. SAA0: Source Address Absent Partition for PAN0

3. SAP1: Source Address Present Partition for PAN1
4. SAA1: Source Address Absent Partition for PAN1

### SOURCE ADDRESS TABLE PARTITIONS




Four register control bits enable each partition individually. If a partition is enabled, and a MAC poll request is received on the corresponding PAN (e.g. SAP0 or SAA0 partitions on PAN0), then the partition is searched by the packet processor, looking for a source address match (or absence), and the result of the search affects the FramePending status; if a partition is disabled, the partition is not searched, and will not influence FramePending status. The register bits are described in the table below.

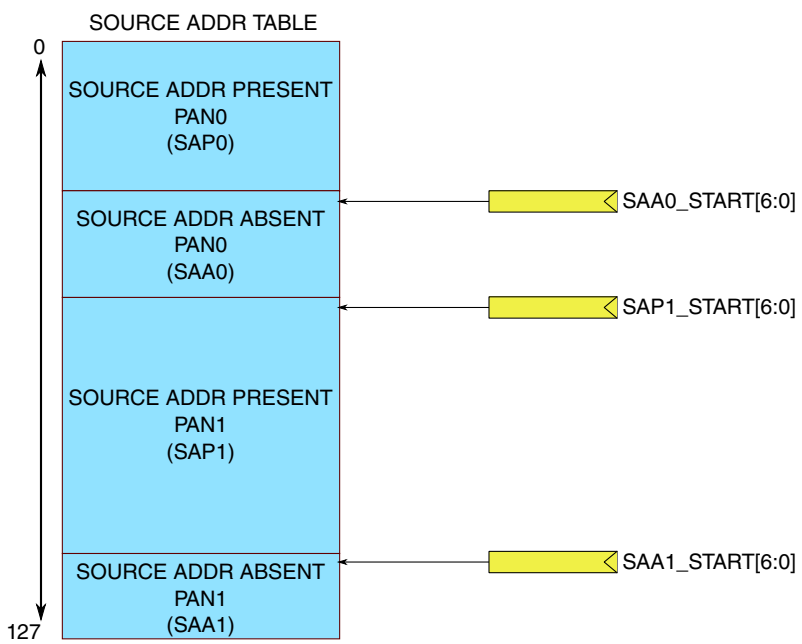
| REGISTER NAME | DESCRIPTION |
|---|---|
| SAP0_EN | Enables SAP0 Partition |
| SAA0_EN | Enables SAA0 Partition |
| SAP1_EN | Enables SAP1 Partition |
| SAA1_EN | Enables SAA1 Partition |

For maximum flexibility in supporting all combinations of message-pending status, neighbor-table status, and dual PAN mode, the dividing-line between all the partitions is fully programmable. The SAP0 partition always starts at table index 0. Three registers control the starting point for the other 3 partitions. The starting point is defined the first table index (0-127) of that partition. The registers are described below, along with some software restrictions to ensure table consistency.

| REGISTER NAME | DESCRIPTION | RESTRICTIONS |
|---|---|---|
| SAA0_START[6:0] | First Index of SAA0 partition. | If SAA0_START=0, then there is no SAP0 partition (SAP0_EN should be set to 0) |
| SAP1_START[6:0] | First Index of SAP1 partition. | Software must ensure that:<br>**SAP1_START ≥ SAA0_START**<br>If SAP1_START = SAA0_START, then there is no SAA0 partition (SAA0_EN should be set to 0) |
| SAA1_START[6:0] | First Index of SAA1 partition. | Software must ensure that:<br>**SAA1_START ≥ SAP1_START**<br>If SAA1_START = SAP1_START, then there is no SAP1 partition (SAP1_EN should be set to 0) |

A diagram which depicts one example of how the dividing-line registers affect partitioning of the table, is shown below.

### SOURCE ADDRESS TABLE PARTITIONS -- NON-EQUAL SIZES (EXAMPLE)




Each table partition consists of one or more indices, which can be used to store source address checksums. The first index for any partition, is defined by the aforementioned *PARTITION*_START[6:0] register, where *PARTITION*_ is one of {SAP0, SAP1, SAA0, SAA1}. The last index for any partition, is defined by the next higher partition's *PARTITION*_START[6:0] register, minus 1. When populating the table with source address information, software must be sure to select an index that is within the defined range for the partition. For example, when installing a checksum in the SAP1 partition:

$$SAP1_START \leq Selected_Index \leq (SAA1_START - 1)$$

To populate the table partitions, a 7-bit register SAM_INDEX[6:0] is provided, as well as a “index enable bit”, and a “index invalidate” bit. A 16-bit SAM_CHECKSUM[15:0] is provided, into which software will write the computed checksum. Installing a checksum into any partition is referred to as “Enabling an Index”, and removing a checksum from any partition is referred to as “Invalidating an Index”. A description of the registers is provided in the following table:

| REGISTER NAME | DESCRIPTION |
|---|---|
| SAM_CHECKSUM[15:0] | Software-computed source address checksum, to be installed into a table index |
| SAM_INDEX[6:0] | Contains the table index to be enabled or invalidated. Software must ensure that the index is within the range of the desired partition. |
| SAM_INDEX_WR | For 32-bit writes, this must be set to indicate that the table entry specified by SAM_INDEX[6:0] is to be written; if SAM_INDEX_WR=0, the table entry is not written, but the SAM_INDEX[6:0] register is updated. For 8-bit writes, this bit is ignored. |
| SAM_INDEX_EN | Enable the index selected by SAM_INDEX[6:0] for searching. |
| SAM_INDEX_INV | Invalidate the index selected by SAM_INDEX[6:0] |
| FIND_FREE_IDX | After modifying Valid bits (enabling or invalidating), write this bit to 1 to force HW to update the “First Free Index” registers to account for the changed Valid bits. This HW update process takes 4us. SW can poll SAM_BUSY to determine when the table update is complete. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| SAM_BUSY | HW is in the process of updating the Source Address table, either in response to a poll indication from the packet processor, or due to SW setting FIND_FREE_IDX=1. In the latter case, SW should poll SAM_BUSY until low before accessing the “First Free Index” registers. Read-only bit. |
| INVALIDATE_ALL | Writing a 1 to this bit clears all 128 Valid bits. Invalidates the entire table. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |

All the index-control registers reside within a single 32-bit word, so that a single 32-bit write can store a new checksum in the table, and simultaneously enable, or invalidate, the index. For 32-bit writes, SAM_INDEX_WR must be set to 1 to cause the SAM_CHECKSUM[15:0] to be loaded into the table at the address specified by SAM_INDEX[6:0]. For 32-bit reads, it is necessary first to update SAM_INDEX[6:0] to the desired table index, if it is not already set to the desired index; this requires a write to SAM_INDEX[6:0] with SAM_INDEX_WR=0, so as to not update the table contents. Then, a subsequent 32-bit read will return the table contents at the desired index.

For systems that don't support 32-bit writes, the register fields should be written in the following order:

1. SAM_INDEX[6:0]
2. SAM_CHECKSUM (the 2 bytes can be written in either order)
3. SAM_INDEX_EN (to enable), or SAM_INDEX_INV (to invalidate)

During operation, when a packet is received and the packet processor searches the table for a source address match, only enabled indices within each partition are searched. Invalidated indices are ignored.

To assist software in finding an available index to store the next source address checksum, the packet processor makes available the following 4 read-only registers:

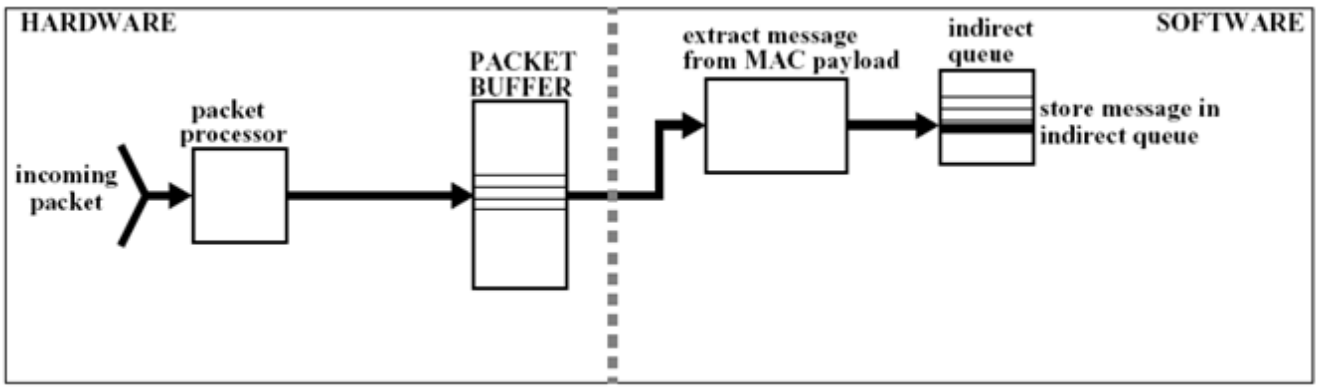
| REGISTER NAME | DESCRIPTION |
|---|---|
| SAP0_1ST_FREE_IDX[6:0] | First non-enabled (invalid) index in the SAP0 partition |
| SAA0_1ST_FREE_IDX[6:0] | First non-enabled (invalid) index in the SAA0 partition |
| SAP1_1ST_FREE_IDX[6:0] | First non-enabled (invalid) index in the SAP1 partition |
| SAA1_1ST_FREE_IDX[6:0] | First non-enabled (invalid) index in the SAA1 partition |

## Operational Use

The following flow description illustrates how hardware and software interact to utilize the Source Addressing Management feature to facilitate pending-message handling and neighbor table management. This flow assumes both SAP and SAA acceleration are enabled, and Dual PAN mode is in effect. There are 11 sequential steps.

### Step 1: Message Packet Reception, Extracting the Message, Indirect Queue Management

A message (data frame) is received by a coordinator. The message is received from one end device, and is intended for another end device. The coordinator's packet processor receives the packet, and stores it in the packet buffer. Coordinator software downloads the packet data from the packet buffer, extracts the message from the MAC payload, and stores the message in its indirect queue.




### Step 2: Software computes address checksum

Coordinator software also extracts the destination information (Destination PANID and Destination Address), from the data packet stored in the packet buffer. Software uses this information to compute a 16-bit checksum. This checksum uniquely identifies the sending device, with a high degree of reliability. The checksum is computed according to the following rules:

| Destination Addressing Mode | Rule |
|---|---|
| Mode 2 (short address) | Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536 |
| Mode 3 (long address) | Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536<br>Checksum = (Checksum + DstAddr[31:16]) % 65536<br>Checksum = (Checksum + DstAddr[47:32]) % 65536<br>Checksum = (Checksum + DstAddr[63:48]) % 65536 |

### Step 3: Software populates the address table

Coordinator software chooses an index in the Source Address Table in which it will store the just-computed checksum. If the data packet was received on PAN0, an index in the SAP0 partition should be chosen. Software may consult the register SAP0_1ST_FREE_IDX[6:0] to find the first available index in this partition. If the data packet was received on PAN1, an index in the SAP1 partition should be chosen. Software may consult the register SAP1_1ST_FREE_IDX[6:0] to find the first available index in this partition.

If an index is chosen which is already valid, software should first invalidate the index with the following procedure:

1. Set SAM_INDEX[6:0] to the selected index
2. Set SAM_INDEX_EN=0 and SAM_INDEX_INV=1

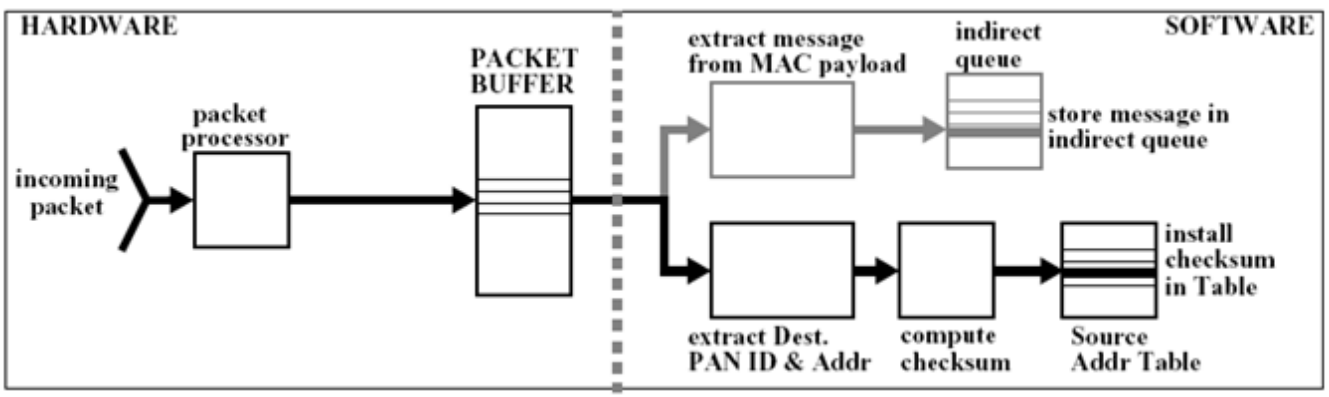
These steps can be performed in a single, 32-bit register write.

To install the source address checksum into the index and enable the index, the following procedure should be used:

1. Program SAM_CHECKSUM[15:0] to the computed checksum
2. Set SAM_INDEX[6:0] to the selected index
3. Set SAM_INDEX_EN=1 and SAM_INDEX_INV=0

These steps can be performed in a single, 32-bit register write.

The index is now enabled, and the packet processor will include it in its search when the next MAC Poll Request is received.




### Step 4: Incoming Poll request, Packet Processor parses the Auxiliary Security Header

An incoming packet is received. The packet processor performs routine packet filtering. After determining that the packet is type ‘MAC Command’, the packet processor must perform additional packet parsing in order to locate the packet’s Command Frame Identifier. This is the octet that distinguishes this commandas being of type “data request”, from other MAC Command types. For the packet processor, this extra parsing is more complex, because the Command Frame Identifier is part of the MAC payload; this is theonly scenario where the packet processor is required to parse the payload. This is further complicated bythe fact that the revision 2006 (and subsequent) of the 802.15.4 wireless MAC standard, includes a variable-length “Auxiliary Security Header”, prior to the MAC payload. This header is only present for FrameVersion > 0, and SecurityEnabled = 1. The packet processor must conditionally parse this header in order to locate the start of the MAC payload, where the Command Frame Identifier is located.The packet processor uses 2 subfields of the Frame Control Field to parse the Auxiliary Security Header (A.S.H.): FrameVersion, and SecurityEnabled. The packet processor parses the Auxiliary Security Header (A.S.H.) according to the following table:

| Length of Auxiliary Security Header (Octets) | SecurityEnabled (Frame Control Field) | FrameVersion (Frame Control Field) | KeyIdentifierMode (Security Control Field) | Remarks |
|---|---|---|---|---|
| **0** | 0 | 00 or 01 | - | If !SecurityEnabled, no ASH regardless of FrameVersion |
| **0** | 1 | 00 | 0 | If packet is 2003-compliant, no ASH |
| **5**<br>Security Control (1 octet) +<br>Frame Counter (4 octet) | 1 | 01 | 0x00 | ASH present. Length of ASH computed by HW. |
| **6**<br>Security Control (1 octet) +<br>Frame Counter (4 octet) +<br>Key Identifier (1 octet) | 1 | 01 | 0x01 | ASH present. Length of ASH computed by HW. |
| **10**<br>Security Control (1 octet) +<br>Frame Counter (4 octet) +<br>Key Identifier (5 octet) | 1 | 01 | 0x2 | ASH present. Length of ASH computed by HW. |
| **14**<br>Security Control (1 octet) +<br>Frame Counter (4 octet) +<br>Key Identifier (9 octet) | 1 | 01 | 0x3 | ASH present. Length of ASH computed by HW. |
| - | - | 10 or 11 | - | HW will handle these FrameVersion's indentically to FrameVersion 01 |

The contents of the A.S.H. are irrelevant to the packet processor; only its length is important. This is because the octet which immediately follows the A.S.H. is the Command Frame Identifier. This is all the packet processor needs to know.

## Step 5: Hardware extracts the Command Frame Identifier

After parsing the A.S.H. (if present), the next octet in the MAC payload, in accordance with the 802.15.4 standard, is the Command Frame Identifier.

| Command frame identifier | Command name | RFD | | Subclause |
|---|---|---|---|---|
| | | Tx | Rx | |
| 0x01 | Association request | X | | 5.3.1 |
| 0x02 | Association response | | X | 5.3.2 |
| 0x03 | Disassociation notification | X | X | 5.3.3 |
| 0x04 | Data request | X | | 5.3.4 |
| 0x05 | PAN ID conflict notification | X | | 5.3.5 |
| 0x06 | Orphan notification | X | | 5.3.6 |
| 0x07 | Beacon request | | | 5.3.7 |
| 0x08 | Coordinator realignment | | X | 5.3.8 |
| 0x09 | GTS request | | | 5.3.9 |
| 0x0a–0xff | Reserved | | | — |

The packet processor checks to see if the Command Frame Identifier = 0x4. If it is not, packet processing for this frame is complete, the remainder of the packet is received, without inspection by the packet processor, and is stored the packet buffer. If the Command Frame Identifier = 0x4, then this is a data request, and assuming the prior rules-checking and address-checking on this packet have already passed, then the Source Address Matching function will be applied to this packet, continuing in STEP 6.

### Step 6: Hardware Computes the Source Address Checksum

The packet processor uses the received Source PAN ID and Source Address for this packet to compute a 16-bit Source Address checksum. This checksum uses precisely the same algorithm that software uses to compute its Address checksum (see STEP 2), except that the *Source* PAN ID and address are used, rather than the Destination PAN ID and address. This checksum uniquely identifies the sending device, with a high degree of reliability. Note if the incoming packet's Frame Control Field has the PANIDCompression subfield asserted, the packet processor will substitute "Destination PAN ID" for "Source PAN ID" in the computation, since Source PAN ID will not be present under these circumstances.

### Step 7: Hardware performs "Source Address Present" Table Search

If the MAC poll request was received on PAN0, and SAP0_EN=1 to enable searches of the PAN0 “Source Address Present” partition, the packet processor uses its Source Address checksum to look for a match in the SAP0 partition of the table. Only table entries that are enabled, are compared against (see STEP 3); table indices which are not enabled, are ignored in the search.

If a single table match is found (i.e., 1 software-computed checksum in SAP0 partition matches the hardware-computed checksum for the incoming packet), this means that there is a message for this requestor in the coordinator’s message queue. For a SAP0 match, the packet processor takes these 2 steps:

1. Set SAP0_ADDR_PRESENT=1
2. Set SAP0_MATCH[6:0] to the index which matched

If multiple table matches are found (i.e., more than 1 software-computed checksum in the SAP0 partition matches the hardware-computed checksum for the incoming packet), this means that there are multiple messages for this requestor in the coordinator’s message queue. In this case, the packet processor takes the same steps as for a single match, with the SAP0_MATCH[6:0] register being set to the lowest-value matching index (i.e., the smallest number) within the SAP0 partition.

If no source address match is found in the SAP0 table search, the packet processor will set SAP0_ADDR_PRESENT=0.

If, on the other hand, the MAC poll request was received on PAN1, and SAP1_EN=1 to enable searches of the PAN1 “Source Address Present” partition, the packet processor uses its Source Address checksum to look for a match in the SAP1 partition of the table. Only table entries that are enabled, are compared against (see STEP 3); table indices which are not enabled, are ignored in the search.

If a single table match is found (i.e., 1 software-computed checksum in SAP1 partition matches the hardware-computed checksum for the incoming packet), this means that there is a message for this requestor in the coordinator’s message queue. For a SAP1 match, the packet processor takes these 2 steps:

1. Set SAP1_ADDR_PRESENT=1
2. Set SAP1_MATCH [6:0] to the index which matched

If multiple table matches are found (i.e., more than 1 software-computed checksum in the SAP1 partition matches the hardware-computed checksum for the incoming packet), this means that there are multiple messages for this requestor in the coordinator’s message queue. In this case, the packet processor takes the same steps as for a single match, with the SAP1_MATCH[6:0] register being set to the lowest-value matching index (i.e., the smallest number) within the SAP1 partition.

If no source address match is found in the SAP1 table search, the packet processor will set SAP1_ADDR_PRESENT=0.

### Step 8: Hardware performs "Source Address Absent" Table Search

If the MAC poll request was received on PAN0, and SAA0_EN=1 to enable searches of the PAN0 "Source Address Absent" partition, the packet processor uses its Source Address checksum to look for a match in the SAA0 partition of the table. Only table entries that are enabled, are compared against (see STEP 3); table indices which are not enabled, are ignored in the search.

If one (or more) table match is found (i.e., a software-computed checksum in SAA0 partition matches the hardware-computed checksum for the incoming packet), this means that the end device which sent the poll request is a valid child of this parent. For a SAA0 match, the packet processor takes these 2 steps:

1. Set SAA0_ADDR_ABSENT=0
2. Set SAA0_MATCH[6:0] to the first index which matched

If *no* source address match is found in the SAA0 table search, the packet processor will set SAA0_ADDR_ABSENT=1.

If, on the other hand, the MAC poll request was received on PAN1, and SAA1_EN=1 to enable searches of the PAN1 "Source Address Absent" partition, the packet processor uses its Source Address checksum to look for a match in the SAA1 partition of the table. Only table entries that are enabled, are compared against (see STEP 3); table indices which are not enabled, are ignored in the search.

If one (or more) table match is found (i.e., a software-computed checksum in SAA1 partition matches the hardware-computed checksum for the incoming packet), this means that the end device which sent the poll request is a valid child of this parent. For a SAA1 match, the packet processor takes these 2 steps:

1. Set SAA1_ADDR_ABSENT=0
2. Set SAA1_MATCH[6:0] to the first index which matched

If *no* source address match is found in the SAA1 table search, the packet processor will set SAA0_ADDR_ABSENT=1.

### Step 9: Hardware determines FramePending status

The packet processor maintains an internal flag called **rx_frame_pending**, which is sent to the TX_PACKET block after every received packet, to directly control the state of the FramePending subfield of the Frame Control Field in the upcoming auto-TxAck frame (if enabled with AUTOACK=1).

If the Source Address Management feature is completely disabled, i.e.,

SAP0_EN = SAA0_EN = SAP1_EN = SAA1_EN = 0

then the packet processor will set the **rx_frame_pending flag** to the value of the ACK_FRM_PND register, putting the state of FramePending under direct software control.

Otherwise, if *any* of the following conditions are met, the packet processor will set its **rx_frame_pending** flag to 1:

1. The MAC Poll Request was received on PAN0, SAP0_EN=1, and SAP0_ADDR_PRESENT=1
2. The MAC Poll Request was received on PAN1, SAP1_EN=1, and SAP1_ADDR_PRESENT=1
3. The MAC Poll Request was received on PAN0, SAA0_EN=1, and SAA0_ADDR_ABSENT=1
4. The MAC Poll Request was received on PAN1, SAA1_EN=1, and SAA1_ADDR_ABSENT=1

If none of the conditions are met, **rx_frame_pending** flag will be set to 0.

### Step 10: Receive interrupt RXIRQ

The packet processor receives the remainder of the packet, and verifies the FCS. Assuming the CRC check passes, the RXIRQ interrupt is issued. In the coordinator software's RXIRQ Interrupt Service Routine, software can determine if the received packet was a MAC Poll Request by reading the PI bit of the IRQSTS register. If PI=0, the received packet was not a poll request, and no action relating to Source Address Matching need be taken.

Otherwise, if *either* SAP0_ADDR_PRESENT=1 or SAP1_ADDR_PRESENT=1, an auto-TxAck packet is being sent with the FramePending subfield set to 1, so software should follow up by constructing a data packet (or packets) for the requesting end device, with a payload consisting of the message for this device which had been stored in the coordinator's indirect queue.

Also, if *either* SAA0_ADDR_ABSENT=1 or SAA1_ADDR_ABSENT=1, an auto-TxAck packet is being sent with the FramePending subfield set to 1, so software should follow up by constructing a “Leave Message” for the requesting end device.

### Step 11: Software disables the spent index

In the final step, coordinator software must disable (invalidate) the index which was just used to locate and send the stored message to the requesting end device. To disable the index:

1. Set SAM_INDEX[6:0] to the selected index
2. Set SAM_INDEX_EN=0 and SAM_INDEX_INV=1

At this point, the disabled index is now an available table entry, which can be used to install a new Address checksum for another message in the indirect queue.

#### *64.5.2.4.1.3.4 Dual PAN Mode*

In the past, radio transceivers designed for 802.15.4 applications allow a device to associate to one and only one PAN (Personal Area Network), at any given time. The Packet Processor includes hardware support for a device to reside on two networks simultaneously. In optional Dual PAN mode, the device will alternate between the 2 PAN’s, under hardware or software control. Hardware support for Dual PAN operation consists of 2 sets of PAN and IEEE addresses for the device, 2 different channels (one for each PAN), a programmable timer to automatically switch PAN’s (including on-the-fly channel-changing) without software intervention, control bits to configure and enable Dual PAN mode, and read-only bits to monitor status in Dual PAN mode. A device can be configured to be a PAN Coordinator on either network, both networks, or neither.

For the purpose of defining a "PAN" in the context of Dual PAN mode, two sets of network parameters are maintained. In this document, "PAN0" and "PAN1" will be used to refer to the 2 PAN’s. Each parameter set uniquely identifies a PAN for Dual PAN mode. These parameters are:

| PAN0 | PAN1 |
|---|---|
| CHANNEL_NUM0 | CHANNEL_NUM1 |
| MACPANID0 (16-bit register) | MACPANID1 (16-bit register) |
| MACSHORTADDRS0 (16-bit register) | MACSHORTADDRS1 (16-bit register) |
| MACLONGADDRS0 (64-bit registers) | MACLONGADDRS1 (64-bit registers) |
| PANCORDNTR0 (1-bit register) | PANCORDNTR1 (1-bit register) |

During device initialization, if Dual PAN mode is to be utilized, software will program both parameters sets, to configure the hardware for operation on two networks.

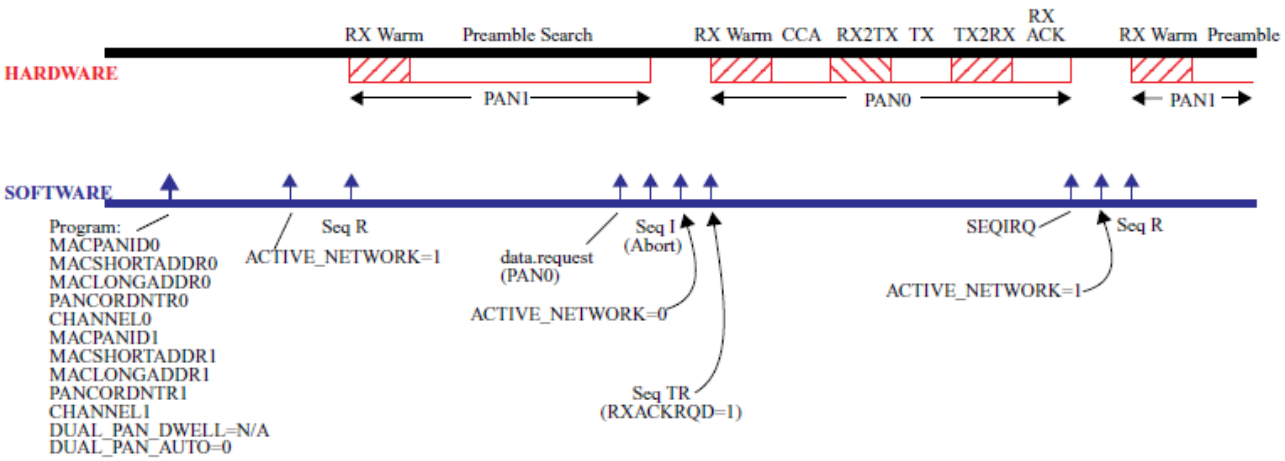
Two modes are available for Dual PAN operation: manual and automatic.

## Manual Dual PAN

In manual Dual PAN mode, software controls which PAN is selected for transmission and reception, at all times. Software populates both PAN network parameter-set registers listed above. Software then selects a PAN to begin transmission or reception. A single control bit **ACTIVE_NETWORK**, selects the PAN. To select PAN0, **ACTIVE_NETWORK** should be set to 0; to select PAN1, **ACTIVE_NETWORK** should be set to 1. Software then initiates an autosequence by writing the appropriate value to XCVSEQ (PHY_CTRL1 Register). At any point, software can elect to switch to the alternate PAN. To do so, software aborts the sequence (if one is active) by writing XCVSEQ=IDLE. Software then toggles **ACTIVE_NETWORK**, and restarts a new sequence with XCVSEQ. Software is responsible for aborting and initiating sequences. Hardware is responsible for warming up the transceiver on the selected channel (CHANNEL_NUM0 or CHANNEL_NUM1), and performing address-filtering on the selected network's address parameters (PANCORDNTR, MacPanID, Short Address, and/or IEEE address). Software overhead is significantly reduced in this mode, since there is no need to re-program all of the the PAN-select registers every time the PAN is toggled in Dual PAN mode; only a single bit (**ACTIVE_NETWORK**) need be written. For manual Dual PAN operation, **DUAL_PAN_AUTO** should be set to 0.

The following diagram depicts Dual PAN operation in manual mode. In this example, software populates both PAN parameters sets during initialization. In this example, software elects to initiate packet reception on PAN1, so **ACTIVE_NETWORK** is set to 1. Software initiates the Sequence R by writing to XCVSEQ. Hardware warms up the receiver on CHANNEL_NUM1 and enters preamble-search. In this example, no packet is received, but a data request from higher-level software requires a switch to PAN0. Software responds by aborting the reception by writing Sequence I to XCVSEQ. Software then sets **ACTIVE_NETWORK** to 0 to select PAN0. In the example, software initiates a Sequence TR. Hardware warms up the receiver for CCA on CHANNEL_NUM0, performs CCA (channel is idle), warms down the receiver, and warms up the transmitter on CHANNEL_NUM0, transmits the requested packet, warms down the transmitter and warms up the receiver again (as per the autosequence) on CHANNEL_NUM0, to receive the acknowledge packet. The acknowledge packet is received and the autosequence completes successfully with RXIRQ and SEQIRQ. Software elects to resume packet reception on PAN1 (which was in progress before the data request). Software toggles **ACTIVE_NETWORK** back to 1, and re-initiates Sequence R.

**DUAL PAN MANUAL MODE**




## Auto Dual PAN

In automatic Dual PAN mode, hardware is responsible for alternating the selected PAN, at a pre-programmed interval. Software programs a "PAN dwell time", which determines the amount of time during which the hardware will receive on a given PAN. This is done by programming the **DUAL_PAN_DWELL** register. Software sets the **DUAL_PAN_AUTO** bit to 1 for automatic Dual PAN operation, and sets **ACTIVE_NETWORK** to correspond to the PAN on which it desires packet reception to begin. Software then initiates a Sequence R. Hardware warms up the receiver on the selected PAN, and begins preamble search. When the dwell time expires, assuming preamble and SFD have not been detected, hardware will switch to the alternate PAN. This includes an on-the-fly channel change. Once switched to the alternate PAN, hardware will use the address filtering parameters (PANCORDNTR, MacPanID, Short Address, and/or IEEE address) programmed for the alternate PAN. Once the dwell time expires on the alternate PAN, assuming preamble and SFD have not been detected, hardware will switch back to the original PAN. This process will repeat indefinitely, until one of the following occurs:

1. A packet is successfully received (address match and CRC valid).
2. Software aborts the Sequence R
3. A TMR3 RX timeout, which automatically aborts the Sequence R
4. A PLL unlock automatically aborts the Sequence R (if unlock aborting is enabled)

If the hardware PAN dwell timer expires *during* a packet reception (i.e., preamble and SFD have been detected during a Sequence R), the PAN-switch will be delayed until the packet is completely received. If the packet is received successfully (address match and

CRC valid), the receive sequence will terminate with RXIRQ and SEQIRQ (if an auto-Acknowledge was requested and enabled, the sequence will complete after the Acknowledge packet was transmitted, and TXIRQ will also be set, in addition to RXIRQ and SEQIRQ); the PAN switch will occur at this point, after the Sequence has returned to IDLE.

If the received packet fails FCS (CRC check), or, if the packet was not intended for the device (address mismatch), the hardware will toggle the PAN, and return to preamble-search on the new PAN.

If the hardware PAN dwell timer expires during a state *other* than RX preamble search during Sequence R (i.e., if expiration occurs during TX, CCA, ED, Sequence TR, or any warmup or transitional state), PAN-switch will be delayed until either:

1. Preamble Search state is reached during a Sequence R (not Sequence TR)

2. Sequence returns to IDLE.

At any point, software can elect to interrupt the automatic Dual PAN operation, by aborting the sequence with a XCVSEQ=IDLE, and the setting the **DUAL_PAN_AUTO** bit to 0. In response, hardware will freeze the current state of the dwell timer, and capture the currently-selected PAN, so that hardware-controlled PAN alternation can resume where it left off, later. Software can then tend to that tasks that caused the Dual PAN interruption. When software wants to resume Dual PAN operation, it then sets **DUAL_PAN_AUTO**=1, and re-initiates Sequence R. Hardware will resume the dwell timer from the point it was frozen earlier, on the same PAN, and will warmup the receiver to receive on the PAN which was being monitored before the interruption. Hardware will dwell on that PAN until the timer expires, and then switch PAN's (assuming no preamble and SFD detected, subject to the conditions listed above). Software can always determine the selected PAN at any time, in any mode, by reading **CURRENT_NETWORK** bit. If **DUAL_PAN_AUTO**=0, then **CURRENT_NETWORK** = **ACTIVE_NETWORK**. If **DUAL_PAN_AUTO**=1, then **CURRENT_NETWORK** is controlled by hardware.

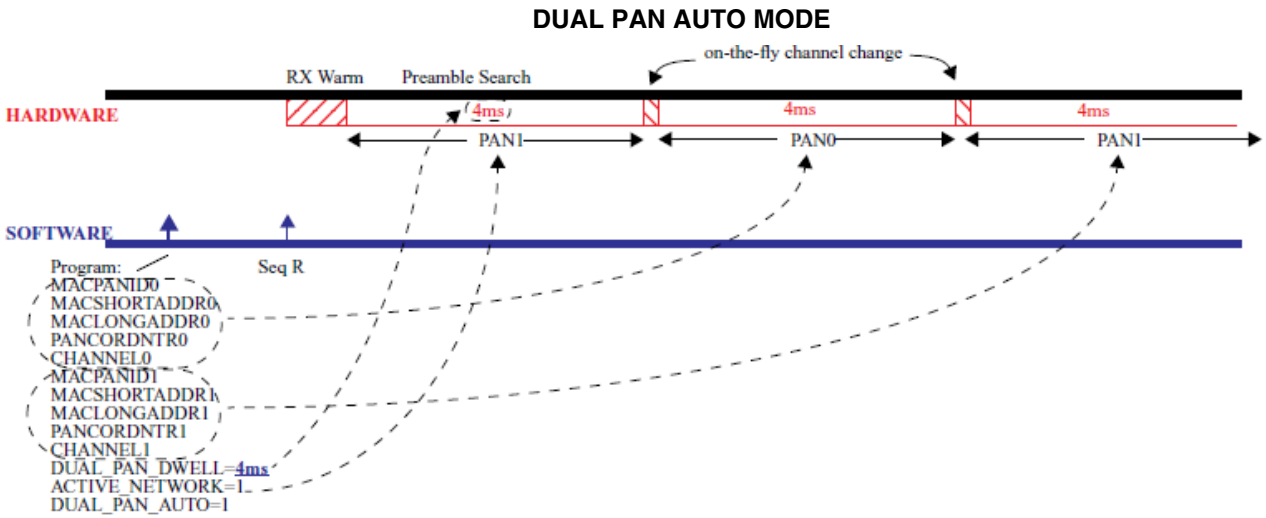
To initiate auto Dual PAN operation for the first time, or to reset the dwell timer to its programmed value and allow software to select the initial PAN for auto Dual PAN operation, software should program the desired initial PAN to **ACTIVE_NETWORK** (0 for PAN0, 1 for PAN1), and then program the desired dwell time into the **DUAL_PAN_DWELL** register. Hardware always responds to a write to **DUAL_PAN_DWELL**, by resetting the dwell timer to its programmed value, and selecting the initial PAN for auto Dual PAN operation, based on the state of **ACTIVE_NETWORK**. A write to **DUAL_PAN_DWELL**, always re-initilizes the DWELL TIMER to the programmed value. If a write to **DUAL_PAN_DWELL** occurs during an autosequence, the DWELL TIMER will begin counting down immediately. If a

write to **DUAL_PAN_DWELL** occurs when there is no autosequence underway, the DWELL TIMER will not begin counting until the next autosequence begins; it will begin counting at the start of the autosequence warmup. Under all circumstances, **DUAL_PAN_AUTO** must be asserted to allow the DWELL TIMER to count. If **DUAL_PAN_AUTO**=0, the DWELL TIMER will freeze and hold its state, until **DUAL_PAN_AUTO**=1. However, a write to **DUAL_PAN_DWELL** will still initialize the DWELL TIMER to its programmed value, regardless of the state of **DUAL_PAN_AUTO**.

At any point during auto Dual PAN mode, software can ascertain the PAN currently being serviced by hardware, by reading the **CURRENT_NETWORK** bit. Software can also determine the time-remaining in the current dwell, by reading the **DUAL_PAN_REMAIN** Register.

In auto Dual PAN mode, software retains responsibility for initiating and aborting sequences. There is no provision for hardware-initiated sequences. TMR2 may be used to initiate an autosequence at a future time. TMR3 may be used as an RX hardware-abort mechanism.

The following diagram depicts Dual PAN operation in automatic mode. In this example, software populates both PAN parameters sets during initialization. In this example, software wants to initiate packet reception in auto Dual PAN mode, on PAN1, so **ACTIVE_NETWORK** is set to 1. For auto Dual PAN operation, software programs the desired PAN-switching interval into **DUAL_PAN_DWELL** register, in this case, 4ms. Software initiates the Sequence R by writing to XCVSEQ. The dwell timer begins counting as soon as the receiver warmup begins. Hardware warms up the receiver on CHANNEL_NUM1 (since PAN1 was selected as the initial PAN in this example), and begins preamble search. If a packet is received during this interval, hardware applies PAN1 address filtering. In this example, no preamble is detected at the point at which the dwell timer expires. So, at this point, hardware executes a PAN-switch to PAN0. which includes an on-the-fly channel change. The on-the-fly channel change requires approximately 93us and no preamble detection is possible during this "blind spot". (For more details on how the hardware executes the Dual PAN on-the-fly channel change, see Chapter ZSM Sequence Manager, Section Dual PAN Mode.) After the PAN switch, hardware resumes preamble search on CHANNEL_NUM0, and applies PAN0 address filtering if a packet is received. In this example, no preamble is detected at the point at which the dwell timer expires. So, at this point, hardware performs a PAN-switch back to PAN1. And so on and so forth.




## Two-PAN-one-channel

A special case occurs if a device is operating in Dual PAN mode (either Auto or Manual mode), and both PAN's occupy the same channel; in other words, CHANNEL_NUM0=CHANNEL_NUM1. In this case, hardware will detect this condition automatically, and apply both sets of address-filtering parameters simultaneously. Hardware will provide Data Indication if the received packet matches either the PAN0 address parameter set {**MACPANID0, MACSHORTADDRS0, MACLONGADDRS0, PANCORDNTR0**}, or the PAN1 address parameter set {**MACPANID1, MACSHORTADDRS1, MACLONGADDRS1, PANCORDNTR1**}. At Data Indication, two status bits can be queried by software to quickly determine on which PAN the packet was received, **RECD_ON_PAN0** or **RECD_ON_PAN1**. In the "Two-PAN-one-channel" scenario, it is possible for a packet to satisfy both sets of addressing parameters (PAN0 and PAN1). When this happens, both **RECD_ON_PAN0** and **RECD_ON_PAN1** will be set. In the "Two-PAN-one-channel" scenario, the **CURRENT_NETWORK** bit should be ignored, since both networks are simultaneously active.

If Dual PAN mode is not to be utilized, **ACTIVE_NETWORK** should be maintained at 0 (default) to select the PAN0 parameters for transceiving. In single PAN operation, the PAN1 parameter registers need not be programmed, and can be used as scratchpad. CHANNEL_NUM1 should not be programmed, and should instead be left in its default state, if Dual PAN mode is not in use.

## Dual PAN Channel Override

In Dual PAN mode, in case there is a need to generate a frequency which may be offset from the 16 prescribed 5MHz-spaced channels, to, for example, avoid interference on one of the Dual PAN channels, a method has been provided to do that, by designating one of the two PAN channels to use the transceiver's set of direct frequency-programming registers, instead of CHANNEL_NUMx. Programming the direct frequency-programming registers – integer, numerator, and denominator, allows an RF frequency to be selected with much more precision than the 5MHz granularity of the 802.15.4 mapped-channel registers, CHANNEL_NUM0 and CHANNEL_NUM1. This is referred to as "Dual PAN Channel Override". For more details on this feature, see Chapter ZSM Sequence Manager, Section Dual PAN Channel Override.

#### *64.5.2.4.1.3.5 Active Promiscuous Mode*

**Functional description**

The 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) defines the following behavior for Promiscuous Mode operation:

> "In promiscuous mode, the MAC sublayer shall pass all frames received after the first filter directly to the upper layers without applying any more filtering or processing."

Where first level filtering is described as:

> "For the first level of filtering, the MAC sublayer shall discard all received frames that do not contain a correct value in their FCS field in the MFR (see 7.2.1.9)"

Promiscuous mode allows a device to receive and process all 802.15.4 frames which pass FCS check (CRC), regardless of whether or not they are addressed to the device. This allows a device to receive and process all valid traffic on a network. In Promiscuous mode, the Packet Processor will provide a Data Indication on all such packets. This capability allows a simple network sniffer to be constructed, which can display comprehensive packet data for all packets transmitted on a network within receiving range of the device.

Since a device operating in promiscuous mode is designed to do so discreetly, automatic acknowledgement of all received packets is inhibited, regardless of the state of the Ack Request field in the Frame Control Field of the received packet; otherwise, the sniffing device would be acknowledging received frames for which it was not the addressee.

The Packet Processor features an "Active Promiscuous Mode". In this mode, the Packet Processor will process packets according to the same rules that it applies to Promiscuous mode, except that it will automatically transmit an acknowledgement packet *for frames*

*that are addressed to the device*. In other words, in Active Promiscuous mode, if the received frame meets all the address and packet filtering requirements that would have applied in non-promiscuous mode, an acknowledgement packet will be transmitted; a Data Indication will still be provided for all valid packets (those which pass FCS check). Note that all existing packet processor rules governing transmission of acknowledge packets, are still in effect; i.e., AUTOACK bit of the PHY_CTRL1 Register must be set to 1, and the Ack Request field in the Frame Control Field of the received packet must be set to 1.

The Packet Processor can be also programmed to provide a Data Indication for packets which fail FCS check, by setting CRC_MSK=0. Received packets which fail FCS check are never acknowledged, under any circumstances.

## Special Handling for Broadcast Packets

Special rules apply to received packets which contain a broadcast PAN Identifier (0xffff), and/or a broadcast short address (0xffff). According to the 802.15.4 standard,

> “For valid frames that are not broadcast, if the Frame Type subfield indicates a data or MAC command frame and the Acknowledgment Request subfield of the Frame Control field is set to one, the MAC sublayer shall send an acknowledgment frame."

An exception is made for received packets which contain a broadcast PAN Identifier, and a long address which matches the device’s IEEE address. Such packets are acknowledged, because the IEEE address uniquely identifies the device, regardless of PAN. This special case will apply also in Active Promiscuous mode; such packets will be acknowledged. The following diagram describes the packet processor’s "acknowledgement decision" for all permutations of broadcast indicators in the received packet’s addressing fields.

ACKNOWLEDGE-ON-BROADCAST (ALL PERMUTATIONS)

| DESTINATION ADDRESS MODE | PAN ID = !BROADCAST<br>DSTADDR = !BROADCAST | PAN ID = !BROADCAST<br>DSTADDR = BROADCAST | PAN ID = BROADCAST<br>DSTADDR = BROADCAST | PAN ID = BROADCAST<br>DSTADDR = !BROADCAST |
|---|---|---|---|---|
| SHORT | ACK | NO ACK | NO ACK | NO ACK |
| LONG | ACK | | | ACK |

## Special Handling for Broadcast Packets

The Packet Processor features an autosequence to automatically receive, and verify, Acknowledge Frames after a Transmit frame has been sent by the device. This is accomplished by programming a Sequence TR with RXACKRQD=1. During normal operation, these receive Acknowledge Frames are not stored in the Packet Buffer. They are merely checked for matching Sequence Number and Frame Version, with a Data Indication issued for a valid match on both. In Active Promiscuous Mode, receive Acknowledge frames *will* be stored into the Packet Buffer, just like any other received frame. This saves software from having to re-construct the contents of the Acknowledge packet, from the contents of the packet which was originally transmitted. Ordinary Promiscuous Mode should therefore not be used with Sequence TR with RXACKRQD=1. Use Active Promiscuous instead.

#### *64.5.2.4.1.3.6 Provisions to Enable Software Support for 802.15.4e and 802.15.4-2015*

Addendum 802.15.4e to the 802.15.4-2011 standard introduces 3 new variations on the 802.15.4 MAC Frame Structure:

1. FrameVersion=2 (Frame Types: Beacon, Acknowledge, Data, and MAC Command)
2. LLDN Frame Type
3. Multipurpose Frame Type

In addition, the 802.15.4-2015 revision to the standard introduces a new Frame Type: Extended (Frame Type = 0b111).

All three of the “4e-compliant packets”, as well as the Extended Frame Type, modify the format and definition of the Frame Control Field in such a way as to render parsing of the MHR by the packet processor impossible, without major upgrades to the hardware. For example, the packet processor hardware does not support 8-bit (simple) addressing, nor Sequence Number Suppression.

As in interim solution, provisions have been added to the packet processor to allow these new packet variations to be recognized and conditionally admitted (Data Indication to software). Three new Frame Type control bits, and a modification of the Frame Version Filtering bits, are included in the Packet Processor, and affect hardware packet parsing as indicated in the following table:

| REGISTER | DEFINITION |
|---|---|
| FRM_VER_FILTER[3:0] | Frame Version selector. The incoming packet's Frame Control Field is parsed to obtain the FrameVersion subfield, and that value is compared against this register, in accordance with the following:<br><br>xxx1: Accept received packets with FrameVersion=00<br>xx1x: Accept received packets with FrameVersion=01<br>x1xx: Accept received packets with FrameVersion=10<br>1xxx: Accept received packets with FrameVersion=11<br><br>These filtering rules apply to Beacon, Acknowledge, Data, and MAC Command Frame Types, since these frame types require a 2-octet Frame Control Field which embeds a 2-bit FrameVersion subfield. Later frame types introduced in the 802.15.4e addendum (LLDN, Multipurpose) don’t guarantee a FrameVersion subfield with the original meaning, so these filtering rules do not apply to those frame types. See registers LLDN_FT and MULTIPURPOSE_FT. |
| LLDN_FT | 1: LLDN frame type enabled (Frame Type 4)<br>0: reject all LLDN frames |
| MULTIPURPOSE_FT | 1: Multipurpose frame type enabled (Frame Type 5)<br>0: reject all Multipurpose frames |
| EXTENDED_FT | 1: Extended frame type enabled (Frame Type 7)<br>0: reject all Extended frames |

The Extended Frame Type does not comply with the General MAC Frame Format, which is shown in the following diagram:

| Octets: 1/2 | 0/1 | 0/2 | 0/2/8 | 0/2 | 0/2/8 | variable | variable | | variable | 2/4 |
|---|---|---|---|---|---|---|---|---|---|---|
| Frame Control | Sequence Number | Destination PAN Identifier | Destination Address | Source PAN Identifier | Source Address | Auxiliary Security Header | Information Elements (IE) | | Frame Payload | FCS |
| | | Addressing fields | | | | | Header IEs | Payload IEs | | |
| MHR | | | | | | | MAC Payload | | | MFR |

Figure 86—General MAC frame format

**Figure 64-145. General MAC Frame Format**

Instead, the Extended Frame has its own unique format, which is shown in the following diagram:

| Bits: 0-2 | 3-5 | variable |
|---|---|---|
| Frame Type | Extended Frame Type | Extended Frame Payload |

Figure 105—Extended frame format

**Figure 64-146. Extended Frame Format**

Because the Extended Frame Payload, which is not defined in the 802.15.4-2015 revision to the standard, would be intended to contain and fully specify the FCS, the Packet Processor will not be able to verify FCS on received packets with Frame Type Extended. FCS checking will need to be implemented in software.

**Note:** If the EXTENDED_FT bit is set to enable Data Indication on packets of Frame Type Extended, software should expect a substantial increase in the frequency of notifications, as a consequence of the lack of hardware verification of FCS, which will have implications for software workload as well as power consumption in the SoC.

When the packet processor receives any of the 4e-compliant frames, if the corresponding control bit is set, the hardware will skip Stage 2 filtering, and merely verify FCS. If FCS passes, the packet processor will notify software of the packet, without inspection of the addressing fields or payload. For such packets, software will be responsible for parsing the MHR and payload sections of the frame. If the corresponding control bit is not set, the packet processor will reject the received packet and there will be no Data Indication.

When the packet processor receives a packet of Frame Type Extended, if the EXTENDED_FT bit is set, the hardware will skip Stage 2 filtering, and after the entire packet is received, the packet processor will merely notify software of the packet, without inspection of the Extended Frame Type bits or the Extended Frame Payload. For such packets, software will be responsible for parsing the Extended Frame Type and the Extended Frame Payload sections of the frame. If the corresponding control bit is not set, the packet processor will reject the received packet and there will be no Data Indication.

Like all existing frames prior to 802.15.4e and 802.15.4-2015, a received frame may require an acknowledgement. In the case of the new 802.15.4e and 802.15.4-2015 frames, the required timing for the acknowledge frame transmission, has been modified. The acknowledgement no longer immediately follows the packet reception. For this reason, upon reception of any of these 4e-compliant 2015-compliant packets, the packet processor will cancel any auto-ACK transmission, regardless of the state of the PHY_CTRL[AUTOACK] bit and AR field of the received packet's Frame Control Field. Software will be responsible for constructing the acknowledgment packet and transmitting it at the appropriate time using a Sequence T.

To help expedite software processsing of the 4e- and 2015-compliant Frame Types and Versions, additional status bits have been added to the 802.15.4 register set, which are described in the following table:

| STATUS BIT | RESIDES IN REGISTER | DEFINITION |
|---|---|---|
| ENH_PKT_STATUS | IRQSTS | 1: The last packet received was 4e- or 2015-compliant (SW should query the RX_FRAME_FILTER register for additional status bits)<br>0: The last packet received was neither 4e- nor 2015-compliant |
| BEACON_FV2_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType Beacon with FrameVersion 2<br>0: The last packet received was not a FrameType Beacon with FrameVersion 2 |
| DATA_FV2_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType Data with FrameVersion 2<br>0: The last packet received was not a FrameType Data with FrameVersion 2 |
| ACK_FV2_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType Acknowledge with FrameVersion 2<br>0: The last packet received was not a FrameType Acknowledge with FrameVersion 2 |
| CMD_FV2_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType MAC Command with FrameVersion 2<br>0: The last packet received was not a FrameType MAC Command with FrameVersion 2 |
| LLDN_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType LLDN<br>0: The last packet received was not a FrameType LLDN |

*Table continues on the next page...*

| | | |
|---|---|---|
| MULTIPURPOSE_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType Multipurpose with FrameVersion 2<br>0: The last packet received was not a FrameType Multipurpose |
| EXTENDED_RECD | RX_FRAME_FILTER | 1: The last packet received was a FrameType Extended<br>0: The last packet received was not a FrameType Extended |

The new status bits are mapped into the 802.15.4 Address space as shown in the diagram below:

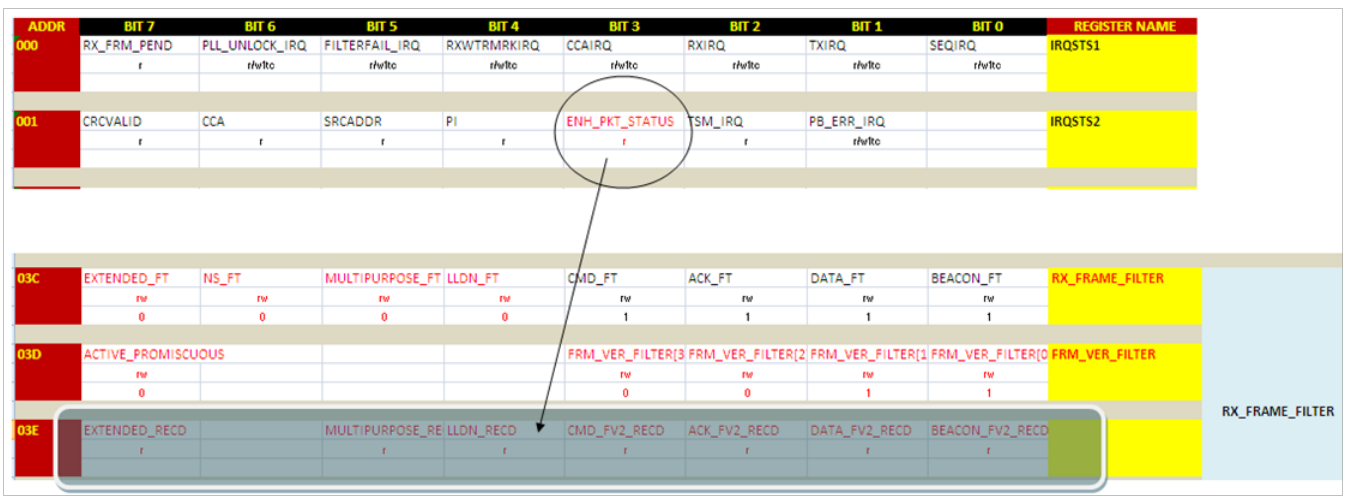



**Figure 64-147. Enhanced Packet Status Bits**

#### *64.5.2.4.1.3.7 Clocks*

Packet Processor is a fully synchronous module. It has a single clock input **clk_250khz**. The clock frequency, 250KHz, is the 802.15.4 bit rate. This clock input is driven by a divide-down of the RF Oscillator, generated in the CRM (Clocks and Resets module). The Packet Processor has no asynchronous interfaces.

#### *64.5.2.4.1.3.8 Reset*

The Packet Processor has a single, active-low, asynchronous reset input: **rst_b** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

#### *64.5.2.4.1.3.9 Interrupts*

The Packet Processor does not generate any interrupts.

*64.5.2.4.1.3.10 Appendix A*

## FILTERFAIL CODES

A description of the FILTERFAIL codes can be found in the following table.

| FILTERFAIL CODE BIT | REASON FOR FILTERFAIL |
|---|---|
| [0] | Fails Stage 1 Frame Length Checking (FL < 5 or FL > MAXFRAMELENGTH)<br>**Note:** FL < 3 will not generate an SFD, so this bit will not be set |
| [1] | Fails Stage 1 Section 7.2.1.1.6 or Section 7.2.1.1.8 Checking (DST_ADDR_MODE or SRC_ADDR_MODE = 1) |
| [2] | Fails Stage 1 Section 7.2.1.1.5 Checking (Illegal PAN_ID_COMPRESSION Usage) |
| [3] | Fails Stage 1 Frame Version Checking |
| [4] | Fails Stage 2 Auto-RxAck Checking (Illegal Ack Frame Format in Sequence TR) |
| [5] | Fails Stage 2 Frame Type Checking (Incorrect Frame Filter Bit setting) |
| [6] | Fails Stage 2 Frame Length Checking (Illegal Beacon, Data, or Cmd FL) |
| [7] | Fails Stage 2 Addressing Mode Checking (Illegal Addressing Mode for Beacon, Data, OR Cmd) |
| [8] | Fails Stage 2 Sequence Number Matching (Sequence TR Only) |
| [9] | Fails Stage 2 PAN ID or Address Checking (Beacon, Data, or Cmd) |

*64.5.2.4.1.3.11 Appendix B*

## LENIENCY BITS

The Packet Processor performs filtering on all received packets, in order to determine whether the packet is intended for the device. The packet filtering is based on rules. In case any of the packet filtering rules need to be overriden, a 40-bit "leniency register" has been provided. When the leniency register is programmed to its default value (0), all hardware packet filtering rules are in effect, and if an incoming packet violates any rule, a "Filter Fail" will occur (packet will be rejected). When a given leniency register bit is asserted, the packet filtering rule assigned to that bit will not be in effect, and if any incoming packet violates that rule (but no other rules), then a "Filter Fail" will not occur, the packet will not be rejected, the packet will be treated as "intended for the device", and software will be notified of the incoming packet. The table below shows the assignment of leniency bits to packet filtering rules.

| LENIENCY BIT | PACKET FILTERING RULE OVERRIDDEN |
|---|---|
| leniency[0] | Override Stage 1 Frame Length Checking (FL < 5 or FL > MAXFRAMELENGTH) |
| leniency[1] | Override Stage 1 Section 7.2.1.1.6 Checking (DST_ADDR_MODE = 1) |
| leniency[2] | Override Stage 1 Section 7.2.1.1.8 Checking (SRC_ADDR_MODE = 1) |
| leniency[3] | Override Stage 1 Section 7.2.1.1.5 Checking (Illegal PAN_ID_COMPRESSION Usage) |

*Table continues on the next page...*

| LENIENCY BIT | PACKET FILTERING RULE OVERRIDDEN |
|---|---|
| leniency[4] | Override Stage 2 Auto-RxAck Checking (Illegal Ack Frame Format in Sequence TR) |
| leniency[5] | Override Stage 2 Frame Length Checking (Illegal Beacon, Data, or Cmd FL) |
| leniency[6] | Override Stage 2 Beacon Frame Address Mode Violations |
| leniency[7] | Override Stage 2 Data Frame Address Mode Violations |
| leniency[8] | Override Stage 2 MAC Command Frame Address Mode Violations |
| leniency[9] | Override Stage 2 Sequence Number Matching |
| leniency[10] | Override Stage 2 Filter for DST_ADDR_MODE_NONE/SRC_ADDR_MODE_SHORT (Beacon Only) |
| leniency[11] | Override Stage 2 Filter for DST_ADDR_MODE_NONE/SRC_ADDR_MODE_SHORT (Data and MAC Command Only) |
| leniency[12] | Override Stage 2 Dst PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_NONE (Data and MAC Command Only) |
| leniency[13] | Override Stage 2 Dst Short Addr Filter for DST_ADDR_MODE_SHORT/ SRC_ADDR_MODE_NONE (Data and MAC Command Only) |
| leniency[14] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_SHORT/ PAN_ID_COMPRESSION (Beacon Only) |
| leniency[15] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_SHORT/ PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[16] | Override Stage 2 Dst Short Addr Filter for DST_ADDR_MODE_SHORT/ SRC_ADDR_MODE_SHORT/PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[17] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESSION (Beacon Only) |
| leniency[18] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESS ION (Data and MAC Command Only) |
| leniency[19] | Override Stage 2 Dst Addr Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[20] | Override Stage 2 Src PAN ID Filter for DST_ADDR_MODE_NONE/SRC_ADDR_MODE_LONG/ NO_PAN_ID_COMPRESSION (Beacon Only) |
| leniency[21] | Override Stage 2 Src PAN ID Filter for DST_ADDR_MODE_NONE/SRC_ADDR_MODE_LONG/ NO_PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[22] | Override Stage 2 Dst PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_NONE (Data and MAC Command Only) |
| leniency[23] | Override Stage 2 Dst Long Addr Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_NONE (Data and MAC Command Only) |
| leniency[24] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_LONG/ PAN_ID_COMPRESSION (Beacon Only) |
| leniency[25] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_LONG/ PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[26] | Override Stage 2 Dst Long Addr Filter for DST_ADDR_MODE_SHORT/ SRC_ADDR_MODE_LONG/PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[27] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_SHORT/ PAN_ID_COMPRESSION (Beacon Only) |
| leniency[28] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_SHORT/ PAN_ID_COMPRESSION (Data and Mac Command Only) |
| leniency[29] | Override Stage 2 Dst Long Addr Filter for DST_ADDR_MODE_LONG/ SRC_ADDR_MODE_SHORT/PAN_ID_COMPRESSION (Data and Mac Command Only) |

*Table continues on the next page...*

| LENIENCY BIT | PACKET FILTERING RULE OVERRIDDEN |
|---|---|
| leniency[30] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_LONG/ NO_PAN_ID_COMPRESSION (Beacon Only) |
| leniency[31] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_LONG/ NO_PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[32] | Override Stage 2 Short Addr Filter for DST_ADDR_MODE_SHORT/SRC_ADDR_MODE_LONG/ NO_PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[33] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESSION (Beacon Only) |
| leniency[34] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESSI ON (Data and MAC Command Only) |
| leniency[35] | Override Stage 2 Long Addr Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_SHORT/ NO_PAN_ID_COMPRESSION (Data and MAC Command Only) |
| leniency[36] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_LONG (Beacon Only) |
| leniency[37] | Override Stage 2 PAN ID Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_LONG (Data and MAC Command Only) |
| leniency[38] | Override Stage 2 Long Addr Filter for DST_ADDR_MODE_LONG/SRC_ADDR_MODE_LONG (Data and MAC Command Only) |
| leniency[39] | Allow an auto-TxAck frame to be sent, after a receive frame which has all of the following parameters:<br>1. Destination PAN ID = Broadcast (0xFFFF)<br>2. Destination Address = !Broadcast (not 0xFFFF)<br>3. Destination Address Mode = Short<br><br>Nominally, the SEQ_MGR inhibits auto-TxAck on such frames. |

## 64.5.2.4.2 Sequence Manager

### 64.5.2.4.2.1 Introduction

This Section describes the 802.15.4 Sequence Manager.

#### *64.5.2.4.2.1.1 Overview*

The 802.15.4 Sequence Manager (ZSM) is a hardware state machine which controls the timing of all transmit, receive, and CCA operations for the 802.15.4 Link Layer. The ZSM manages high-level 802.15.4 required timing (autosequences), and interfaces directly with a lower-level, protocol-neutral sequence manager called the Transceiver Sequence Manager (TSM), which manages the direct control of the digital, analog, and RF components of the 2.4GHz transceiver. See the TSM Chapter for more details.

For each digital and analog element in the 2.4GHz transceiver, the ZSM and TSM operate seamlessly and collaboratively, to perform some or all of the following functions:

- activation
- deactivation

- initialization
- mode transitions
- initiating calibration routines
- clock enabling and disabling
- interrupt triggering and status generation

Designed into the ZSM are a variety of complex sequences, called autosequences, which are sequential concatentations of simpler, building-block sequences. The hardware autosequences provide the following advantages:

- smaller SW memory footprint: HW automates operations previously done by SW
- increases MCU sleep times, reducing current consumption
- handles timing-critical sequences (rx-to-tx), e.g., data-polling, not possible in SW

Sequences can be initiated by software directly, or alternatively, can be initiated automatically at the expiration of a hardware timer. The general parameters of the sequences can be programmed by software prior to initiating the sequences. Such parameters allow software to select, for example:

- whether CCA is required ahead of a transmit sequence
- whether an automatic Ack frame should be transmitted after a receive sequence
- slotted vs. unslotted mode
- type of CCA (e.g., Energy Detect, CCA Mode 2)
- whether the device is a PAN Coordinator or not
- whether the device resides on 2 networks simultaneously (Dual PAN mode)

The ZSM also includes support for Dual PAN Mode, and Active Promiscuous Mode.

#### *64.5.2.4.2.1.2 Features*

- Supports complex sequences
- Unburdens software by automating many transceiver operations
- Data-polling accelerator hardware enables critical-timing sequences
- Includes Continuous CCA for low-latency-transmit applications
- Includes capability to conditionally perform CCA prior to every transmit operation
- Includes capability to conditionally transmit an Ack frame after every receive operation
- Supports software- or timer-initiated sequences
- Fully supports 2003 and 2006 versions of the 802.15.4 standard
- Supports slotted and unslotted modes
- Supports beacon-enabled and non-beacon-enabled networks
- Supports orderly transceiver shutdown on PLL unlock event
- Increases software visibility by making SEQ_STATE state readable

- Supports Dual PAN Mode, including on-the-fly channel changing
- Supports Active Promiscuous Mode

#### *64.5.2.4.2.1.3 Modes and operations*

The ZSM controls, coordinates, and automates all 802.15.4 transceiver operations within the 2.4GHz radio. Software selects the desired sequence, sets the general parameters, and (optionally) configures a hardware timer to trigger the sequence at a precise time in the future. Subsequently, the MCU can enter a low-power state, or tend to other activities, while the sequence manager state machine conducts the programmed transceiver operations. When the sequence completes, an interrupt alerts the MCU. At this point the MCU can check the completion status of the sequence, download received data from the packet buffer, and then prepare the next sequence. Sequences can be simple transceiver operations. For example, transmit one frame, or, perform a CCA on a channel. Alternatively, the sequence manager supports complex sequences, which use simple operations as building blocks, strung together in a back-to-back fashion, with precise timing intervals between operations. For example, a “TR” sequence calls for a transmit frame followed by a receive frame. For complex sequences, interrupts can optionally be generated at the completion of each stage of the sequence, providing greater visibility for the MCU into the timeline of the sequence. Software can determine the precise state of the sequence manager state machine at any time. Software can opt to abort a sequence at any time; when this happens, the sequence manager will return to its IDLE state in an orderly fashion.

#### *64.5.2.4.2.1.4 Block diagram*

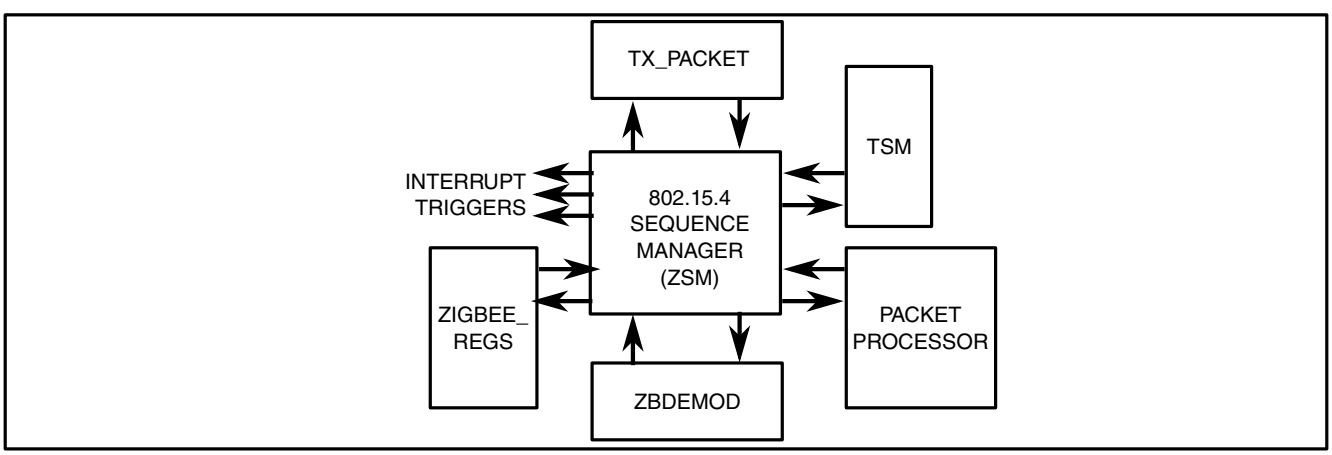



**Figure 64-148. Block diagram**

### 64.5.2.4.2.2 Memory Map and register definition

Numerous registers exist to provide configuration and status for the 802.15.4 Sequence Manager. See table below.

<table>
<tr><th>Field</th><th>R/W</th><th>Description</th></tr>
<tr><td>XCVSEQ</td><td>rw</td><td>The Transceiver Sequence Selector register selects a sequence for the sequence manager to execute. Sequence initiation can be immediate, or scheduled (see TMRTRIGEN). A write of XCVSEQ=IDLE will abort any ongoing sequence. A write of XCVSEQ=IDLE must always be performed after a sequence is complete, and before a new sequence is programmed. Any write to XCVSEQ other than XCVSEQ=IDLE during an ongoing sequence, shall be ignored. The mapping of XCVSEQ to sequence types is as follows:
<table>
<tr><th>XCVSEQ</th><th>SEQUENCE</th><th>DESCRIPTION</th></tr>
<tr><td>0</td><td>I</td><td>IDLE</td></tr>
<tr><td>1</td><td>R</td><td>RECEIVE</td></tr>
<tr><td>2</td><td>T</td><td>TRANSMIT</td></tr>
<tr><td>3</td><td>C</td><td>CCA</td></tr>
<tr><td>4</td><td>TR</td><td>TRANSMIT/RECEIVE</td></tr>
<tr><td>5</td><td>CCCA</td><td>CONTINUOUS CCA</td></tr>
</table></td></tr>
<tr><td>TMRTRIGEN</td><td>rw</td><td>Timer2 Trigger Enable.<br>1: allow timer TC2 (or TC2') to initiate a preprogrammed sequence (see XCVSEQ register).<br>0: programmed sequence initiates immediately upon write to XCVSEQ.</td></tr>
<tr><td>CCABFRTX</td><td>rw</td><td>CCA Before TX. Applies only to Sequences T and TR, ignored during all other sequences.<br>1: at least one CCA measurement is required prior to the transmit operation (see also SLOTTED).<br>0: no CCA required, transmit operation begins immediately.</td></tr>
<tr><td>SLOTTED</td><td>rw</td><td>Slotted Mode, for beacon-enabled networks. Applies only to Sequences T, TR, and R, ignored during all other sequences. Used, in concert with CCABFRTX, to determine how many CCA measurements are required prior to a transmit operation. Also used during R sequence to determine whether the ensuing transmit acknowledge frame (if any) needs to be synchronized to a backoff slot boundary.</td></tr>
<tr><td>RXACKRQD</td><td>rw</td><td>Receive Acknowledge Frame required. Applies only to Sequence TR, ignored during all other sequences.<br>1: A receive Ack frame follows the transmit frame.<br>0: An ordinary receive frame (not an Ack frame) follows the transmit frame.</td></tr>
<tr><td>AUTOACK</td><td>rw</td><td>Auto Acknowledge Enable. Applies only to Sequence R and Sequence TR, ignored during other sequences.</td></tr>
</table>

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | 1: sequence manager will follow a receive frame with an automatic hardware-generated Tx Ack frame, assuming other necessary conditions are met.<br>0: sequence manager will not follow a receive frame with a Tx Ack frame, under any conditions; the sequence will terminate after the receive frame. |
| CRC_MSK | rw | CRC Mask.<br>1: sequence manager requires CRCVALID=1 at the end of the received frame in order for the receive operation to complete successfully; if CRCVALID=0, sequence manager will return to preamble-detect mode after the last octet of the frame has been received.<br>0: sequence manager ignores CRCVALID and considers the receive operation complete after the last octet of the frame has been received. |
| PROMISCUOUS | rw | Prevents auto-TxAck on received packets<br>1: promiscuous mode, no auto-TxAck<br>0: auto-TxAck allowed, depending on other conditions. |
| CCATYPE[1:0] | rw | Clear Channel Assessment Type. Selects one of four possible functions for CCA or ED, per below. (See Chapter CCA/ED/LQI for more details).<br>00: ENERGY DETECT<br>01: CCA MODE 1<br>10: CCA MODE 2<br>11: CCA MODE 3 |
| TC3TMOUT | rw | 1: Allow a TMR3 timeout to abort any RX operation<br>0: Don't allow a TMR3 timeout to abort any RX operation<br>See Appendix A for more details on how autosequences are affected by a TMR3 timeout |
| SEQ_STATE[4:0] | r | This read-only register reflects realtime 802.15.4 Sequence Manager (ZSM) state |
| ACTIVE_PROMISCUOUS | rw | 1: Provide Data Indication on all received packets under the same rules which apply in PROMISCUOUS mode, however acknowledge those packets under rules which apply in non-PROMISCUOUS mode<br>0: normal operation (Default) |
| ACTIVE_NETWORK | rw | Selects the PAN on which to transceive, by activating a PAN parameter set (PAN0 or PAN1). In Manual Dual PAN mode (or Single PAN mode), this bit selects the active PAN parameter set (channel and addressing parameters) which governs all autosequences. In Auto Dual PAN mode, this bit selects the PAN on which to begin transceiving, latched at the point at which DUAL_PAN_DWELL register is written.<br>1: Select PAN1<br>0: Select PAN0 (Default) |
| DUAL_PAN_AUTO | rw | Activates automatic Dual PAN operating mode. In this mode, PAN-switching is controlled by hardware at a pre-programmed rate, determined by DUAL_PAN_DWELL.<br>1: Auto Dual PAN Mode |

*Table continues on the next page...*

<table>
<tr><th>Field</th><th>R/W</th><th>Description</th></tr>
<tr><td></td><td></td><td>0: Manual Dual PAN mode (or Single PAN mode). Default.<br><br>Whenever DUAL_PAN_AUTO=0, CURRENT_NETWORK=ACTIVE_NETWORK at all times. In other words, software directly controls which PAN is selected.<br><br>Whenever DUAL_PAN_AUTO=1, CURRENT_NETWORK is controlled by hardware</td></tr>
<tr><td>CURRENT_NETWORK</td><td>r</td><td>This read-only bit indicates which PAN is currently selected by hardware in automatic Dual PAN mode<br><br>1: PAN1 is selected<br><br>0: PAN0 is selected</td></tr>
<tr><td>DUAL_PAN_DWELL[7:0]</td><td>rw</td><td>Channel Frequency Dwell Time. In Auto Dual PAN mode, hardware will toggle the PAN, after dwelling on the current PAN for the interval described below (assuming Preamble/SFD not detected). A write to DUAL_PAN_DWELL, always re-initilizes the DWELL TIMER to the programmed value. If a write to DUAL_PAN_DWELL occurs during an autosequence, the DWELL TIMER will begin counting down immediately. If a write to DUAL_PAN_DWELL occurs when there is no autosequence underway, the DWELL TIMER will not begin counting until the next autosequence begins; it will begin counting at the start of the sequence warmup.<br><br>
<table>
<tr><th>PRESCALER (bits [1:0])</th><th>TIMEBASE (bits [7:2])</th><th>RANGE (min) - (max)</th></tr>
<tr><td>00</td><td>0.5ms</td><td>0.5 - 32ms</td></tr>
<tr><td>01</td><td>2.5ms</td><td>2.5 - 160ms</td></tr>
<tr><td>10</td><td>10ms</td><td>10 - 640ms</td></tr>
<tr><td>11</td><td>50ms</td><td>50ms - 3.2seconds</td></tr>
</table>
<br>A write to DUAL_PAN_DWELL also causes the value of ACTIVE_NETWORK to get latched into the hardware. This latched value will be the starting point for the automatic dual-pan mode (i.e., start on PAN0 or on PAN1). The starting value takes effect immediately (if sequence is underway and DUAL_PAN_AUTO=1), or is otherwise delayed until sequence starts and DUAL_PAN_AUTO=1.</td></tr>
<tr><td>DUAL_PAN_REMAIN[5:0]</td><td></td><td>This read-only register indicates time remaining before next PAN switch in auto Dual PAN mode. The units for this register, depend on the PRESCALER setting (bits [1:0]) in the DUAL_PAN_DWELL register, according to the following table:<br><br>
<table>
<tr><th>DUAL_PAN_DWELL PRESCALER</th><th>DUAL_PAN_REMAIN UNITS</th></tr>
<tr><td>00</td><td>0.5ms</td></tr>
<tr><td>01</td><td>2.5ms</td></tr>
<tr><td>10</td><td>10ms</td></tr>
<tr><td>11</td><td>50ms</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | The readback value indicates that between N-1 and N timebase units remain until the next PAN switch. For example, a DUAL_PAN_REMAIN readback value of 3, with a DUAL_PAN_DWELL PRESCALER setting of 2 (10ms), indicates that between 20ms (2*10ms) and 30ms (3*10ms), remain until the next automatic PAN switch. |
| RECD_ON_PAN0 | r | Indicates the packet which was just received, was received on PAN0.<br>In Dual PAN mode operating on 2 different channels, RECD_ON_PAN0 will be set if CURRENT_NETWORK=0 when the packet was received, regardless of FILTERFAIL status.<br>In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN0 will be set only if a valid packet was received on PAN0 (PAN0's FILTERFAIL_FLAG is deasserted). |
| RECD_ON_PAN1 | r | Indicates the packet which was just received, was received on PAN1.<br>In Dual PAN mode operating on 2 different channels, RECD_ON_PAN1 will be set if CURRENT_NETWORK=1 when the packet was received, regardless of FILTERFAIL status.<br>In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN1 will be set only if a valid packet was received on PAN1 (PAN1's FILTERFAIL_FLAG is deasserted). |
| ACKDELAY[5:0] | rw | Provides a fine-tune adjustment of the time delay between Rx warmdown and the beginning of Tx warmup for an Tx Acknowledge packet. ACKDELAY register will apply to both SLOTTED and UNSLOTTED TxAck, but only to TxAck (not T sequences). This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Default = 0x0.<br>Max ACKDELAY = 0x1F.<br>Min ACKDELAY = 0x2d. |
| TXDELAY[5:0] | rw | Provides a fine-tune adjustment of the time delay between post-CCA Rx warm-down and the beginning of Tx warm-up for an Tx (non-Ack) packet. TXDELAY register will apply in both SLOTTED and UNSLOTTED modes, but only to T sequences (e.g., T, TR, and T(R) ), not TxAck operations. This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Default = 0x0.<br>Max TXDELAY = 0x1F.<br>Min TXDELAY = 0x2d |
| CLR_NEW_SEQ_INHIBIT | rw | when asserted, overrides the automatic hardware locking of the programmed XCVSEQ while an autosequence is underway. Asserting this feature will allow software to change the programmed |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | autosequence “on-the-fly”, without aborting and returning to idle between sequences. Overriding the hardware lockout of XCVSEQ should be used with caution, since the Sequence Manager is not designed (or verified) for manual state transitions between one type of autosequence and other (i.e., Sequence T -> Sequence R). |
| NO_RX_RECYCLE | rw | when asserted, prevents the Sequence Manager from automatically re-starting the receiver when a packet is received which results in a FilterFail or CRC failure. Normally, the Sequence Manager jumps to the RX_CYC state, and then resumes from there with a new Rx warmup, in search of a new packet. When this bit is set, the Sequence Manager will instead return to idle state, and issue a SEQIRQ, after a FilterFail or CRC failure. |
| LATCH_PREAMBLE | rw | 1: Make PREAMBLE_DET and SFD_DET bits of PHY_STS Register “sticky”, i.e.,occurrences of preamble and SFD detection are latched and held until the start of the next autosequence<br>0: Don’t make PREAMBLE_DET and SFD_DET bits of PHY_STS Register “sticky”, i.e, these status bits reflect the realtime, dynamic state of preamble_detect and sfd_detect. (default) |
| XCVSEQ_ACTUAL[2:0] | r | indicates the programmed sequence that has been recognized by the Sequence Manager.<br>Takes into account the fact that sequence-change commands from software are ignored<br>while a sequence is underway (see new_seq_inhibit bit). Read-only bits. |
| SEQ_IDLE | r | when asserted, indicates that the Sequence Manager is in its idle state. Read-only bit. |
| NEW_SEQ_INHIBIT | r | when asserted, indicates that a new programmed autosequence has commenced (TMR2 match has occurred if TMRTRIGEN=1). Once this bit is asserted, software is blocked from commanding any new autosequences (other than Sequence I to abort the current sequence), until the current sequence completes. Hardware will ignore a sequence-change command from software while this bit is asserted. Hardware will automatically deassert this bit once the sequence completes. Read-only bit. |
| RX_TIMEOUT_PENDING | r | when asserted, indicates that a TMR3 timeout (RX timeout) flag has been set by Hardware, but the Sequence Manager has not yet aborted because an RX operation is not currently underway. This would be the case, for example, during a Sequence TR, if a TMR3 timeout were to occur during the transmit operation of this sequence; the sequence would not be aborted by Hardware until the receive operation begins. This bit will always be 0 if TC3TMOUT=0. Read-only bit. |
| RX_MODE | r | indicates that an RX operation is in progress. CCA and ED operations are considered RX operations. Read-only bit. |
| TMR2_SEQ_TRIG_ARMED | r | when asserted, indicates that TMR2 has been programmed and is armed to trigger a new autosequence, when Sequence Manager timer-triggering mode is selected (i.e., TMRTRIGEN=1). When timer-triggering mode is selected, TMR2 must be re-programmed (using either T2CMP or T2), in advance of each new sequence. Once TMR2 is programmed, this bit will be asserted, and will remain asserted until the new sequence commences (at TMR2 match). Hardware will deassert this bit when the new sequence starts. When TMRTRIGEN=0, this bit should be ignored. Read-only bit. |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| SW_ABORTED | r | when asserted, indicates that the autosequence has terminated due to an Software abort. Software can abort any programmed autosequence by writing Sequence I to XCVSEQ. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| TC3_ABORTED | r | when asserted, indicates that the autosequence has terminated due to an TC3 (TMR3) timeout during a receive operation. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| PLL_ABORTED | r | when asserted, indicates that the autosequence has terminated due to an PLL unlock event. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| CCCA_BUSY_CNT[5:0] | r | For Sequence CCCA mode only, this register indicates the number of “busy” CCA attempts which occurred during the autosequence, before the channel was detected to be idle. This register can also be read in real-time (during the autosequence) to determine how many busy CCA attempts have occurred to that point. The register saturates at 63 (i.e, if there are more than 63 busy attempts, the register will continue to read 63). This register is automatically cleared to zero by hardware when the next autosequence commences. Read-only register. |
| SEQ_T_STATUS[5:0] | r | Status of the just-completed (or ongoing) Sequence T or Sequence TR autosequence. This register is valid at all times during, and after, the Sequence T or Sequence TR. Not valid for other types of autosequences. This is a read-only register. The bits of this register map to status, according to the following table:<br>[0] 1st CCA complete (CCABFRTX=1)<br>[1] 2nd CCA complete (SLOTTED=1)<br>[2] Tx operation complete<br>[3] Rx Rec ycle occurred (Sequence TR only)<br>[4] Rx operation complete (Sequence TR only)<br>[5] TxAck operation complete(Sequence TR only) |
| CONTINUOUS_EN | rw | Enable Continuous TX or RX mode (testmode). See Appendix B. |
| SLOT_PRELOAD[7:0] | rw | This register represents the number that gets loaded into the slot_timer at SFD detect, which ultimately determines when the next slot boundary will occur. Due to processing delays within the analog front-end and digital modem, the point at which SFD is detected by the modem, is delayed relative to over-the-air timing. Since this timing may not be known prior to until actual silicon, and since this is such a critical timing parameter for slotted operations, it has been made programmable. The default value is 0x74. This timing parameter is critical for the Sequence R autosequence in slotted mode, when an automatic TxAck is required. |

### 64.5.2.4.2.3 Functional description

This section describes the supported sequences in more detail. All sequences can be initiated by software directly (instantaneously) by writing the desired sequence to the XCVSEQ register, with the TMRTRIGEN bit deasserted. Alternatively, software can

schedule the initiation of a sequence at a precise time in the future, by programming the desired start time into the T2CMP (or T2PRIMECMP) timer compare register, and setting the TMRTRIGEN bit. The XCVSEQ register field is writable and readable, so software can read this register to determine which sequence it had programmed earlier. After the sequence completes (SEQIRQ interrupt), software should set the XCVSEQ to IDLE (0x0), before initiating a new sequence. If the sequence-complete interrupt (SEQIRQ) is masked, software can determine when the sequence is complete by polling the SEQ_STATE register, until it returns to SEQ_IDLE (0x0). While a sequence is ongoing, software can abort the sequence at any time by writing IDLE (0x0) to XCVSEQ; the sequence manager state machine will respond with an orderly return to the SEQ_IDLE state, and a SEQIRQ will be generated. While a sequence is underway, software should not attempt to change XCVSEQ (other than aborting the sequence as previously described); the sequence manager will ignore all mid-sequence attempts to change XCVSEQ. As mentioned previously, software should write the IDLE value to XCVSEQ before changing sequences. If TMR2 triggering is enabled (TMRTRIGEN=1), after the scheduled sequence is automatically launched at TMR2 match, and then completes, software must re-arm TMR2 triggering by writing to either T2CMP, or T2PRIMECMP. At least the LS byte of one of these registers must be written to re-arm. This hardware interlock is designed to prevent the inadvertent re-triggering of sequences, which would otherwise occur after the 24-bit Event Timer rolls over.

The supported sequences are shown in the following table.

**Table 64-43. Supported Sequences**

| XCVSEQ | Sequence | Description |
|---|---|---|
| 0 | I | Idle |
| 1 | R | Receive Sequence<br>- conditionally followed by a TxAck |
| 2 | T | Transmit Sequence |
| 3 | C | Standalone CCA |
| 4 | TR | Transmit /Receive Sequence<br>- transmit unconditionally followed by either an R or RxAck |
| 5 | CCCA | Continuous CCA<br>- sequence completes when channel is idle |
| 6 | Reserved | |
| 7 | Reserved | |

All sequences can be aborted by a PLL unlock detection. The unlock detection is qualified by the sequence manager; in other words, an unlock detection during the early stages of the PLL warmup is expected, and is not allowed to abort the sequence at that

point. The primary rationale for the PLL unlock abort, is to prevent transmission on an incorrect, or unstable, frequency during a transmit operation, or to increase battery life by not prolonging a receive or CCA operation that is bound to fail. Software can disable the PLL unlock auto-abort. See Sequence Aborting.

#### *64.5.2.4.2.3.1 Supported Sequences*

##### *64.5.2.4.2.3.1.1 Sequence I (Idle)*

When the IDLE value is written to XCVSEQ, the sequence manager goes to its SEQ_IDLE state. If not already in SEQ_IDLE, the sequence manager executes an orderly warmdown to return to SEQ_IDLE. A SEQIRQ interrupt is then issued. Writing IDLE value to XCVSEQ is the proper way to abort a sequence.

When a sequence is aborted in this way (called a software abort), the SW_ABORTED bit of the ABORT_STS register in 802.15.4 space, will become set. This bit will be self-cleared at the start of the next autosequence.

##### *64.5.2.4.2.3.1.2 Sequence R (Receive)*

Sequence R is the basic receive sequence. Sequence R can be used to receive all types of 802.15.4 PHY- and MAC-compliant frames, including reserved frame types. However, reception of Acknowledge frames is not recommended using Sequence R. This is because reception of an Acknowledge frame, usually follows a transmitted frame, with a designated Sequence Number, so that the received Acknowledge frame can be verified for the matching Sequence Number. In a standalone Sequence R, there is no Sequence Number to verify against, so any Acknowledge frame is merely transferred to the Packet Buffer. (Note: the appropriate way to receive Acknowledge frames is with Sequence TR, with the RXACKRQD bit asserted). Using Sequence R, *all* frames which pass frame-filtering rules and CRC check, are transferred to the Packet Buffer.

The basic execution of a successful Sequence R is as follows:

- MCU writes TC2 compare value to the desired initiation time, and sets TMRTRIGEN (optional)
- MCU writes XCVSEQ=R
- Wait for TC2 match (if TMRTRIGEN=1)
- SEQ_MGR executes RxWarmup
- Preamble Detected
- SFD Detected
- FrameLength octet received, transferred to Packet Buffer
- *N* additional octets received (*N* = FrameLength), transferred to Packet Buffer. *Note if (FrameLength > 127), then N = 127.*
- If frame-filtering rules and CRC check pass:

- ---> RXIRQ interrupt issued
- ---> SEQ_MGR executes RxWarmdown
- ---> If received Frame Control Field indicates AckRequest=1
- ---- ---> SEQ_MGR executes TxWarmup
- ---- ---> an Ack frame is transmitted using the received Sequence Number
- ---- ---> SEQ_MGR executes TxWarmdown
- SEQIRQ interrupt issued

When Sequence R is initiated, the ZSM sequence manager warms up the receiver (analog and digital elements), via the TSM (Transceiver Sequence Manager). After warmup is complete, timer TMR3 can be enabled as a “bracketing” timer (software option). If a TMR3 timer match occurs, and software has asserted the TC3TMOUT bit, the sequence manager will warm down the receiver and return to SEQ_IDLE state. Assuming no TMR3 timeout, reception proceeds in preamble-search mode. Once a preamble is detected, reception continues in SFD-search mode. At this point, the sequence manager’s slot timer is activated. This is because a slotted “transmit acknowledge” frame (TxAck) may be required later in this sequence. The slot timer is loaded with the following value (in us):

Slot Timer Preload = 160 + TXWARMUPTIME + (2*ACKDELAY).

The 160 term arises because, at the point of SFD detection, we are exactly 10 symbols into the backoff slot, and the symbol period is 16us. The slot timer will count up and eventually rollover at 319, to 0. (There are 320us in a backoff slot). The receiver receives the next octet (FrameLength). Starting with this octet, and continuing through the remainder of the frame, each octet that is received is transferred to Packet Buffer. After each octet is transferred to Packet Buffer, the Packet Buffer address is incremented by 1. The next 2 octets (Frame Control Field) are then received, which are parsed to obtain the AckRequest field, and the FrameVersion field. The next octet is then received (SequenceNumber), and stored. The remainder of the frame, determined by FrameLength, is then received. As the octets are received, the packet data is parsed and subjected to packet-filtering rules (see Chapter Packet Processor). If any of the packet-filtering rules fail, or if the CRC check on the FCS field fails, the sequence manager will return the receiver to preamble-search mode, after the last octet has been received and transferred to Packet Buffer. Assuming the packet-filtering rules and CRC pass, an interrupt (RXIRQ) is issued to the MCU. At this point, the sequence manager must determine if a TxAck is required. A TxAck is required if the following 3 conditions are all met:

- AUTOACK=1 (register bit)
- PROMISCUOUS=0 (register bit)
- The received FrameControlField has AckRequest=1

If these conditions are met, and SLOTTED mode is not in effect, the sequence manager loads its TxAck timer with the following value (in us):

TxAck Timer Preload = 192 - TXWARMUPTIME - 2*ACKDELAY

The 192us is the non-slotted RX-to-TX turnaround requirement in the 802.15.4 standard. The TXWARMUPTIME takes into account the fact that the TxAck timer must expire early for the Tx warmup to be complete at exactly the 192us point. The ACKDELAY is a signed, fine-tune adjustment to the TxAck transmission time (+/- 62us).

If SLOTTED mode is in effect, the slot timer was previously loaded (at SFD detect), but the sequence manager needs to determine how much time is left in the current backoff slot. If less than 192us remain, the sequence manager must wait an *additional* backoff slot before initiating the Ack transmission. So it sets a flag indicating it must allow the slot counter to rollover an additional iteration.

Finally, at TxAck timer expiration (in non-SLOTTED mode), or at the qualified rollover of the slot timer (in SLOTTED mode), a Tx warmup is initiated. The conclusion of the Tx warmup will coincide precisely with a backoff slot boundary if SLOTTED mode is in effect. The sequence manager will then initiate the transmission of the Ack frame. The entire Ack frame will be built up by hardware; the Packet Buffer is not used for the auto-TxAck feature. The hardware will insert the following parameters into the Ack frame's Frame Control Field:

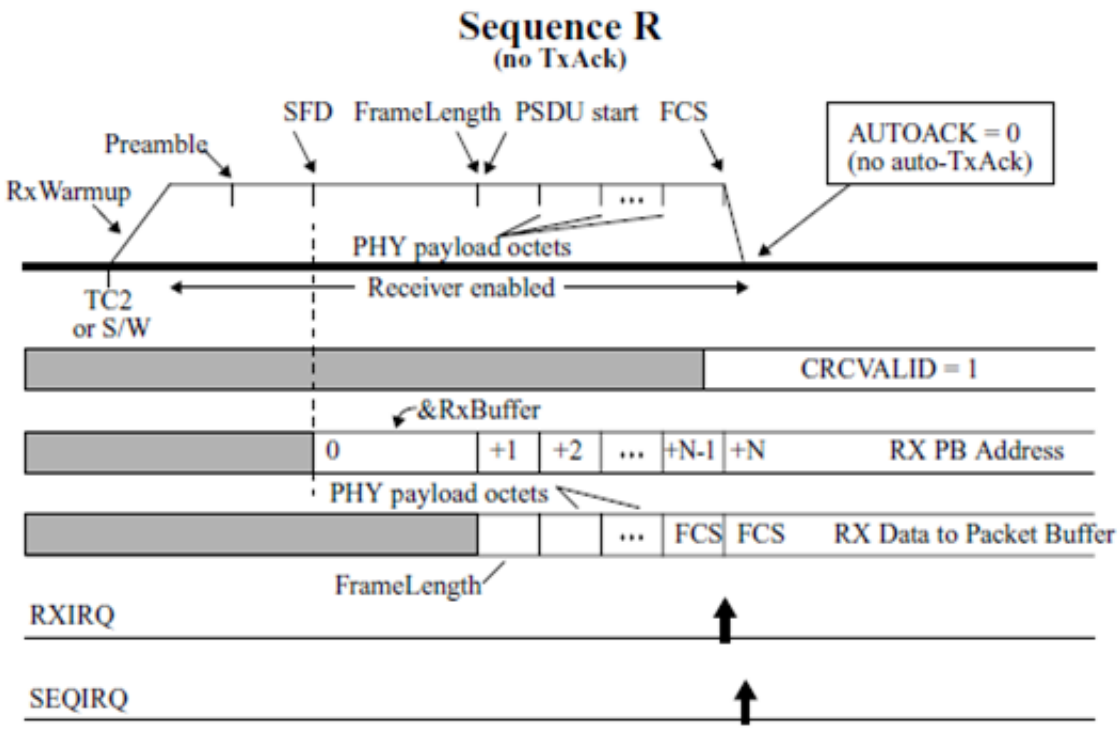
**Table 64-44. Frame Control Field for Hardware-generated Auto-TxAck Frame**

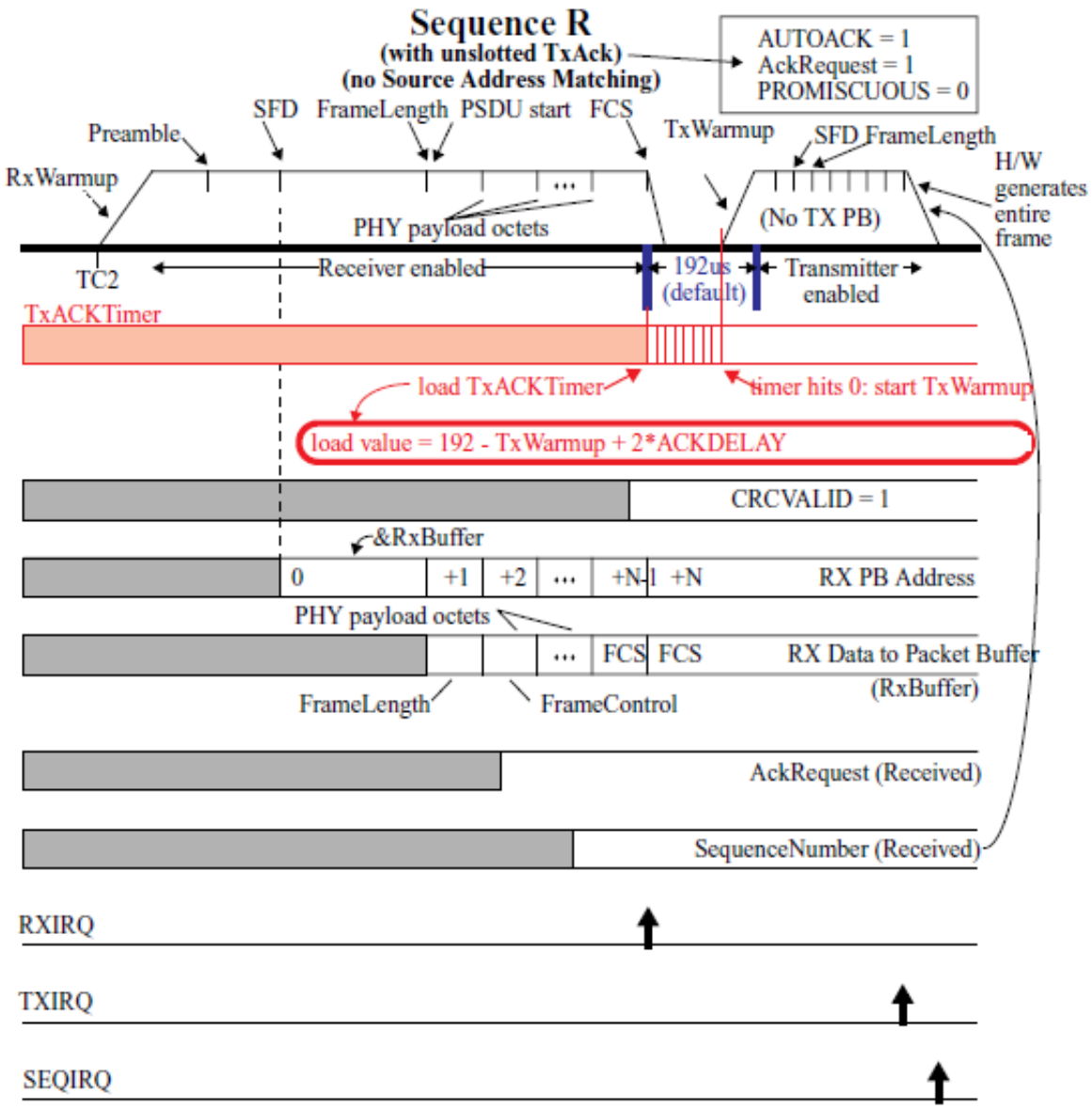
| FCF Field | Value |
|---|---|
| FrameType | Acknowledge |
| SecurityEnabled | 0 |
| FramePending | Determined by the Packet Processor.<br>(see Packet Processor Chapter, Source Address Management.) |
| AckRequest | 0 |
| PanIDCompression | 0 |
| Reserved (bits 7-9) | 000 |
| DstAddrMode | 0 |
| FrameVersion | 0 |
| SrcAddrMode | 0 |

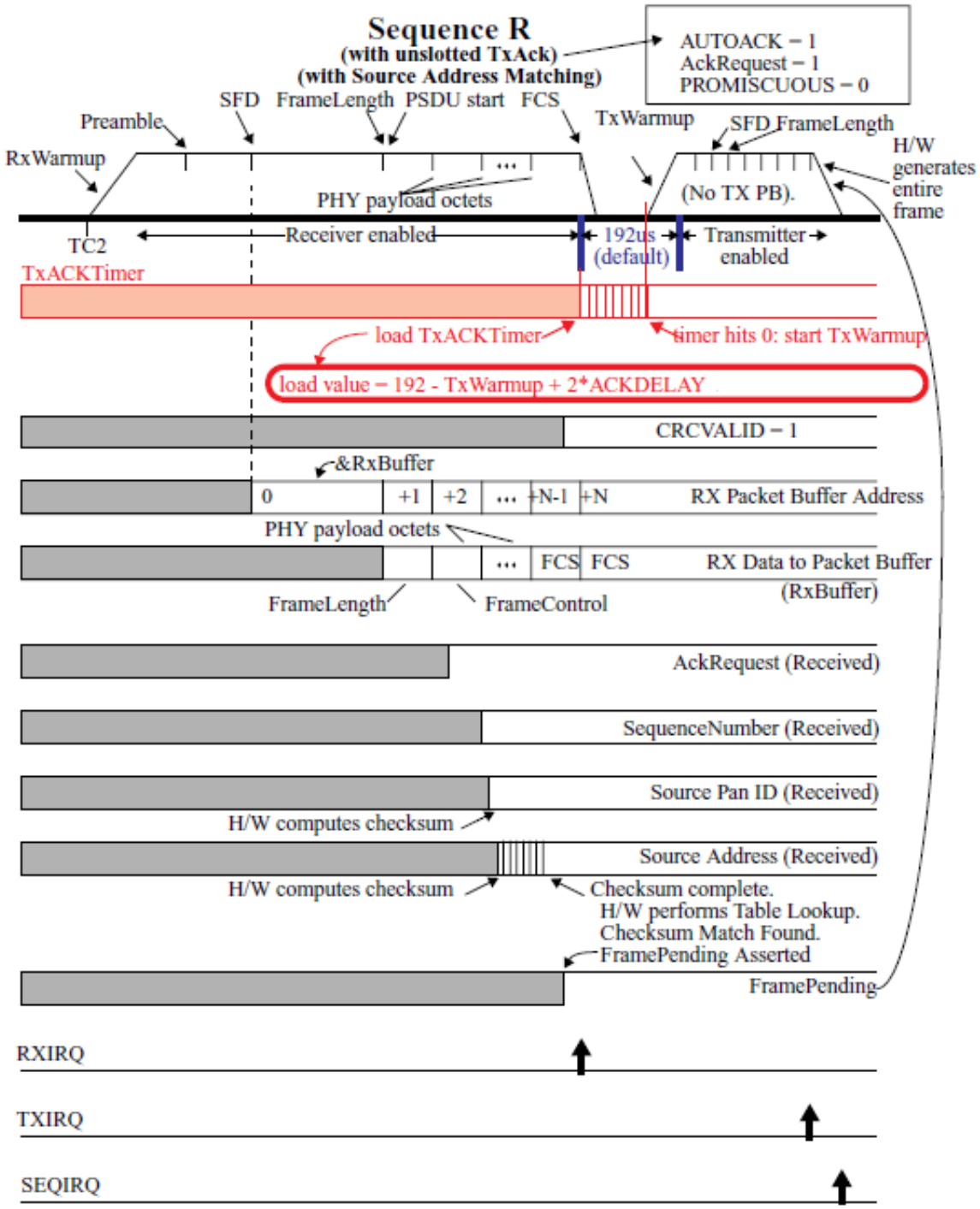
The Sequence Number for the Ack packet, will be the Sequence Number obtained from the previously received frame.

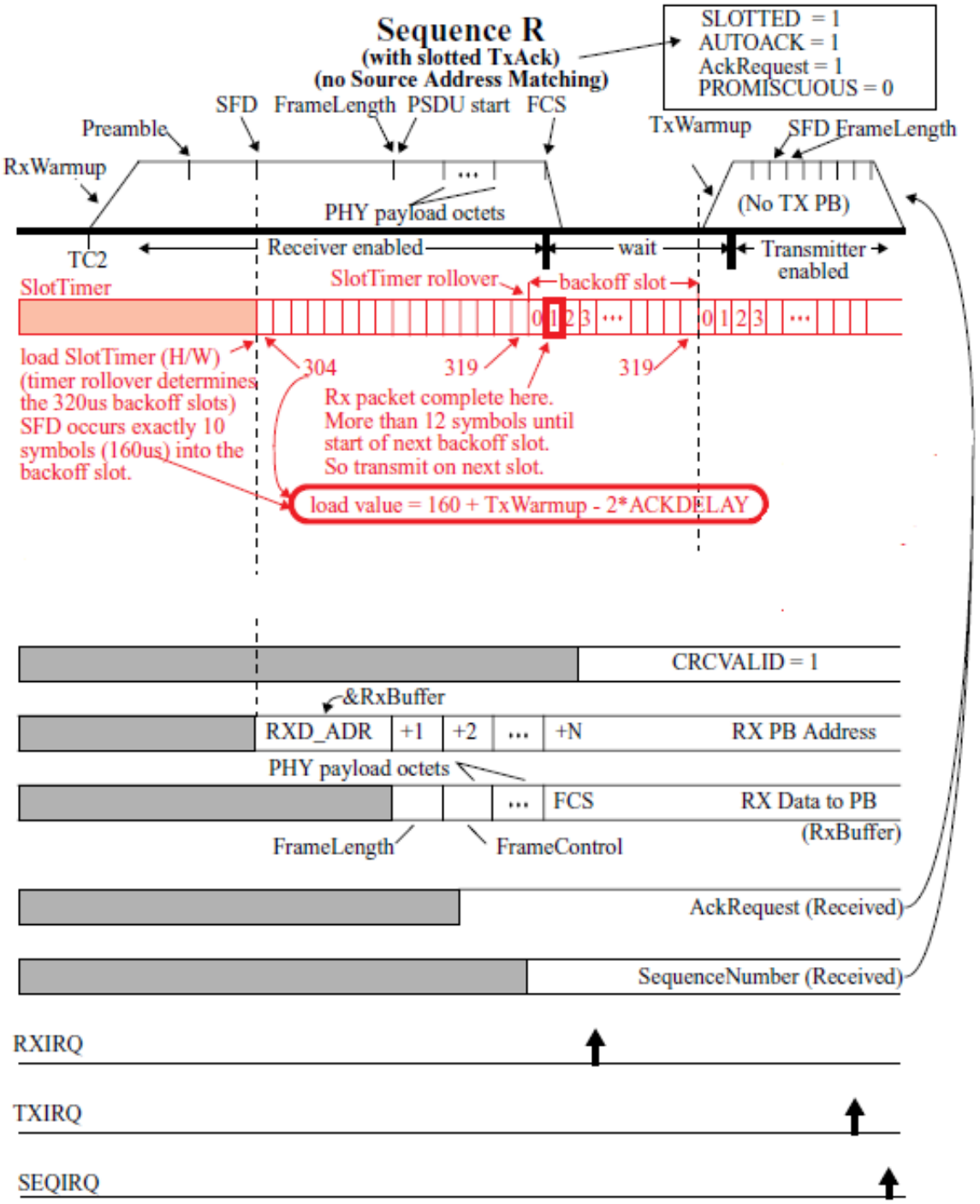
After the Ack frame has been transmitted, the sequence manager will issue a TXIRQ interrupt, and then perform a Tx warmdown. Finally, a SEQIRQ interrupt will be issued, and the sequence manager returns to SEQ_IDLE state.

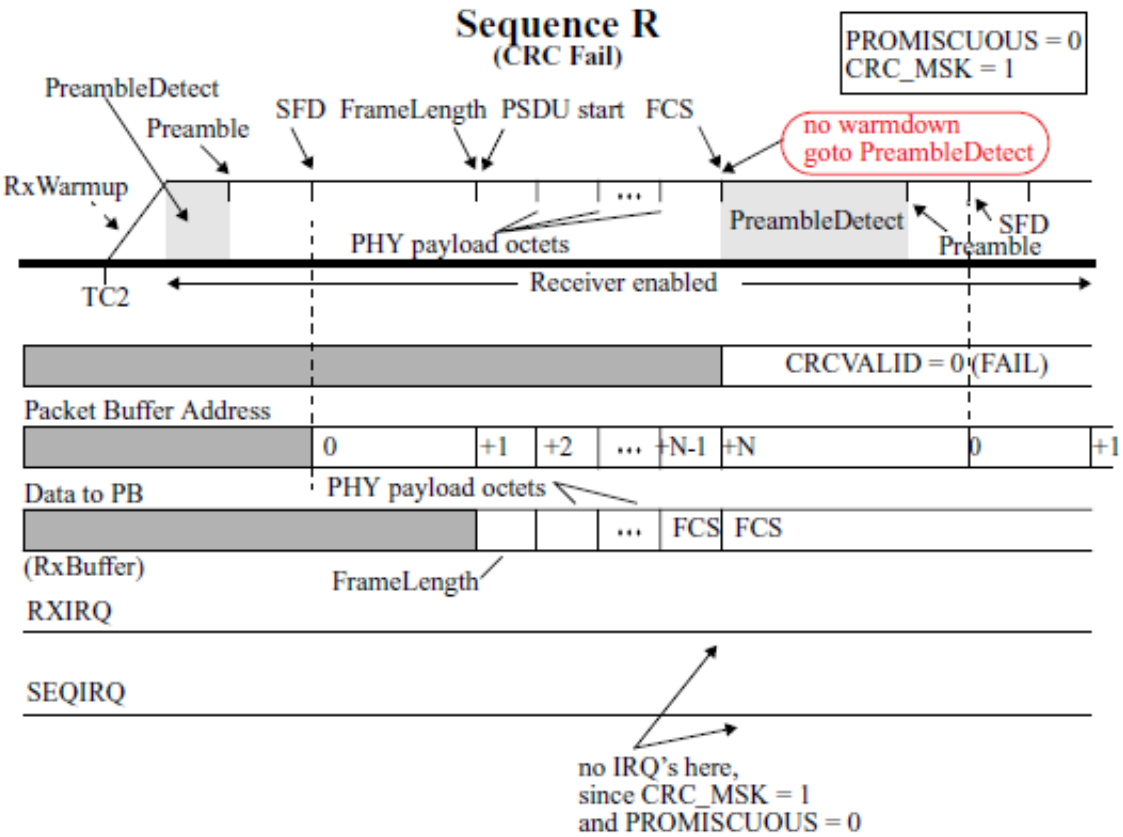
## Sequence R Timing Diagrams

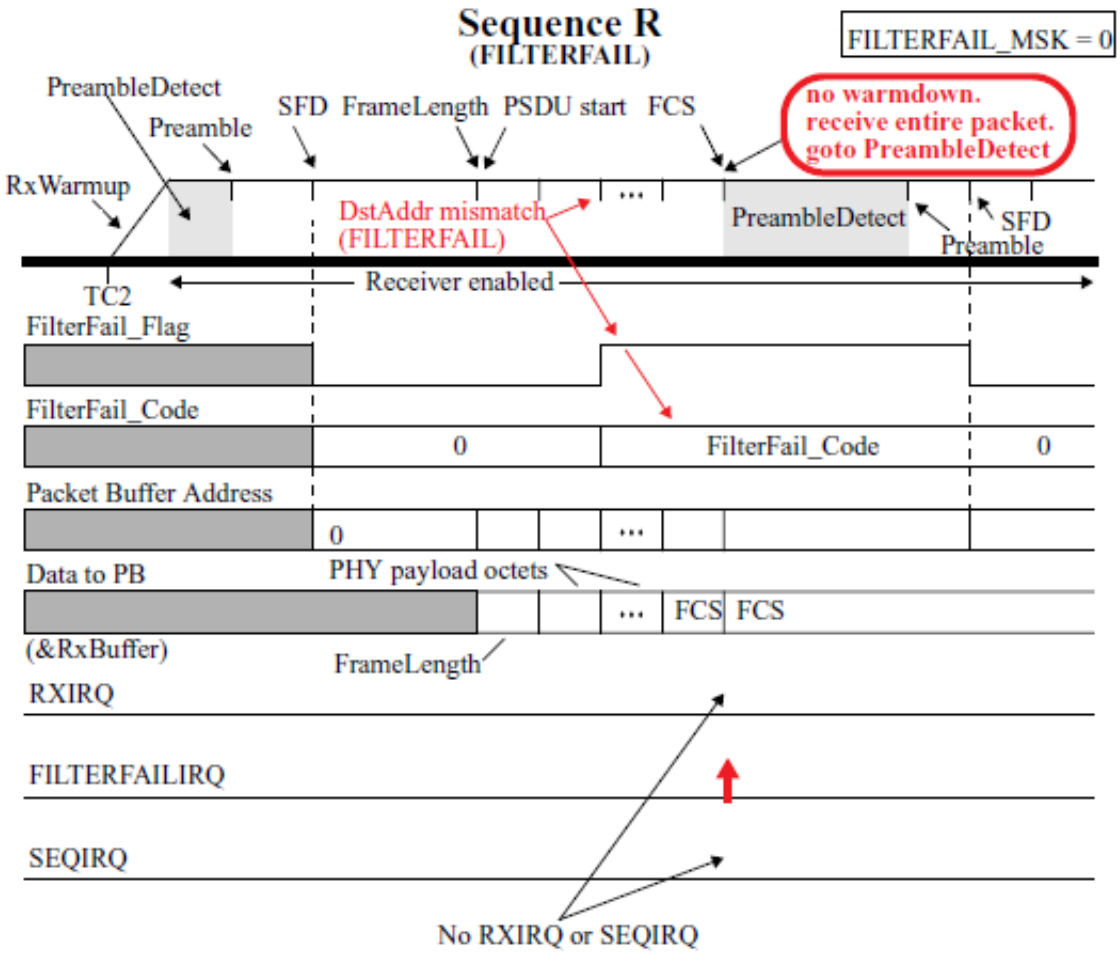



Sequence R
(with unslotted TxAck)
(no Source Address Matching)
AUTOACK = 1
AckRequest = 1
PROMISCUOUS = 0
SFD
FrameLength
PSDU start
FCS
Preamble
RxWarmup
TxWarmup
SFD FrameLength
H/W generates entire frame
PHY payload octets
(No TX PB)
TC2
Receiver enabled
192us (default)
Transmitter enabled
TxACKTimer
load TxACKTimer
timer hits 0: start TxWarmup
load value = 192 - TxWarmup + 2*ACKDELAY
CRCVALID = 1
&RxBuffer
0
+1
+2
+N-1
+N
RX PB Address
PHY payload octets
FCS
FCS
RX Data to Packet Buffer (RxBuffer)
FrameLength
FrameControl
AckRequest (Received)
SequenceNumber (Received)
RXIRQ
TXIRQ
SEQIRQ


Sequence R
(with unslotted TxAck)
(with Source Address Matching)
AUTOACK = 1
AckRequest = 1
PROMISCUOUS = 0
Preamble
SFD
FrameLength
PSDU start
FCS
TxWarmup
SFD FrameLength
H/W generates entire frame
RxWarmup
PHY payload octets
(No TX PB).
TC2
Receiver enabled
192us (default)
Transmitter enabled
TxACKTimer
load TxACKTimer
timer hits 0: start TxWarmup
load value = 192 - TxWarmup + 2*ACKDELAY
CRCVALID = 1
&RxBuffer
0
+1
+2
+N-1
+N
RX Packet Buffer Address
PHY payload octets
FCS
FCS
RX Data to Packet Buffer (RxBuffer)
FrameLength
FrameControl
AckRequest (Received)
SequenceNumber (Received)
Source Pan ID (Received)
H/W computes checksum
Source Address (Received)
H/W computes checksum
Checksum complete.
H/W performs Table Lookup.
Checksum Match Found.
FramePending Asserted
FramePending
RXIRQ
TXIRQ
SEQIRQ


Sequence R
(with slotted TxAck)
(no Source Address Matching)
SLOTTED = 1
AUTOACK = 1
AckRequest = 1
PROMISCUOUS = 0
Preamble
SFD
FrameLength
PSDU start
FCS
TxWarmup
SFD FrameLength
RxWarmup
PHY payload octets
(No TX PB)
TC2
Receiver enabled
wait
Transmitter enabled
SlotTimer
SlotTimer rollover
backoff slot
load SlotTimer (H/W)
(timer rollover determines the 320us backoff slots)
SFD occurs exactly 10 symbols (160us) into the backoff slot.
304
319
319
Rx packet complete here.
More than 12 symbols until start of next backoff slot.
So transmit on next slot.
load value = 160 + TxWarmup - 2*ACKDELAY
CRCVALID = 1
&RxBuffer
RXD_ADR
+1
+2
+N
RX PB Address
PHY payload octets
FCS
RX Data to PB
(RxBuffer)
FrameLength
FrameControl
AckRequest (Received)
SequenceNumber (Received)
RXIRQ
TXIRQ
SEQIRQ


Sequence R
(CRC Fail)
PROMISCUOUS = 0
CRC_MSK = 1
PreambleDetect
Preamble
SFD
FrameLength
PSDU start
FCS
no warmdown
goto PreambleDetect
RxWarmup
PHY payload octets
PreambleDetect
SFD
Preamble
TC2
Receiver enabled
CRCVALID = 0 (FAIL)
Packet Buffer Address
0
+1
+2
+N-1
+N
0
+1
Data to PB
PHY payload octets
FCS
FCS
(RxBuffer)
FrameLength
RXIRQ
SEQIRQ
no IRQ's here,
since CRC_MSK = 1
and PROMISCUOUS = 0




#### *64.5.2.4.2.3.1.3 Sequence T (Transmit)*

Sequence T is the basic, standalone transmit sequence. Sequence T can be used to transmit all types of 802.15.4 PHY- and MAC-compliant frames. However, transmission of Acknowledge frames is not recommended using Sequence T. This is because transmission of an Acknowledge frame, usually follows a received frame, with a designated Sequence Number. The ZSM sequence manager automates the transmission of an Ack frame which follows a received frame, using Sequence R with the AUTOACK bit asserted. This is the recommended method for transmitting an Ack frame. This is especially beneficial, because the transmitted Ack frame octets are generated by hardware, and so no setup of a transmit packet in Packet Buffer, is required. However, sequence manager hardware does not prohibit Ack frames using Sequence T.

Sequence T allows for the insertion of 1 or 2 CCA (clear channel assessment) measurements prior to transmission, to ensure that the selected channel is idle. The CCA-before-TX feature is governed by two register bits, CCABFRTX and SLOTTED. All CCA measurements must indicate channel-idle, in order for the sequence manager to proceed to transmission; if the channel is determined by CCA to be busy, the sequence

manager will terminate the sequence without a transmission. The number of CCA measurements attempted by the sequence manager, and the timing of the measurements, is shown in the following table.

**Table 64-45. Timing of CCA Operations for Sequence T**

| CCABFRTX | SLOTTED | Number of CCA measurements | Timing of Initiation of CCA measurement #1 | Timing of Initiation of CCA measurement #2 | Timing of First Bit of Transmitted Frame (assumes channel idle) |
|---|---|---|---|---|---|
| 0 | X | 0 | - | - | Immediately follows TxWarmup |
| 1 | 0 | 1 | Immediately follows RxWarmup | - | Immediately follows RxWarmdown and TxWarmup |
| 1 | 1 | 2 | Immediately follows RxWarmup | 320us after Initiation of CCA measurement #1 | 320us after Initiation of CCA measurement #2 |

Any CCA mode can be used for Sequence T and Sequence TR (CCA Modes 1, 2, and 3 are available. See Chapter CCA/ED/LQI). Software must configure the CCATYPE bits prior to initiating the autosequence, to select the CCA Mode.

The basic execution of a successful Sequence T is as follows:

- MCU writes TC2 compare value to the desired initiation time, and sets TMRTRIGEN (optional)
- MCU sets CCABFRTX if one or more CCA's are required prior to transmit
- MCU sets SLOTTED if in slotted mode (2 CCA's will be attempted)
- MCU writes XCVSEQ=T
- Wait for TC2 match (if TMRTRIGEN=1)
- If CCABFRTX=1 :
- ---> SEQ_MGR executes RxWarmup
- ---> SEQ_MGR initiates CCA measurement (takes 128us)
- ---> if CCA indicates channel busy, sequence terminates, CCAIRQ & SEQIRQ issued
- ---> otherwise, if CCA indicates channel idle:
- ---- ---> if SLOTTED=0, SEQ_MGR executes a RxWarmdown, followed by a TxWarmup
- ---- ---> if SLOTTED=1, SEQ_MGR initiates a 2nd CCA, 320us after initiating 1st CCA.
- ---- ---> if SLOTTED=1, and channel busy, sequence terminates, CCAIRQ & SEQIRQ issued

- ---- ---> if SLOTTED=1, and channel idle, SEQ_MGR executes a RxWarmdown, followed by a TxWarmup
- if SLOTTED=1, SEQ_MGR waits until 320us elapse after 2nd CCA initiation
- SEQ_MGR executes TxWarmup
- Preamble transmitted
- SFD transmitted
- FrameLength octet is obtained from Packet Buffer Address 0, and transmitted
- CRC engine is reset
- *N* additional octets are obtained from Packet Buffer and transmitted (*N* = FrameLength - 2)
- Each octet that is transmitted, is shifted through the CRC engine
- After *N*th octet transmitted, FCS is available from CRC engine.
- FCS lower octet transmitted
- FCS upper octet transmitted
- SEQ_MGR issues TXIRQ
- SEQ_MGR executes TxWarmdown
- SEQ_MGR issues SEQIRQ

When Sequence T is initiated, the sequence manager first must determine if one or more CCA measurements are required. If CCABFRTX bit is asserted, the sequence manager warms up the receiver (analog and digital elements), via the TSM (Transceiver Sequence Manager). Immediately after the conclusion of the warmup, a CCA measurement is initiated. The CCA measurement takes 128us. If CCA indicates the channel is busy, the sequence manager warms down the receiver, issues CCAIRQ and SEQIRQ interrupts, and terminates the sequence. If, however, CCA indicates the channel is idle, the sequence manager inspects the SLOTTED bit to determine what to do next. If the SLOTTED bit is clear, the sequence manager issues a CCAIRQ interrupt, and then executes a warmdown of the receiver. If, however, the SLOTTED bit is set, the sequence manager does not issue a CCAIRQ interrupt; instead, the sequence manager initiates a second CCA measurement, precisely 320us after the initiation of the first CCA. This 320us interval is timed by the slot timer, which had been preloaded with 320 (microseconds) at the instant that Rx warmup was complete. At the conclusion of the second CCA, the sequence manager issues a CCAIRQ. If the second CCA indicates the channel is busy, the sequence manager warms down the receiver, issues a SEQIRQ interrupt, and terminates the sequence. If, however, CCA indicates the channel is idle, the sequence manager executes a warmdown of the receiver. Prior to executing the warmdown, the sequence manager reloads the slot timer, in order to precisely schedule the ensuing transmission. This time, the slot timer is loaded with a smaller value, in order to account for the fact that the timer must expire early in order to trigger the Tx warmup, which must complete at precisely the next backoff slot boundary. Thus, this time the slot timer is preloaded with:

Slot Timer Preload = 320 - TXWARMUPTIME + (2*TXDELAY).

The 320us is the duration of the backoff slot. The TXWARMUPTIME takes into account the fact that the TxAck timer must expire early for the Tx warmup to be complete at exactly the 320us point. The TXDELAY is a signed, fine-tune adjustment to the TxAck transmission time (+/- 62us).

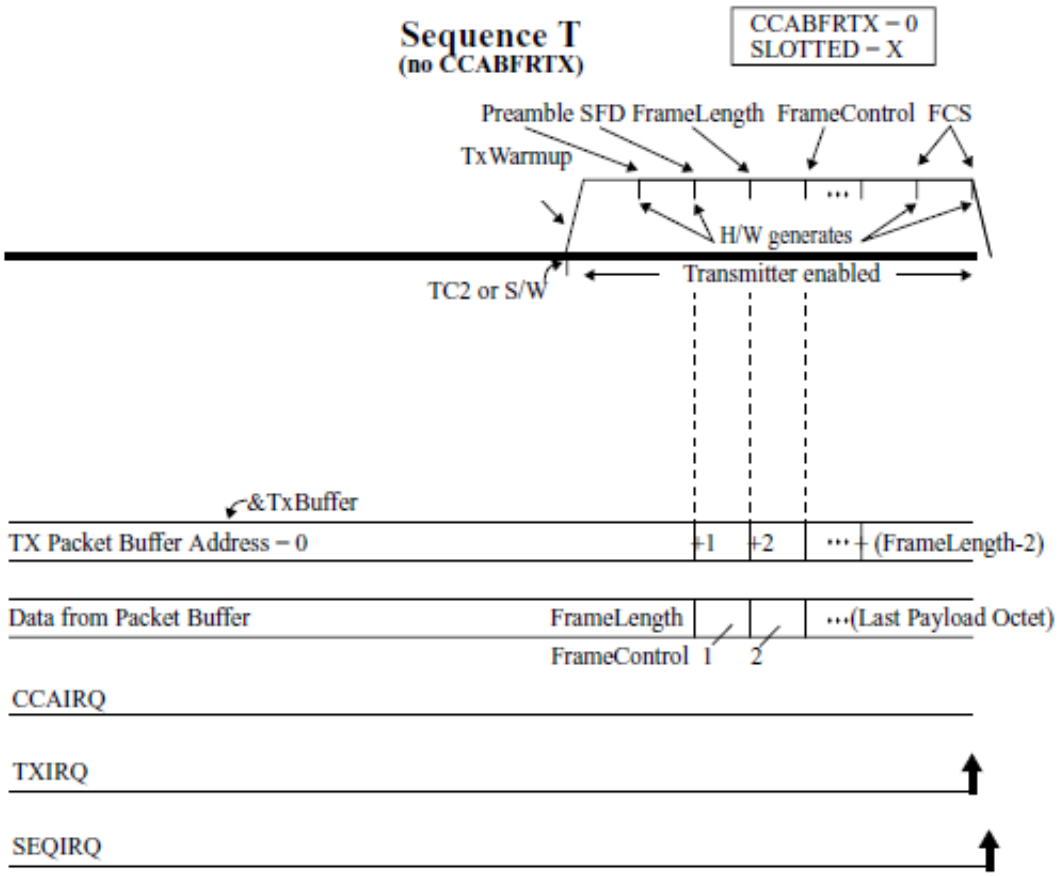
If the SLOTTED bit is asserted, once the slot timer expires, the sequence manager executes a Tx warmup. The conclusion of this warmup coincides with the backoff slot boundary. If the SLOTTED bit is deasserted, the sequence manager does not wait for the slot timer, and instead executes a Tx warmup immediately after the Rx warmdown.

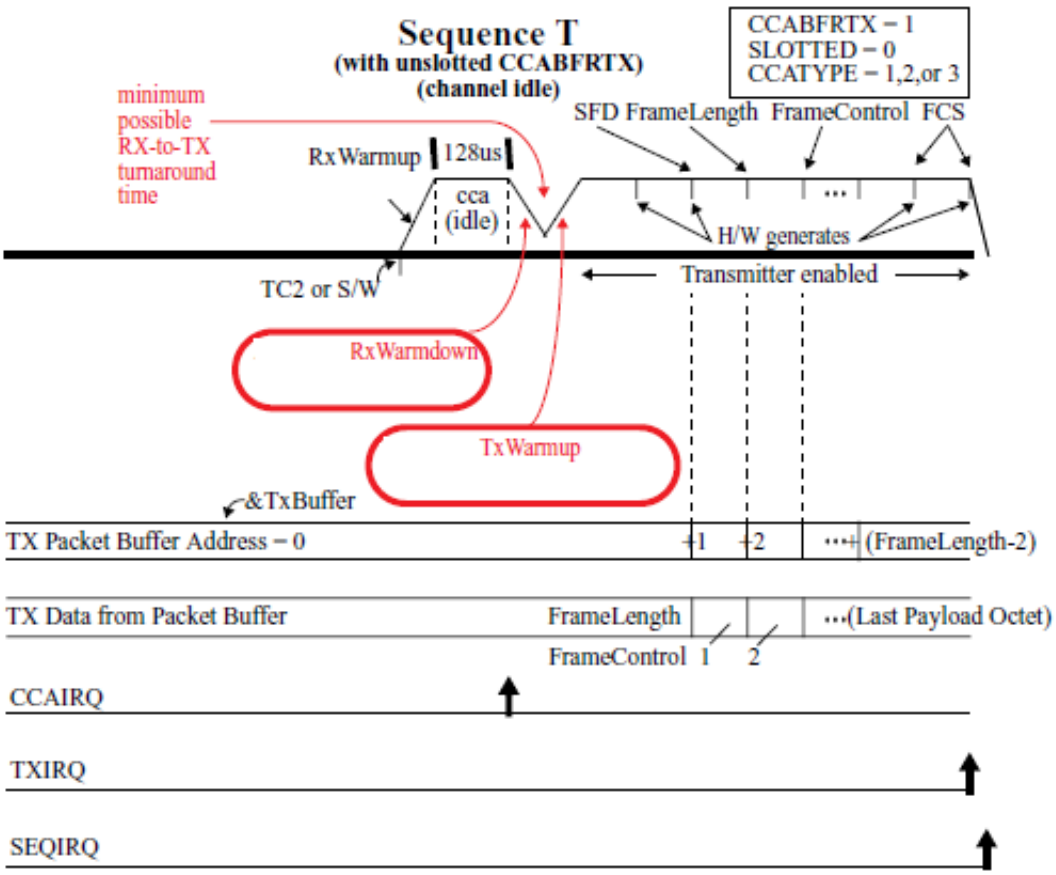
At this point, the sequence manager begins the transmit operation. The preamble and SFD are transmitted first. The preamble and SFD octets are generated by hardware, and are not included in the Packet Buffer. Then, the FrameLength octet is obtained from the Packet Buffer (address 0), and the buffer address pointer is incremented by one. The FrameLength octet is transmitted. The CRC engine is reset at this point. The FrameLength octet is used to determine how many more octets remain in the Packet Buffer. The number of remaining octets is:

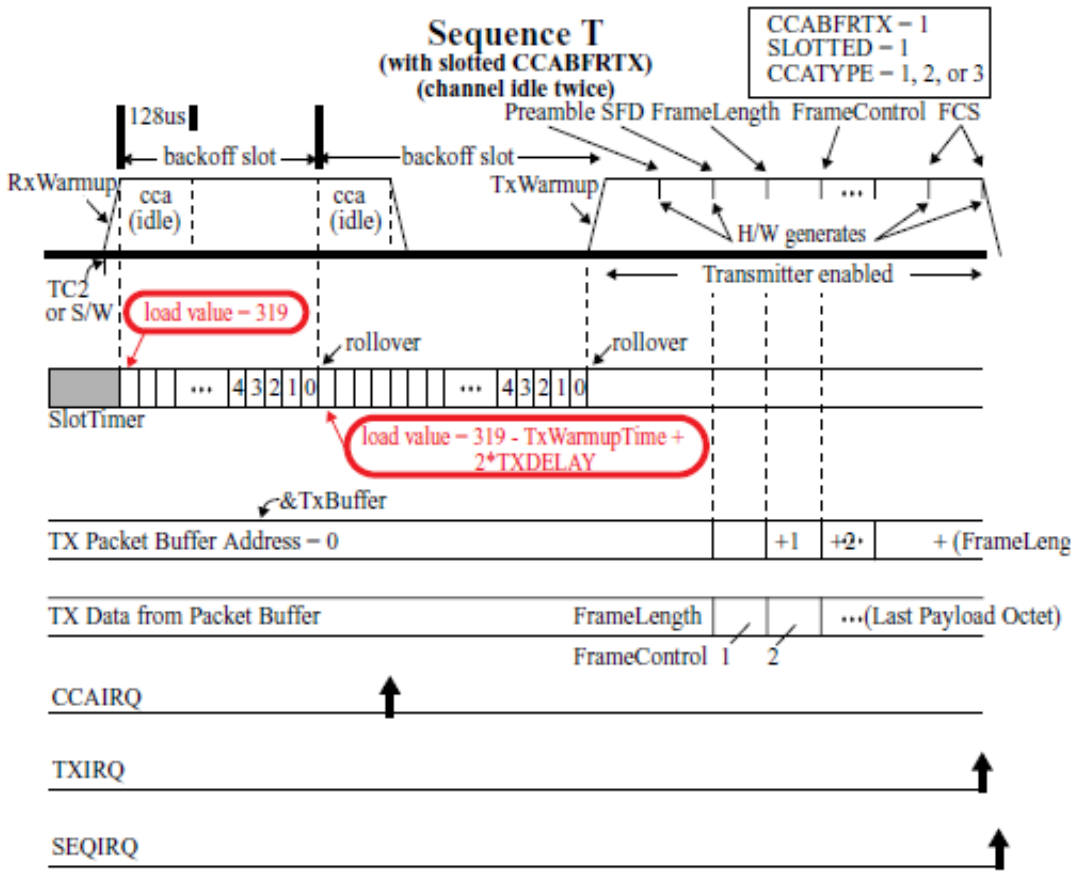
Number of octets remaining in Tx buffer = FrameLength - 2

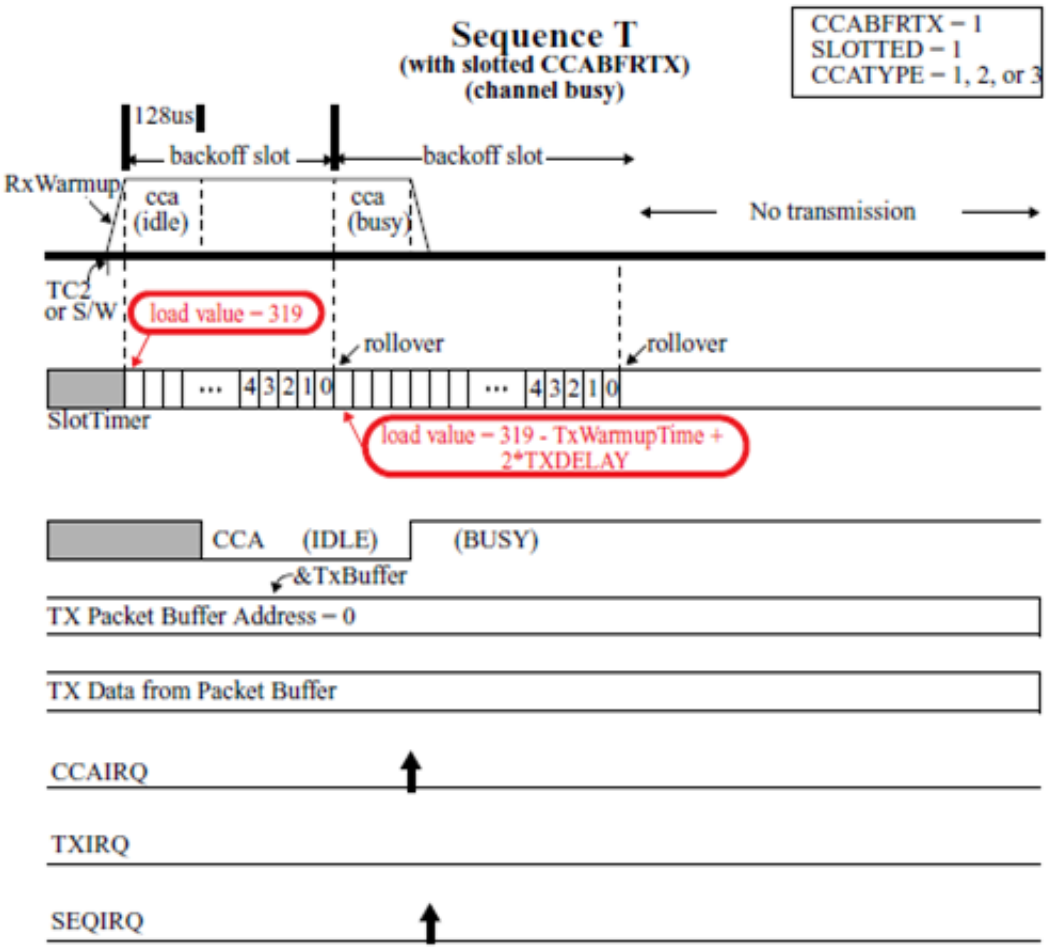
This is because the final 2 octets of the transmit frame constitute the FCS (Frame Check Sequence), and this value is hardware-computed. Thus, FCS is not part of the Tx buffer. Each of the octets in the Tx buffer is transmitted, and simultaneously shifted through CRC, to generate the FCS field. Once the Tx buffer is drained and all of its octets transmitted, the 2 FCS octets are transmitted, starting with the least significant octet. At this point, the transmit operation is complete. The sequence manager issues a TXIRQ interrupt. The sequence manager then executes a complete Tx warmdown, and follows up with a SEQIRQ interrupt. The sequence manager returns to SEQ_IDLE state.

## Sequence T Timing Diagrams


Sequence T
(no CCABFRTX)
CCABFRTX = 0
SLOTTED = X
Preamble SFD FrameLength FrameControl FCS
TxWarmup
H/W generates
TC2 or S/W
Transmitter enabled
&TxBuffer
TX Packet Buffer Address = 0
+1
+2
(FrameLength-2)
Data from Packet Buffer
FrameLength
(Last Payload Octet)
FrameControl 1
2
CCAIRQ
TXIRQ
SEQIRQ


Sequence T
(with unslotted CCABFRTX)
(channel idle)
CCABFRTX = 1
SLOTTED = 0
CCATYPE = 1,2,or 3
minimum possible RX-to-TX turnaround time
RxWarmup
128us
cca (idle)
SFD FrameLength FrameControl FCS
H/W generates
Transmitter enabled
TC2 or S/W
RxWarmdown
TxWarmup
&TxBuffer
TX Packet Buffer Address = 0
+1
+2
(FrameLength-2)
TX Data from Packet Buffer
FrameLength
(Last Payload Octet)
FrameControl 1
2
CCAIRQ
TXIRQ
SEQIRQ


Sequence T
(with slotted CCABFRTX)
(channel idle twice)
CCABFRTX = 1
SLOTTED = 1
CCATYPE = 1, 2, or 3
128us
backoff slot
backoff slot
Preamble SFD FrameLength FrameControl FCS
RxWarmup
cca
(idle)
cca
(idle)
TxWarmup
H/W generates
Transmitter enabled
TC2
or S/W
load value = 319
rollover
rollover
4 3 2 1 0
4 3 2 1 0
SlotTimer
load value = 319 - TxWarmupTime +
2*TXDELAY
&TxBuffer
TX Packet Buffer Address = 0
+1
+2
+ (FrameLeng
TX Data from Packet Buffer
FrameLength
(Last Payload Octet)
FrameControl 1
2
CCAIRQ
TXIRQ
SEQIRQ




#### *64.5.2.4.2.3.1.4 Sequence C (CCA)*

Sequence C is the standalone CCA sequence. During Sequence C, the sequence manager executes a Clear Channel Assessment (CCA). The result of the CCA measurement is reported back to software in the CCA bit of the IRQSTS2 Register. The CCA bit indicates either a busy channel (1), or an idle channel (0).

The basic execution of a Sequence C is as follows:

- MCU writes TC2 compare value to the desired initiation time, and sets TMRTRIGEN (optional)
- MCU writes XCVSEQ=C
- Wait for TC2 match (if TMRTRIGEN=1)
- SEQ_MGR asserts one of {**cca1_en, cca2_en, cca3_en, ed_en** } to CCA_DIG
- SEQ_MGR executes RxWarmup, via Transceiver Sequence Manager
- SEQ_MGR initiates CCA measurement by asserting **rx_cca_en** to CCA_DIG
- SEQ_MGR waits for CCA to complete

- CCA bit is set to the CCA status (1=busy 0=idle)
- CCAIRQ interrupt issued
- SEQ_MGR executes RxWarmdown
- SEQIRQ interrupt issued

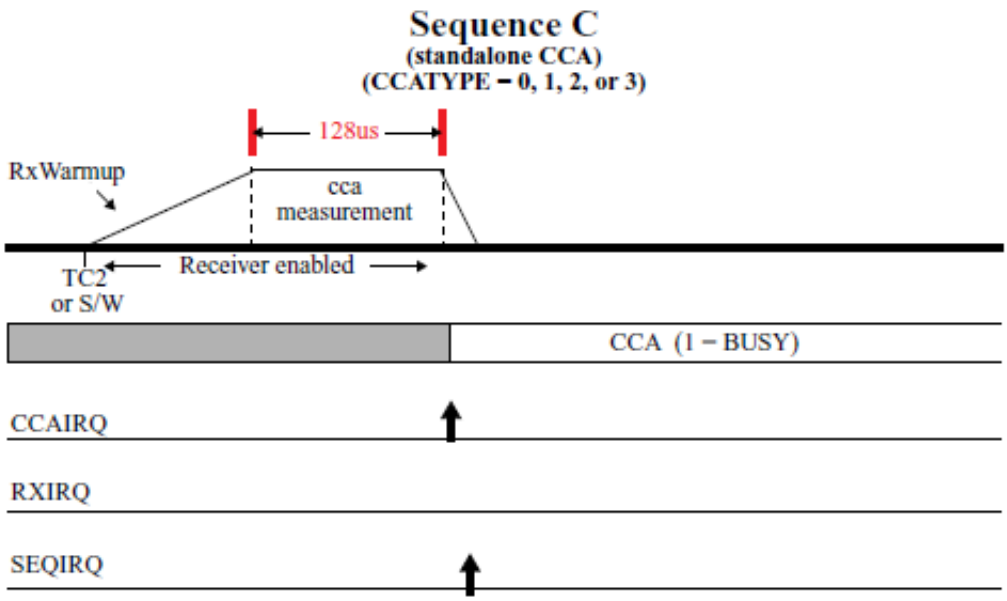
When Sequence C is initiated, the sequence manager asserts one of 4 control signals to CCA_DIG, based on which CCA or ED mode is programmed into CCATYPE[1:0]. The mapping of CCATYPE[1:0] to control signal is shown in the following table:

| CCATYPE[1:0] | CCA_DIG CONTROL SIGNAL |
|---|---|
| 00 | ed_en |
| 01 | cca1_en |
| 10 | cca2_en |
| 11 | cca3_en |

The asserted control signal will remain asserted for the duration of the Sequence C. The sequence manager warms up the receiver (analog and digital elements), via the Transceiver Sequence Manager (TSM). The TC3 “bracketing” timer has no effect on the sequence manager during Sequence C, so it cannot abort the sequence. At the competion of the warmup, the sequence manager intiates the CCA by asserting the **rx_cca_en** signal to CCA_DIG. While CCA is enabled, the CCA_DIG measures and averages the energy or signal on the channel. At the completion of the CCA process, the CCA bit is set to the status of the channel (idle or busy). The CCAIRQ is issued, the receiver is warmed down, the SEQIRQ is issued, and the sequence manager returns to SEQ_IDLE state.

Any CCATYPE setting may be used for Sequence C.

## Sequence C Timing Diagrams




#### *64.5.2.4.2.3.1.5 Sequence TR (Transmit/Receive)*

Sequence TR is a combination Transmit/Receive sequence. The sequence is executed as a concatenation of 1 transmit operation followed by 1 receive operation, with a minimum TX-to-RX turnaround time in between. There are 2 permutations of Sequence TR, depending on the RXACKRQD bit. For both permutations, the Sequence T which constitutes the first half of a Sequence TR, is identical to the standalone Sequence T (XCVSEQ=2). This means that the transmit operation can be slotted or unslotted, and can be preceded by 1 or 2 CCA measurements, depending on the state of the CCABFRTX and SLOTTED bits.

If RXACKRQD=0, then Sequence TR is executed as a Sequence T followed by a Sequence R, with a minimum TX-to-RX turnaround time in between. The Sequence R, which constitutes the second half of the Sequence TR, is identical to the standalone Sequence R (XCVSEQ=1). This means that the receive operation can be followed by an automatic, hardware-generated transmit Acknowledge frame, if all the necessary conditions are met. As with basic Sequence R, data for a successful receive operation is always transferred to Packet Buffer.

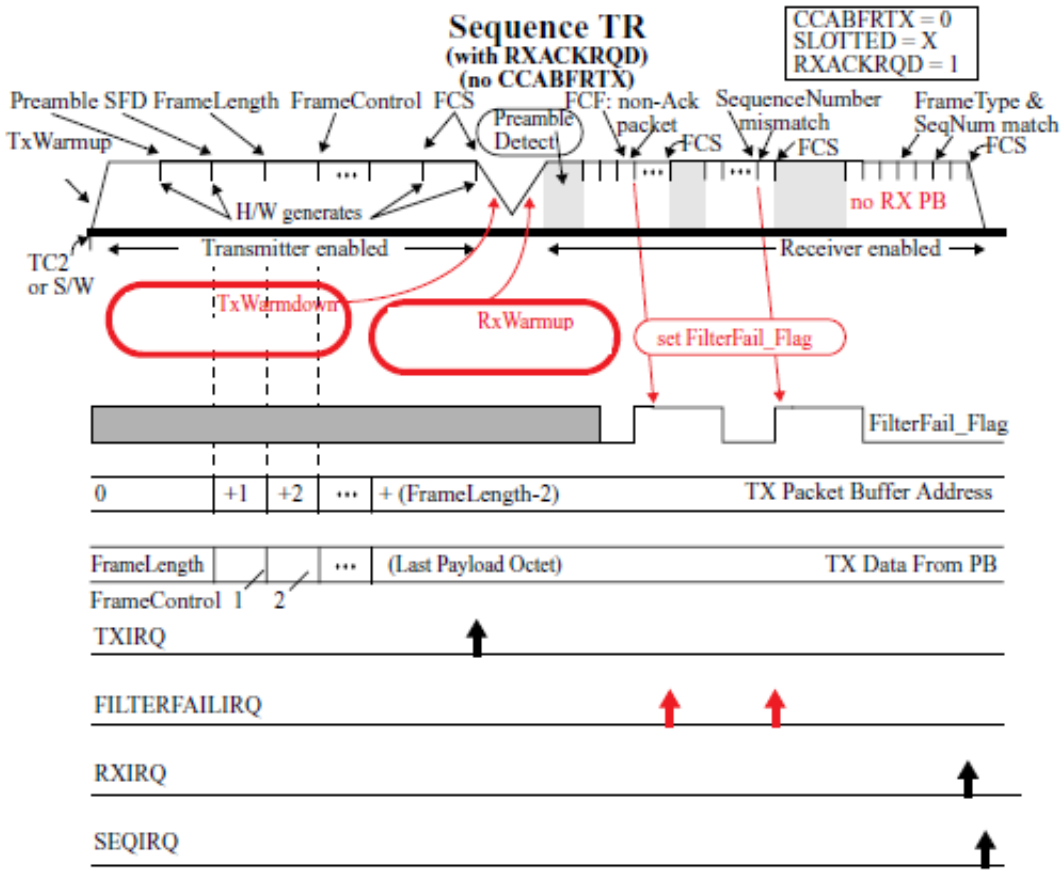
If RXACKRQD=1, then Sequence TR is executed as a Sequence T followed by a Receive-Acknowledge-Only frame. This type of receive operation is special, and not the same as a standalone Sequence R. The Ack-only receive operation filters all incoming frames, looking only for an Acknowledge frame whose Sequence Number matches the Sequence Number which was transmitted in the Sequence T portion of the sequence. All

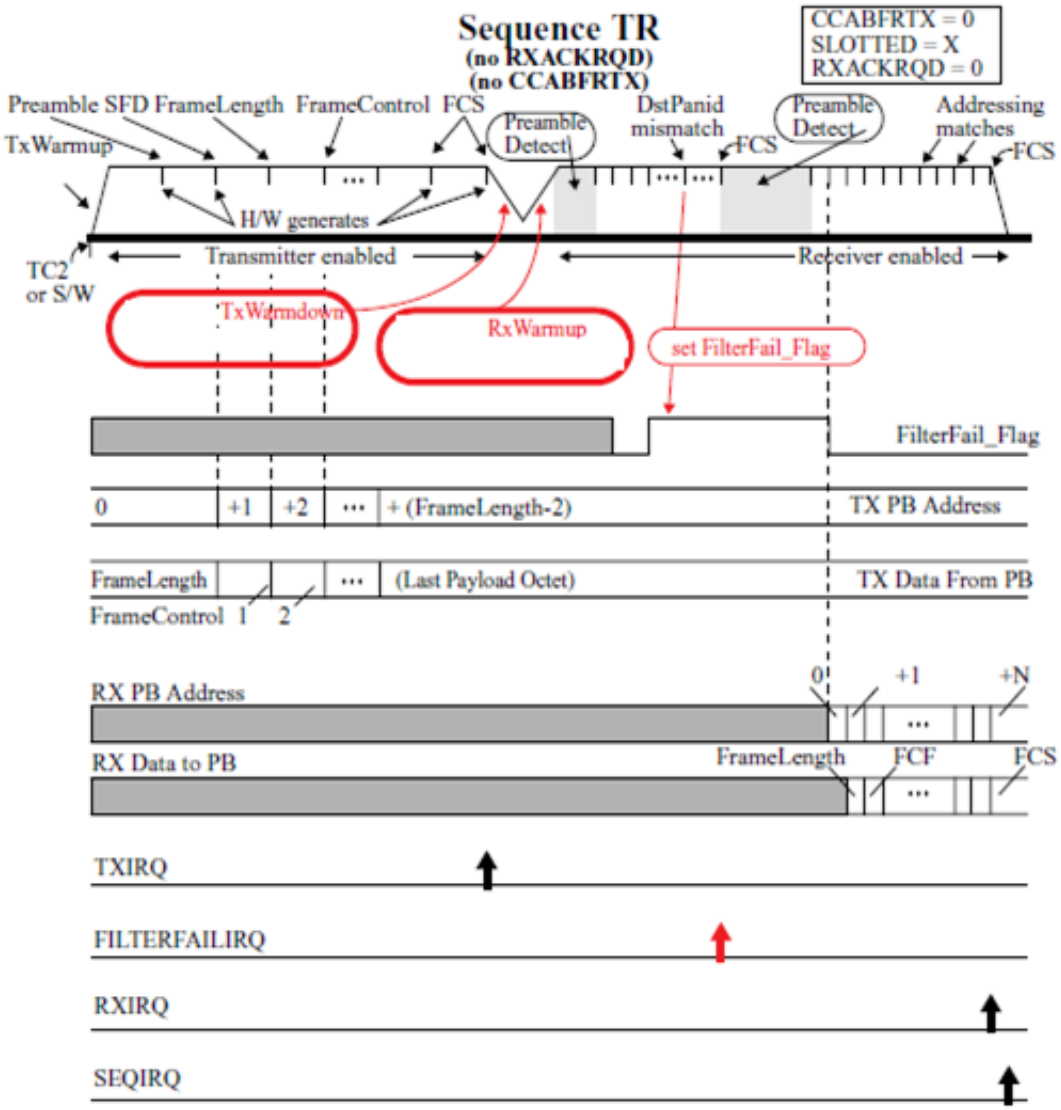
non-matching frames are discarded, and after each non-matching frame, the sequence manager will return the receiver to preamble-detect mode. This receive operation will continue until the matching Ack frame is received. Since this is a Receive-Acknowledge-Only operation, and not a Sequence R, there will be no receive octets transferred to Packet Buffer (an exception is made if ACTIVE_PROMISCUOUS=1; if so, all receive octets are transferred to the Packet Buffer, even for the Receive-Acknowledge-Only frame). The sequence will end with a RXIRQ interrupt, which indicates that the matching Ack frame was successfully received.

Needless to say, if during the transmit operation of a Sequence TR, an Acknowledge frame is being requested of the receiving end device (Frame Control Field: AckRequest=1), then Sequence TR with RXACKRQD=1 is the appropriate procedure. (Using Sequence TR with *RXACKREQ=0* is *not* recommended for this scenario, since there is no Sequence Number matching.)

Conversely, if during the transmit operation of a Sequence TR, an Acknowledge frame is *not* being requested of the receiving end device (Frame Control Field: AckRequest=0), then Sequence TR with RXACKRQD=1 should *never* be used, since a receive acknowledge frame will not be forthcoming. Instead, use Sequence TR with RXACKRQD=0, or simply Sequence T, if no followup receive frame is expected.

## Sequence TR Timing Diagrams


Sequence TR
(with RXACKRQD)
(no CCABFRTX)
CCABFRTX = 0
SLOTTED = X
RXACKRQD = 1
Preamble SFD FrameLength FrameControl FCS
TxWarmup
Preamble Detect
FCF: non-Ack packet
FCS
SequenceNumber mismatch
FCS
FrameType & SeqNum match
FCS
H/W generates
no RX PB
TC2 or S/W
Transmitter enabled
Receiver enabled
TxWarmdown
RxWarmup
set FilterFail_Flag
FilterFail_Flag
0
+1
+2
+ (FrameLength-2)
TX Packet Buffer Address
FrameLength
(Last Payload Octet)
TX Data From PB
FrameControl 1
2
TXIRQ
FILTERFAILIRQ
RXIRQ
SEQIRQ




### *64.5.2.4.2.3.1.6 Sequence CCCA (Continuous CCA)*

Sequence CCCA is the Continuous CCA sequence. This sequence is designed to accommodate situations where channel availability may be infrequent, or low-duty-cycle, and a device needs to be able to transmit at the earliest available opportunity. During Sequence CCCA, the sequence manager repeats CCA measurements continuously until a channel-idle condition is found. The interval between the end of one CCA measurement, and the start of the next, is 63us. The actual interval between consecutive CCA measurements (i.e., from measurement-start to measurement-start), in the CCCA sequence, is 193us. After each CCA iteration, the CCA bit (IRQSTS2 Register) is set to the result of the measurement. As long as the CCA bit is high (busy) at the end of the iteration, the sequence manager will initiate a new measurement. Unlike Sequence C (standalone CCA), there is no CCAIRQ interrupt generated at the end of each CCA

measurement, until the channel-idle condition is detected, at which point CCAIRQ is issued. The SLOTTED bit has no effect on this sequence. A Sequence CCCA will be aborted by the ZSM Sequence Manager if a TMR3 match occurs, and TC3TMOUT=1. CCA Modes 1, 2, or 3, can be used for Continuous CCA.

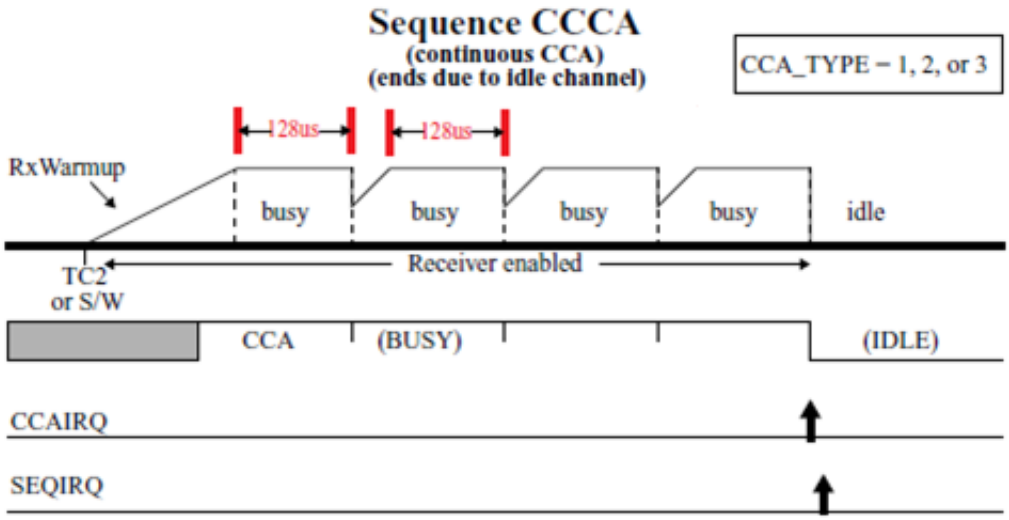
The basic execution of a Sequence CCCA is as follows:

- MCU writes TC2 compare value to the desired initiation time, and sets TMRTRIGEN (optional)
- MCU writes XCVSEQ=CCCA
- Wait for TC2 match (if TMRTRIGEN=1)
- SEQ_MGR asserts one of {**cca1_en, cca2_en, cca3_en** } to CCA_DIG
- SEQ_MGR executes RxWarmup
- SEQ_MGR initiates CCA measurement by asserting **rx_cca_en** to CCA_DIG
- SEQ_MGR waits for CCA to complete
- CCA bit is set to the CCA status (1=busy 0=idle)
- If CCA=1, initiate a new CCA measurement immediately and repeat the previous 4 steps
- If CCA=0, CCAIRQ is issued
- SEQ_MGR executes RxWarmdown
- SEQIRQ interrupt issued

**Note:** The CCA bit (IRQSTS2 register) may change dynamically during the CCA measurement, and are only considered valid (and sampled by the Sequence Manager), at the end of the CCA measurement; therefore the user should not continuously poll CCA during CCCA and be expecting continuously-valid results

**Note:** The user can continuously monitor progress of the Sequence CCCA by reading the CCCA_BUSY_CNT register.

## Sequence CCCA Timing Diagrams

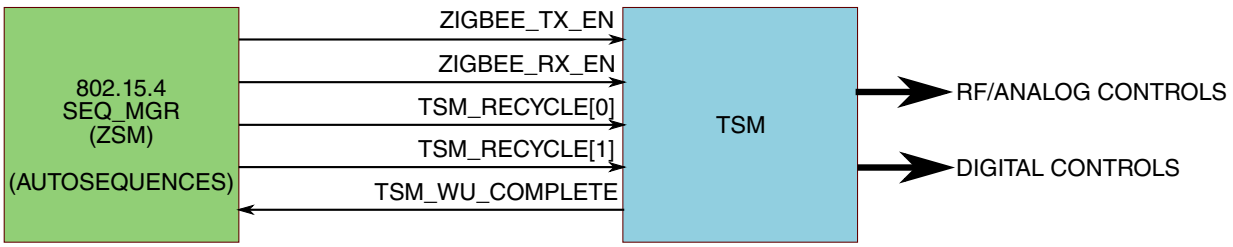



#### *64.5.2.4.2.3.2 ZSM/TSM Interaction*

The 802.15.4 Sequence Manager (ZSM) controls high-level autosequence timing, but offloads low level warmup and warmdown tasks to a dedicated, protocol-neutral Transceiver Sequence Manager, or TSM (see TSM Chapter). The ZSM initiates high-level functions, such as a CCA measurement operation or a transmit acknowledge operation, by commanding the TSM to execute an RX warmup, a TX warmup, a RX warmdown, a TX warmdown, or an RX recycle. Based on these commands, TSM generates the precise timing to control all of the digital, analog, and RF components of the TX and RX chains. The TSM is fully programmable, allowing all of the enabling and control signals to the transceiver components to be independently scheduled with 1us precision. TSM also allows all enable and control signals to be optionally overridden at any time, giving software complete control of each transceiver block, and ultimately the entire warmup/warmdown process if desired.

The ZSM and TSM sequence managers operate on the same clock domain, and both have 1us resolution. The TSM is fully programmable, while the ZSM is largely hardwired, with a few key exceptions. In the ZSM/TSM relationship, ZSM is the master and TSM is the slave. The key ZSM/TSM interface signaling is shown below:

**ZSM/TSM INTERFACE SIGNALS**



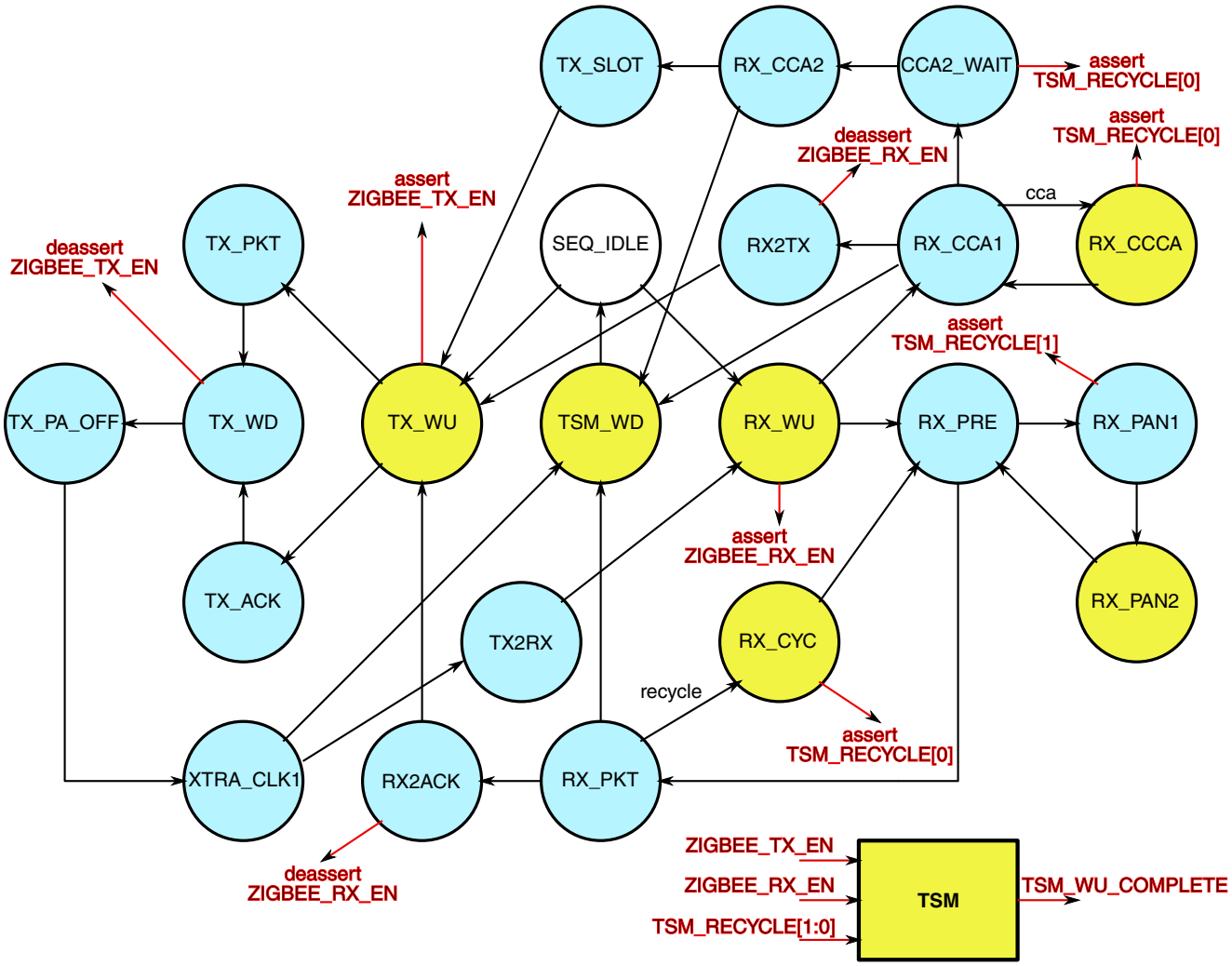

A description of the interface signals is provided in the table below.

| ZSM/TSM INTERFACE SIGNAL | DESCRIPTION |
|---|---|
| ZIGBEE_TX_EN | A low-to-high transition initiates a TSM TX warmup, starting 1us after the transition. A high-to-low transition initiates a TSM TX warmdown, starting 1us after the transition. |
| ZIGBEE_RX_EN | A low-to-high transition initiates a TSM RX warmup, starting 1us after the transition. A high-to-low transition initiates a TSM RX warmdown, starting 1us after the transition. |
| TSM_RECYCLE[0] | This signal asserts in the ZSM RX_CYC state, and causes the TSM to jump from its “ON” phase, back to a point in the TSM RX warmup determined by the TSM’s RECYCLE_COUNT0[7:0] register in XCVR space. This process is used to execute an RX recycle. This signal is also asserted in the ZSM RX_CCCA state, with the same response by the TSM. This process is used to re-initialize the CCA function during Continuous CCA (CCCA) operations. |
| TSM_RECYCLE[1] | This signal asserts in the ZSM RX_PAN1 state, and causes the TSM to jump from its “ON” phase, back to a point in the TSM RX warmup determined by the TSM’s RECYCLE_COUNT1[7:0] register in XCVR space. This process is used to re-initialize the PLL digital on a new RF channel, during a Dual PAN mode on-the-fly channel change. |
| TSM_WU_COMPLETE | For the ZSM states which assert ZIGBEE_TX_EN or ZIGBEE_RX_EN, this 1us-wide signal from the TSM indicates that the warmup is complete, and the ZSM is now free to advance to its next state. The TSM_WU_COMPLETE is asserted 1us before the actual end of sequence, which is determined by the END_OF_TX_WU[7:0] register for TX warmup, and the END_OF_RX_WARMUP[7:0] register for RX warmup. |

#### *64.5.2.4.2.3.3 ZSM State Diagram*

A ZSM state diagram is shown below. The states in yellow, are ZSM states which require interaction with the TSM. For example, a TSM control signal is asserted or deasserted, and/or a wait for warmup-complete from TSM is required; the assertions of the TSM control signals are shown in red. The states in blue have no interaction with TSM, so the timing of these states, and the transitions between such states, are governed by the ZSM itself.

## 802.15.4 SEQUENCE MANAGER AND TSM INTERACTION



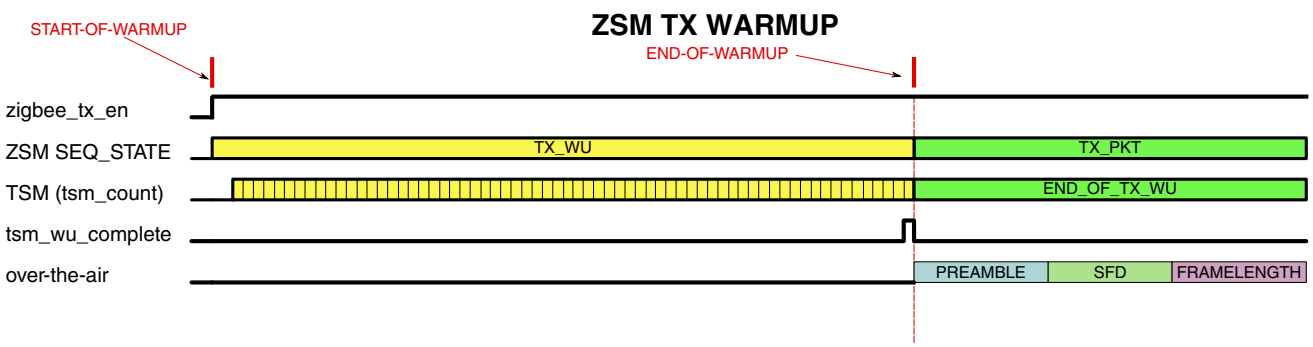

Each ZSM state is represented in the hardware by a 5-bit state vector. The ZSM state can be monitored at any time by way of the read-only SEQ_STATE[4:0] register in 802.15.4 space. The mapping of ZSM state names to state vectors is shown in the table below:

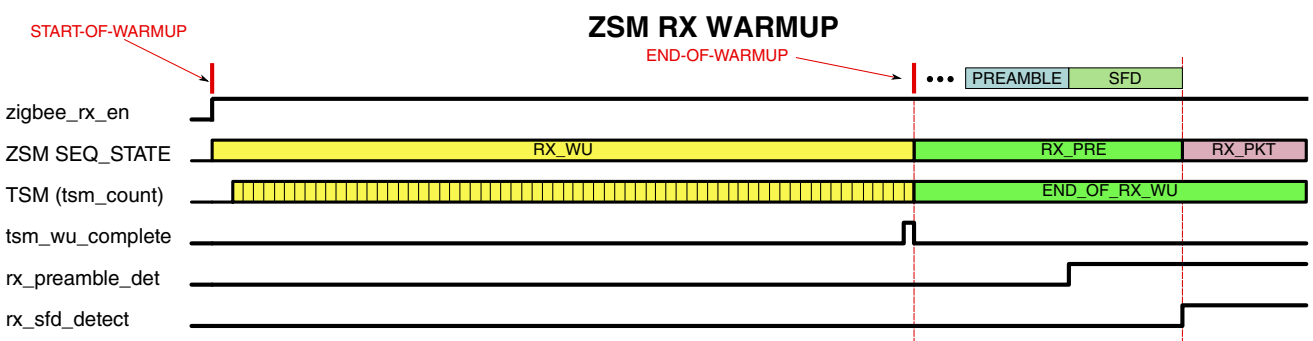
| ZSM STATE | VECTOR |
|---|---|
| SEQ_IDLE | 0x00 |
| RX_WU | 0x10 |
| RX_PRE | 0x14 |
| RX_PKT | 0x15 |
| RX2ACK | 0x16 |
| RX_CYC | 0x17 |
| RX_PAN1 | 0x1D |
| RX_PAN2 | 0x1E |
| RX_CCA1 | 0x18 |

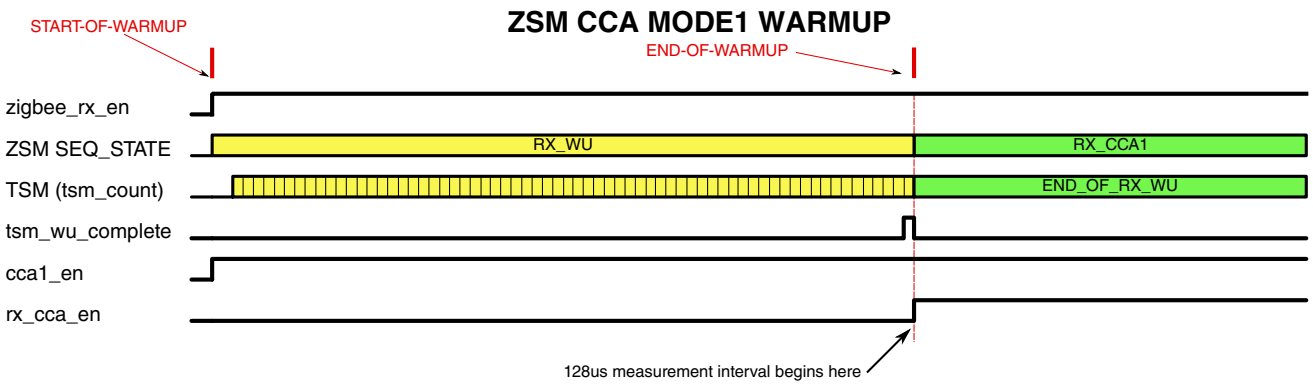
*Table continues on the next page...*

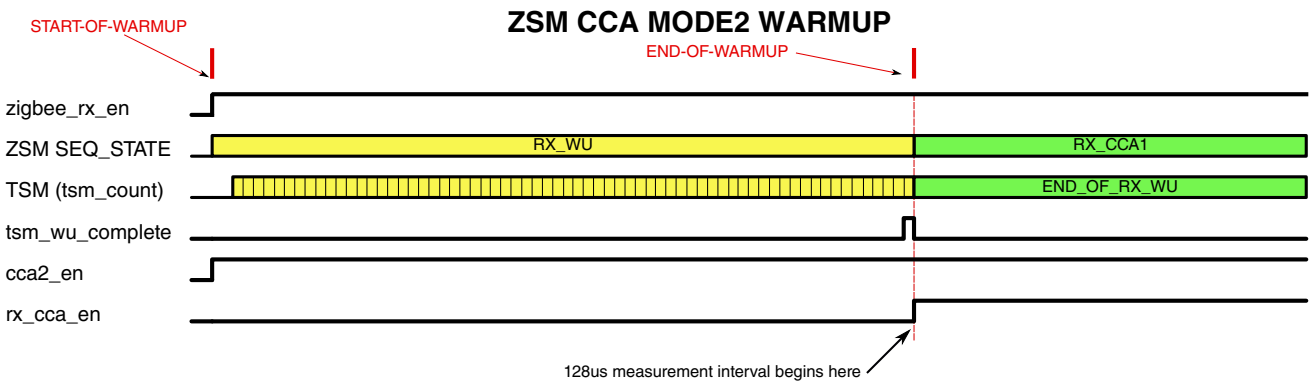
| ZSM STATE | VECTOR |
|---|---|
| CCA2_WAIT | 0x19 |
| RX_CCA2 | 0x1A |
| RX_CCCA | 0x1C |
| RX2TX | 0x1B |
| TX_WU | 0x01 |
| TX_PKT | 0x05 |
| TX_ACK | 0x06 |
| TX_SLOT | 0x07 |
| TX_WD | 0x08 |
| TX_PA_OFF | 0x09 |
| XTRA_CLK1 | 0x0A |
| TX2RX | 0x0B |
| TSM_WD | 0x1F |

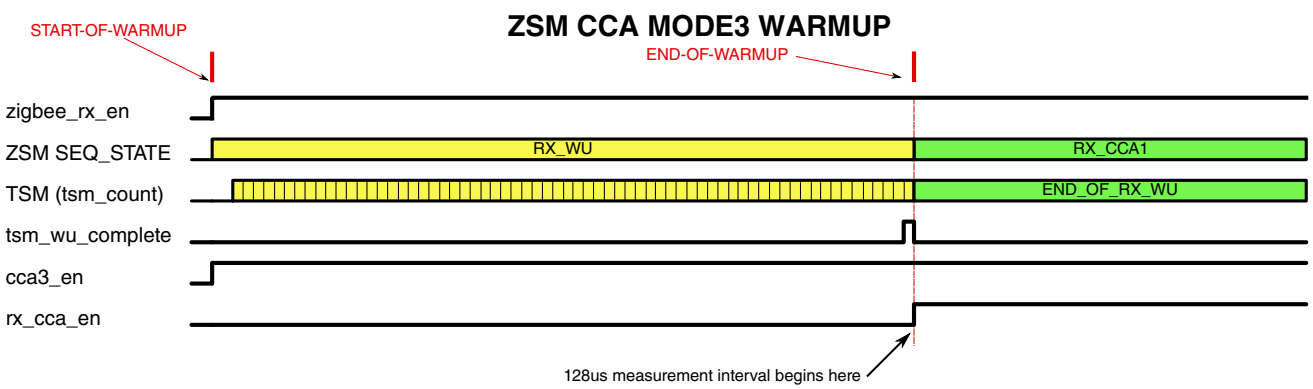
### *64.5.2.4.2.3.4 ZSM Warmup Timing*

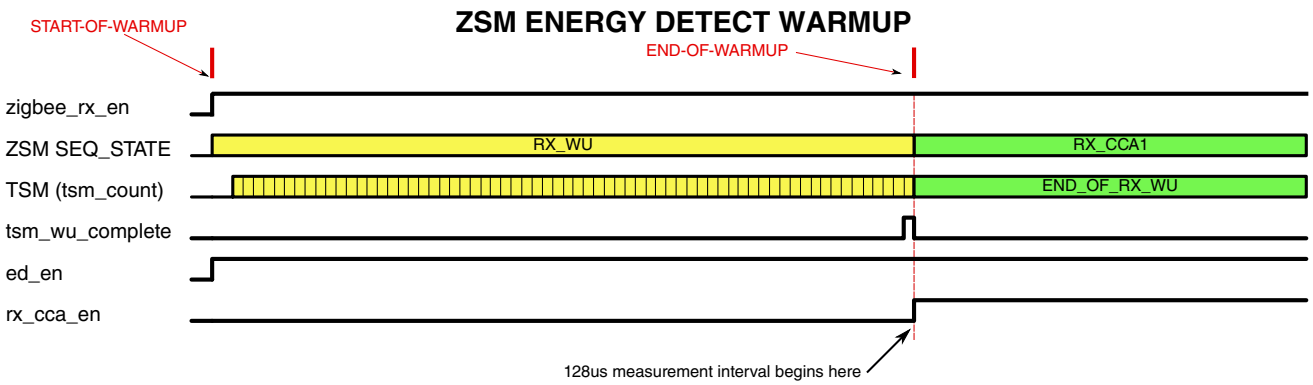



ZSM CCA MODE1 WARMUP
START-OF-WARMUP
END-OF-WARMUP
zigbee_rx_en
ZSM SEQ_STATE
RX_WU
RX_CCA1
TSM (tsm_count)
END_OF_RX_WU
tsm_wu_complete
cca1_en
rx_cca_en
128us measurement interval begins here


ZSM CCA MODE2 WARMUP
START-OF-WARMUP
END-OF-WARMUP
zigbee_rx_en
ZSM SEQ_STATE
RX_WU
RX_CCA1
TSM (tsm_count)
END_OF_RX_WU
tsm_wu_complete
cca2_en
rx_cca_en
128us measurement interval begins here


ZSM CCA MODE3 WARMUP
START-OF-WARMUP
END-OF-WARMUP
zigbee_rx_en
ZSM SEQ_STATE
RX_WU
RX_CCA1
TSM (tsm_count)
END_OF_RX_WU
tsm_wu_complete
cca3_en
rx_cca_en
128us measurement interval begins here




#### *64.5.2.4.2.3.5 Dual PAN Mode*

## Dual PAN On-the-fly Channel Change

The 802.15.4 Sequence Manager (ZSM) supports Dual PAN mode, which allows the device to reside on 2 networks simultaneously, switching back and forth between the 2 networks under software, or hardware control. (See Packet Processor Chapter, Section Dual PAN Mode for more details on Dual PAN mode).

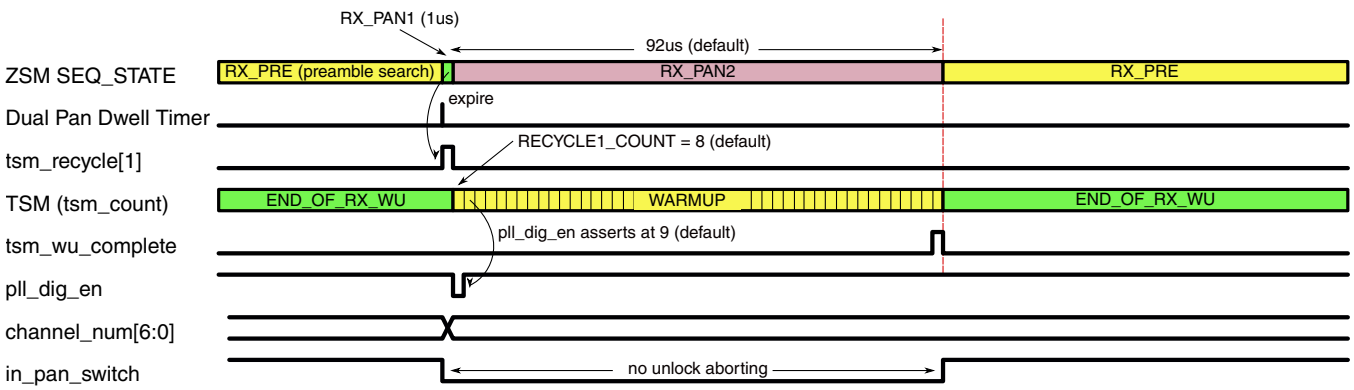
In Automatic Dual PAN Mode, the device must switch from one network to another under hardware control, at a software-programmed rate. Since the 2 networks often occupy different channels (frequencies), the sequence manager supports an “on-the-fly channel change”, whereby, during a Sequence R, in preamble-search state (RX_PRE), the sequence manager will request the TSM to toggle **pll_dig_en** and switch to a new RF channel, wait out a PLL settling time, and then resume preamble search on the new channel. The entire on-the-fly channel change consumes 93us (based on default TSM programming), and consumes 2 ZSM states. The first state (RX_PAN1) issues **tsm_recycle[1]** to the TSM, which causes TSM to jump from its “ON” phase, back to the tsm_count value which is programmed into the RECYCLE_COUNT1[7:0] register in XCVR space. Based on the default programming for RECYCLE1_COUNT, along with the default programming for the TSM timing registers, this TSM recycle will cause the TSM to jump back to the point 1us before **pll_dig_en** is asserted, resulting in a 1us-long deassertion of **pll_dig_en**. The TSM will resume counting from that point, which will re-assert **pll_dig_en**, and the ZSM will simultaneously select the alternate Dual Pan channel. Meanwhile, the ZSM will advance to RX_PAN2 state and wait there pending **tsm_wu_complete** from the TSM. The TSM will resume counting until it once again until it reaches END_OF_RX_WU[7:0], and then will return to its ON phase. At the end of the TSM RX warmup, TSM will issue **tsm_wu_complete** to ZSM, which will return ZSM to RX_PRE (preamble search) state.

Normally, during RX sequences, the PLL is monitored for unlock conditions, and if such an unlock were to occur, the ZSM would abort the RX sequence by deasserting ZIGBEE_RX_EN to TSM. During RX_PAN1 and RX_PAN2 states, ZSM asserts

**in_pan_switch** to the transceiver, which indicates that PLL unlock detection should be temporarily suspended while the channel change and subsequent PLL re-lock takes place. Once the ZSM transitions from RX_PAN2 state back to RX_PRE, ZSM will deassert **in_pan_switch** to resume PLL unlock monitoring.

As mentioned, the entire on-the-fly channel change consumes 93us, based on default TSM programming. This 93us represents a short “blind spot” during which preamble detection can’t occur, while the device switches from one RF frequency to another. See diagram below.

**ZSM DUAL PAN ON-THE-FLY CHANNEL CHANGE**




## Dual PAN Channel Override

When using Dual PAN mode, the channels are assigned to the 2 PAN’s by programming the CHANNEL_NUM0[6:0] and CHANNEL_NUM1[6:0] registers in 802.15.4 space. This method allows a channel number, from 11 to 26, to be programmed into each register, and the channel number is processed downstream into the necessary Integer, Numerator, and Denominator components, required to generate the correct RF frequencies (see PLL Digital Chapter).

In Dual PAN mode, in case there is a need to generate a frequency which may be offset from the 16 prescribed 5MHz-spaced channels, to, for example, avoid interference on one of the Dual PAN channels, a method has been provided to do that, by designating one of the two PAN channels to use the transceiver’s set of direct frequency-programming registers, instead of CHANNEL_NUMx. Programming the direct frequency-programming registers – integer, numerator, and denominator, allows an RF frequency to be selected with much more precision than the 5MHz granularity of the 802.15.4 mapped-channel registers, CHANNEL_NUM0 and CHANNEL_NUM1.

Two bits have been provided in 802.15.4 space to realize this feature: ZB_DP_CHAN_OVRD_SEL and ZB_DP_CHAN_OVRD_EN. When ZB_DP_CHAN_OVRD_EN=1, this enables one of the Dual PAN channels to use the direct frequency programming. The ZB_DP_CHAN_OVRD_SEL bit determines *which* channel uses the direct programming, according to the following table:

| ZB_DP_CHAN_OVRD_EN | ZB_DP_CHAN_OVRD_SEL | PAN0 Frequency Determined by … | PAN1 Frequency Determined by … |
|---|---|---|---|
| 0 | X | CHANNEL_NUM0[6:0] | CHANNEL_NUM1[6:0] |
| 1 | 0 | DIRECT FREQUENCY PROGRAMMING | CHANNEL_NUM1[6:0] |
| 1 | 1 | CHANNEL_NUM0[6:0] | DIRECT FREQUENCY PROGRAMMING |

Direct Frequency Programming is accomplished by setting the PLL's Integer, Numerator, and Denominator registers to the appropriate values for the desired RF frequency. These registers are in XCVR address space, and are shown in the following table:

| DIRECT FREQUENCY PROGRAMMING PARAMETER | REGISTER NAME (XCVR SPACE) | REGISTER MNEMONIC |
|---|---|---|
| Integer | LPM_INTG[6:0] | PLL_LP_SDM_CTRL1 |
| Numerator | LPM_NUM[27:0] | PLL_LP_SDM_CTRL2 |
| Denominator | LPM_DENOM[27:0] | PLL_LP_SDM_CTRL3 |

See the PLL Digital Chapter for RF frequency programming details.

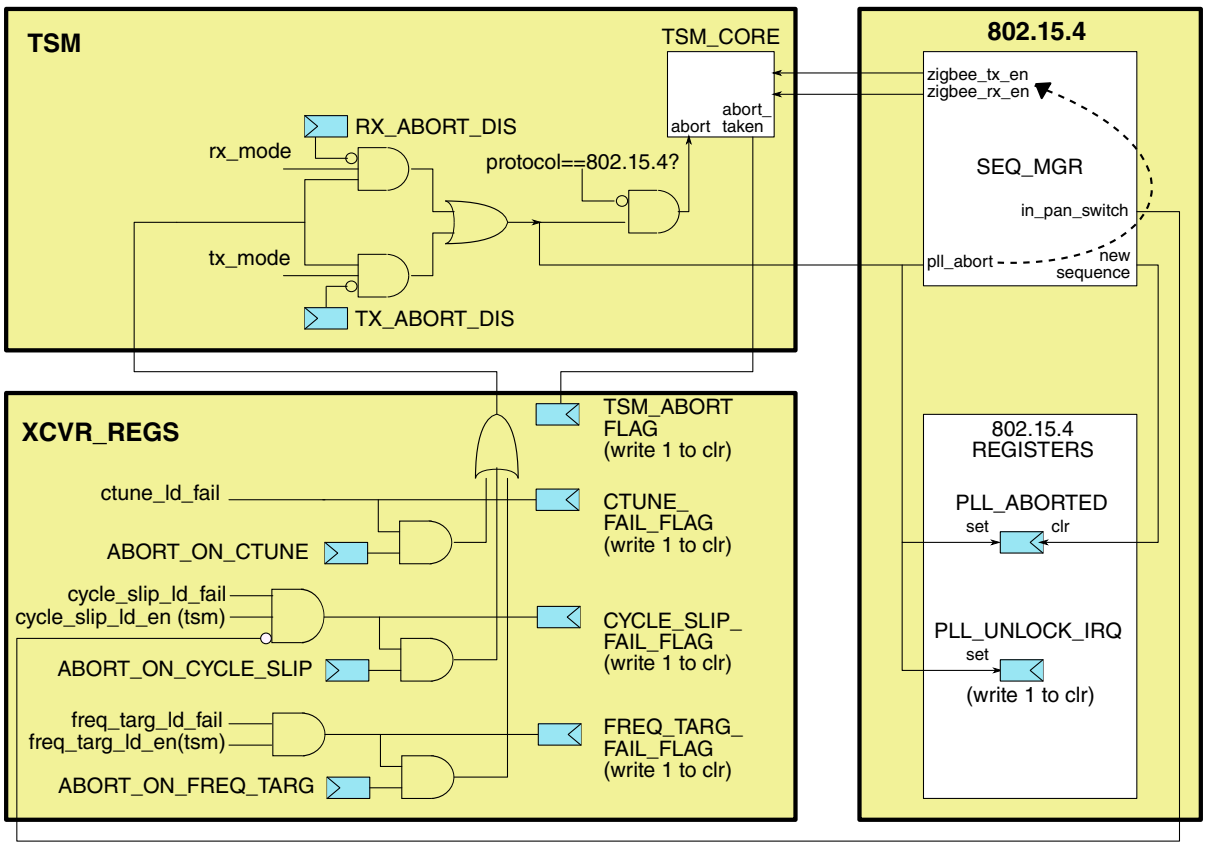
In Dual PAN automatic mode, if ZB_DP_CHAN_OVRD_EN=1, when the hardware switches to the PAN selected by ZB_DP_CHAN_OVRD_SEL as a direct-programming channel, the CHANNEL_NUMx[6:0] register is ignored and the directly programmed frequency is used instead; when the hardware switches back to the other channel, the CHANNEL_NUMx[6:0] register is used once again.

#### *64.5.2.4.2.3.6 Sequence Aborting*

All 802.15.4 sequences can be aborted by a PLL unlock detection. The unlock detection is qualified by the ZSM sequence manager; in other words, an unlock condition during the early stages of the PLL warmup is expected, and is not allowed to abort the sequence at that point. The primary rationale for the PLL unlock abort is to prevent transmission on an incorrect, or unstable, frequency during a transmit operation, or to increase battery life by not prolonging a receive or CCA operation that is bound to fail. Software can disable the PLL unlock auto-abort.

When a PLL unlock event occurs during a 802.15.4 autosequence, the TSM's abort facilities are not used, to allow the ZSM to handle the unlock condition directly, as shown in the following diagram:

**802.15.4 PLL UNLOCK HANDLING**




The combined-abort signal is blocked from reaching the TSM, and instead is routed to the 802.15.4 Sequence Manager (ZSM). This is due to the fact that the ZSM already includes abort-handling logic, which is being re-used from previous Freescale/NXP 802.15.4 products. Keeping the unlock handling intact, facilitates the porting of software from one NXP 802.15.4 product to the next.

For 802.15.4, the logic which determines which ZSM states allow aborting, is designed into the ZSM state machine. Not all states require an abort-on-unlock. When an unlock event occurs during a ZSM state that requires an abort, the ZSM will deassert its initiating signal to TSM (zigbee_tx_en or zigbee_rx_en), which results in a de facto abort, similar to a TSM-generated abort. For a ZSM abort, the PLL_UNLOCK_IRQ bit will become set, as will the PLL_ABORTED bit. The PLL_UNLOCK_IRQ bit is of type write-1-to-clear, and resides in the IRQSTS1 register of 802.15.4 space. The PLL_ABORTED bit is a read-only bit, and resides in the ABORT_STS register of 802.15.4 space; it will self-clear at the start of the next autosequence.

#### *64.5.2.4.2.3.7 Clocks*

The ZSM sequence manager use 2 clocks.

The main clock is **seq_mgr_clk**. This clock establishes the sequence manager timebase (1us), and runs the sequence manager state machine and auxiliary timers.

A secondary clock is "**ck",** which is a gated 32mhz clock from the reference oscillator. This clock is used only for low-latency processing of a few incoming signals, such as the issuing of XCVSEQ sequence commands through the IPS bus interface, which can occur at a rate faster than 1us.

The **seq_mgr_clk** and **ck** derive from the same 32MHz source. Thus, these 2 clocks are in the same clock domain, and will be balanced (skew-controlled) during clock-tree synthesis. The **seq_mgr_clk** will be generated in the CRM. All inputs to the sequence manager are also in this same clock domain, including all the TSM interface signals, and the IPS bus-based control registers; there are no clock-domain-crossings or asynchronous interfaces in the ZSM sequence manager.

#### *64.5.2.4.2.3.8 Reset*

The 802.15.4 Sequence Manager has a single, active-low, asynchronous reset input: **rst_b** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

#### *64.5.2.4.2.3.9 Interrupts*

The 802.15.4 Sequence Manager generates 4 interrupt "triggers":

1. rxirq_trig
2. txirq_trig
3. ccairq_trig
4. filterfail_irq_trig

These triggers are not a direct interrupts to the MCU, but are used to set individual 802.15.4 Interrupt Status bits to become 4 of the 12 802.15.4 Link Layer interrupt sources which, if enabled, will result in the 802.15.4 interrupt **ipi_int_zigbee** being asserted to the MCU. These 4 interrupt triggers will assert the 802.15.4 Interrupt Status Bits RXIRQ, TXIRQ, CCAIRQ, and FILTERFAIL_IRQ, respectively. All of these Interrupt Status Bits reside in the IRQSTS1 register of 802.15.4 Address Space. See Section Interrupts for more details on how the 802.15.4 Link Layer interrupt sources are consolidated into a single MCU interrupt line.

*64.5.2.4.2.3.10 Appendix A*

## Effect of Timer TMR3 on Sequence Manager

Timer TMR3 can be used as a “bracketing” timer, to abort certain sequences after a software-determined timeout period. This timeout mechanism is enabled by setting the TC3TMOUT bit to 1. If TC3TMOUT=0, then the abort mechanism is disabled, and timer TMR3 functions merely as a software timer.

Timer TMR3’s hardware-abort functions are summarized in the following table.

**Table 64-46. TMR3 Hardware Abort Functions**

| SEQUENCE | TC3TMOUT=0 | TC3TMOUT=1 |
|---|---|---|
| I | S/W timer | S/W timer |
| R | S/W timer | Abort sequence |
| T | S/W timer | S/W timer |
| C | S/W timer | S/W timer |
| TR | S/W timer | Abort sequence, (during receive operation only) |
| CCCA | S/W timer | Abort sequence |

*64.5.2.4.2.3.11 Appendix B*

## Continuous TX and RX Modes

Setting the bit CONTINUOUS_EN enables continuous transmission or reception, determined by which type of sequence is subsequently activated using the XCVSEQ register (i.e., Sequence T or Sequence R). Continuous TX and RX modes are designed to be used in tandem; i.e., two communicating devices, one serving as a continuous transmitter, and the other serving as a continuous receiver. The CONTINUOUS_EN resides in the SEQ_MGR_CTRL register of 802.15.4 address space. A description of the continuous TX and RX modes follows:

## Continuous TX Mode

Continuous TX mode instructs the Sequence Manager to stay in a transmitting state forever (TX_PKT state), rather than shutting down the sequence after the 802.15.4 packet has been transmitted. In Continuous TX mode, data to be transmitted must be pre-established in the Packet Buffer prior to initiating a Sequence T. In Continuous TX mode, the first byte of the Packet Buffer determines the length of the buffer, in bytes. In this mode, when the end of the buffer is reached, instead of transmitting FCS (CRC bytes),

the TX modem circles back to the 2nd byte of the buffer (the byte after the length byte), and re-transmits the remainder of the buffer again, and this repeats continuously, until one of the following events occurs:

1) CONTINUOUS_EN is deasserted by SW

2) SW aborts the Sequence T by writing Sequence I to XCVSEQ

3) An PLL Unlock occurs, aborting the Sequence

To prepare for Continuous TX mode, the following steps should be taken:

1) Write the “length” byte to the first byte of the Packet Buffer

2) Write the remaining Packet Buffer contents (a total of “length” bytes)

3) Assert the CONTINUOUS_EN bit

4) Start the Transmit Sequence by writing Sequence T to XCVSEQ.

After Continuous TX mode has been entered, the SEQ_STATE register can be monitored to verify the Sequence Manager has reached the TX_PKT state, and remains in this state.

## Continuous RX Mode

Continuous RX mode instructs the Sequence Manager to stay in a receiving state forever (RX_PKT state), rather than shutting down the sequence after a 802.15.4 packet has been received. For continuous RX mode, it is assumed that another device has been deployed as a continuous transmitter. The length of the continuous transmitter’s TX buffer, will also be the length of the continuous receiver’s RX buffer. For the receiver, in Continuous RX mode, the length of the buffer is determined by the DUAL_PAN_DWELL register. (Needless to say, Continuous RX mode should not be engaged simultaneously with Dual PAN mode, since DUAL_PAN_DWELL is serving an alternate purpose). For Continuous RX mode, DUAL_PAN_DWELL should be programmed to the same value as the “length” byte of the continuous transmitter’s TX buffer. See step 2) above, for Continuous TX mode. In Continuous RX mode, the first byte received (after Preamble and SFD), will be the “length” byte, which can be read from the RX_FRM_LEN register of the receiving device. (It is up to the user to guarantee that this length byte on the TX device matches the contents of the DUAL_PAN_DWELL register on the RX devices, by correctly programming the 2 devices). Each subsequent byte which is received, will be written to the next sequential address of the Packet Buffer, starting at PB address 0. When the number of received bytes matches the contents of DUAL_PAN_DWELL register, the Packet Buffer’s address generation logic will circle back to PB address 0, and store the next received byte to this location. Then, each subsequent received byte, will be written to the next sequential address of the Packet Buffer. In this way, the

contents of TX buffer on the continuous transmitter, will be identical to the contents of the RX buffer on the continuous receiver (assuming no reception errors). To prepare for Continuous RX mode, the following steps should be taken:

1) Program DUAL_PAN_DWELL register with the contents of the “length” byte of the continuously-transmitting device’s TX Buffer

2) Assert the CONTINUOUS_EN bit

3) Start the Receive Sequence by writing Sequence R to XCVSEQ.

After Continuous RX mode has been entered, the SEQ_STATE register can be monitored to verify the Sequence Manager has reached the RX_PKT state, and remains in this state. The contents of the Packet Buffer can also be observed in realtime, to verify they match the contents of the TX device’s TX buffer. The Sequence Manager will remain in the RX_PKT state until one of the following events occurs:

1) CONTINUOUS_EN is deasserted by SW

2) SW aborts the Sequence R by writing Sequence I to XCVSEQ

3) An PLL Unlock occurs, aborting the Sequence

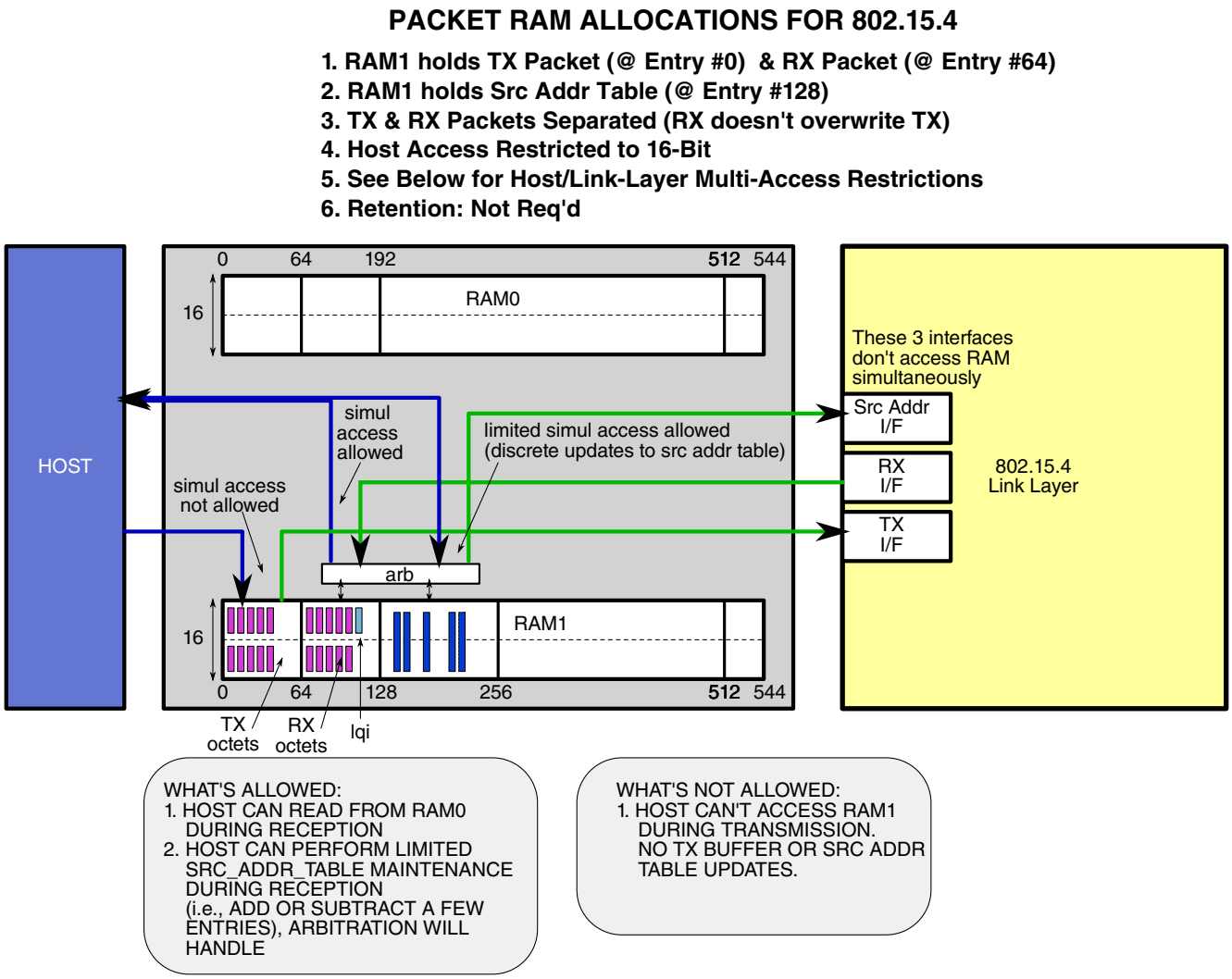
#### 64.5.2.4.3 Packet Storage

The 802.15.4 Link Layer has access to a Packet Buffer to store data to be transmitted, and to receive incoming packet data. The 802.15.4 Link Layer software prepares data to be transmitted, by loading the octets, in order, into the Packet Buffer. After reception, the octets received over the air are stored into the Packet Buffer, in the order they are received. The Packet Buffer contains separate spaces for TX and RX data, so incoming receive octets don’t overwrite previously-loaded content intended for transmission. The Packet Buffer is large enough to accommodate the longest 802.15.4 packet, 127 bytes, for both transmit and receive.

The 802.15.4 Packet Buffer is contained within the multi-protocol Packet RAM. In the Packet RAM, space has been allocated for 802.15.4 packets, with separate spaces for TX and RX. The Packet RAM is 16 bits wide, so the 802.15.4 Link Layer controller hardware will address the RAM in an interleaved fashion as it stores octets to, or fetches octets from, the RAM-based packet buffer. 802.15.4 will utilize space in RAM1, since this RAM is partitionable for multi-protocol operation (RAM0 is not). The 802.15.4 TX buffer space begins at RAM1 Entry #0, and 802.15.4 RX buffer space begins at RAM1 Entry #64. Separate TX and RX spaces means that incoming RX packets don’t overwrite TX packet data.

In addition, the 802.15.4 Link Layer controller includes acceleration for Source Address Matching (see Section Source Address Management). The hardware acceleration includes a Source Address Table, which is 128 entries by 16 bits wide. This table is also implemented in RAM1.

The partitioning of the Packet Buffer RAM1 amongst transmit, receive and source address matching sections for 802.15.4 is shown in the diagram below.




During packet reception, the MCU is allowed to access the RX packet buffer of RAM1, in order to download the packet while it's being received, if so desired. In cases where both the MCU and the 802.15.4 Link Layer controller attempt to access RAM1 during reception, internal hardware arbitration is provided to delay 802.15.4 Link Layer accesses to RAM1 until the MCU completes its access. The longest-duration continuous burst read

by the MCU of RAM1, given a maximum 802.15.4 packet length of 128 octets (127 data octets plus 1 LQI octet) is shorter than the required interval between 802.15.4 Link Layer updates, so the Link Layer can be waited without any possibility of overrun.

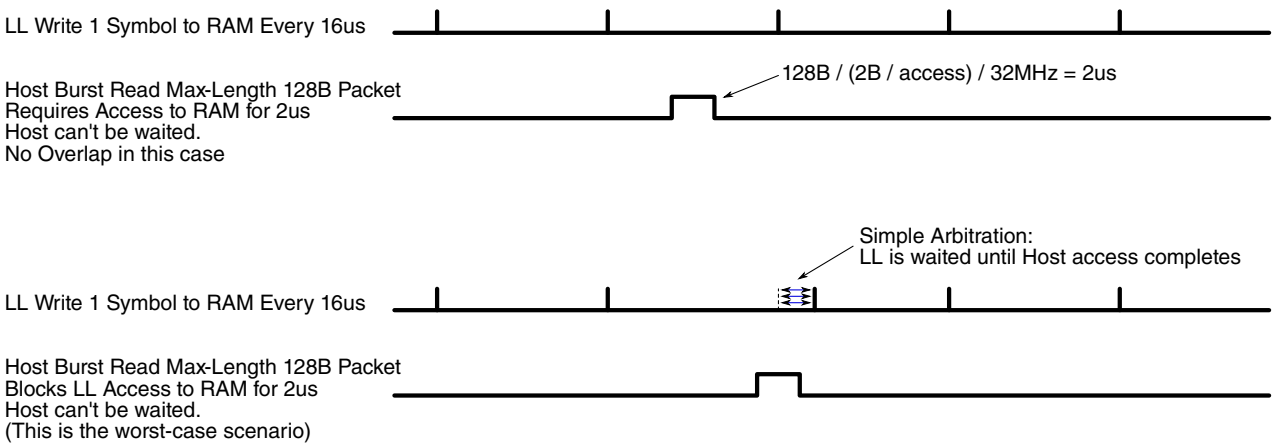
During packet reception, the MCU is allowed to access the Source Address Table partition of RAM1. Access to he Source Address Table is indirect, and is performed through the 802.15.4 Register Interface, specifically by programming SAM_TABLE register with the index of the table entry to be written or read, and the desired checksum in the case of write accesses. Because accesses to Source Address Table consist of discrete writes to 802.15.4 registers, and not long burst accesses, arbitration shall be able to accommodate Link Layer read accesses to the table while the Host access is underway, by delaying the Link Layer access until the Host completes.

During packet transmission, MCU access to RAM1 is not allowed. The 802.15.4 software must program the TX buffer before commanding a TX operation, and must not alter the Source Address Table once transmission has begun.

## Arbitration

Arbitration hardware has been provided to allow the MCU to download data from the Packet RAM while the packet is being received, in order to allow a headstart on software packet processing, if required by the application. In the SoC, there is no provisions to wait the MCU on IPS or APB bus accesses to the radio. Therefore, in case of simultaneous access attempts by the MCU and the 802.15.4 Link Layer to RAM1, arbitration hardware delays the Link Layer access to RAM1 until the MCU completes its access. In the intended scenario, 802.15.4 software sets a RX watermark interrupt, and on this interrupt, proceeds to download the contents of the Packet RX buffer in RAM1. Given the maximum length packet of 127 octets, plus 1 LQI octet, the maximum length read burst by the MCU would be 128 / (2 octets / access) / 32MHz = 2us. The update interval for the 802.15.4 Link Layer is the received symbol period of 16us. Since the longest wait interval the MCU could impose on the Link Layer does not exceed the Link Layer RAM1 update interval, there is no possibility of buffer overrun. The diagram below depicts 2 multi-access scenarios. In the first, the Host and Link Layer accesses do not overlap. In the second, overlap occurs, so arbitration is invoked, and the Link Layer access is slightly delayed.

**RAM ARBITRATION TO ALLOW SIMUL HOST/LL ACCESS DURING RECEPTION**




## 64.5.2.4.4 Memory Mapping

In the SoC memory map, space has been allocated to 802.15.4. Within the SoC, peripherals are allocated memory space in 4KB blocks, called IPS slots. 802.15.4 occupies one such slot. 802.15.4 address space consists of the 802.15.4 registers, as well as the Packet Buffer (RAM), which is memory-mapped into 802.15.4 memory space. The partitioning of address space for 802.15.4 is shown in the diagram below.

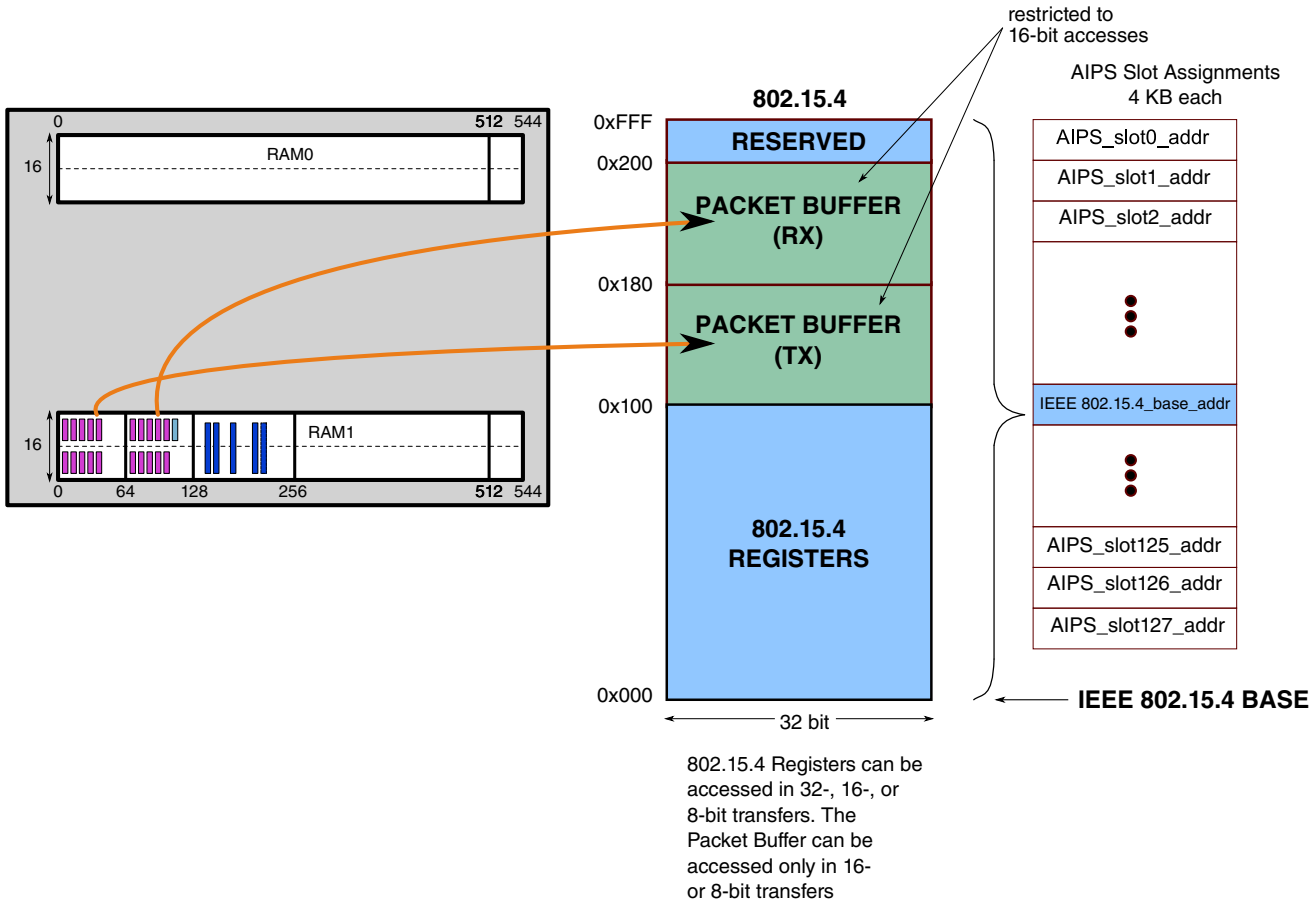



**Figure 64-149. 802.15.4 MEMORY MAPPING**

**NOTE**

See AIPS Memory Map for specific memory assignments details.

The Packet Buffer (RAM), is accessible by the MCU in 16-, or 8-bit accesses. The 802.15.4 Registers are accessible in 32-, 16-, or 8-bit accesses. The Source Address Table partition in RAM, is not accessible directly by the MCU, but via a prescribed protocol using the 802.15.4 SAM_TABLE register (see Section Source Address Management). Therefore, the table is not memory-mapped directly into 802.15.4 address space.

### 64.5.2.4.5 Event Timer

**OVERVIEW**

The 802.15.4 Link Layer Controller contains an internal Event Timer block that manages link layer timing.

The Event Timer consists of a prescaler and a 24-bit counter which increment whenever the 32MHz reference oscillator is operating (i.e., the controller is not in Deep Sleep Mode). Interrupts to the MCU may be generated when the current time of the counter matches several pre-determined values set in programmable registers. The current time is accessible at any time via a read operation, as well as, programmable via a write operation.

The Event Timer provides the following functions:

- Timer to generate current system time
- Interrupt generation at pre-determined system times
- Abort an RX sequence at pre-determined system time
- Latches timestamp value during packet reception
- Initiates timer-triggered sequences

**Note:** The Event Timer is actually a 28-bit value, with 24 integer bits and 4 fractional bits. For most operations involving the Event Timer, such as interrupt-generation and sequence-triggering, only the integer portion need be considered, and the fractional component can be ignored. Extended-precision (28-bit) Event Timer operations are required only to realize the increased accuracy required for entering and exiting Deep Sleep Mode. See Section Extended Precision Event Timer below.

## EVENT TIMER TIME BASE

The Event Timer's base clock (tmr_clk) is derived from a prescaler which is clocked by the 32MHz crystal source. The prescaler provides a counter input frequency of 62.5KHz, which sets the granularity and resolution of the current time. The 24-bit counter automatically rolls over upon reaching its maximum value, yielding a maximum possible Event Timer duration of 268.436 seconds.

## READING CURRENT TIME

"Current Time" is defined as the value of the Event Timer internal counter. This 24-bit value can be read as the upper 3 bytes of the EVENT_TMR register. The EVENT_TMR register resides at at address ZLL_BASE + 0x8, so the upper 3 bytes begin at ZLL_BASE + 0x9. The 24-bit value represents the integer component of the 28-bit Event Timer, which is all that is needed for most Event Timer operations. (For accessing the extended precision Event Timer, see Section Extended Precision Event Timer below.)

## SETTING CURRENT TIME

"Current Time" is defined as the value of the Event Timer internal counter. This 24-bit value is available as the upper 3 bytes of the EVENT_TMR register. The EVENT_TMR register resides at at address ZLL_BASE + 0x8, so the upper 3 bytes begin at ZLL_BASE + 0x9. The current time is programmable, but does not have to be

programmed. In the reset condition, the current time is set to zero. Current time advances from zero at the tmr_clk clock rate and rolls over to zero after reaching its maximum value.

The Event Timer can be updated in one of 2 ways:

EVENT_TMR_LD: Write the desired new EVENT_TMR value into the EVENT_TMR field of the EVENT_TMR register (upper 3 bytes), with the EVENT_TMR_LD bit set.

EVENT_TMR_ADD: Write the desired *increment* to the EVENT_TMR value into the EVENT_TMR field of the EVENT_TMR register (upper 3 bytes), with the EVENT_TMR_ADD bit set. The increment will be added to the current value of EVENT_TMR to become the new EVENT_TMR value. Since the Event Timer value is 24-bits, the addition is modulo $2^{24}$.

Needless to say, EVENT_TMR_LD and EVENT_TMR_ADD are mutually exclusive, and no more than 1 of these bits should be set.

The 24-bit value represents the integer component of the 28-bit Event Timer, which is all that is needed for most Event Timer operations. (For accessing and updating the extended precision Event Timer, see Section Extended Precision Event Timer below.)

## LATCHING THE TIMESTAMP

During packet reception, the 802.15.4 link layer controller has the ability generate a timestamp, or to latch a copy of the "current time" while continuing to increment its internal counter. This timestamp value latched within the Event Timer corresponds to the beginning of a receive packet where the actual packet data begins after the SFD (Start-of-Frame-Delimiter) has been received. The timestamp[23:0] can be accessed by reading the read-only register of the same name, TIMESTAMP, which resides at address ZLL_BASE + 0xC. Once the timestamp has been latched, it shall remain valid until the *next* SFD detection.

## EVENT TIMER COMPARATORS

There are four 24-bit timer compare fields:

1. T1CMP[23:0] at address ZLL_BASE + 0x10
2. T2CMP[23:0] at address ZLL_BASE + 0x14
3. T3CMP[23:0] at address ZLL_BASE + 0x1C
4. T4CMP[23:0] at address ZLL_BASE + 0x20

And a special 16-bit timer compare field, for situations where the upper byte of EVENT_TMR is a don't-care.

- T2PRIMECMP[15:0] at address ZLL_BASE + 0x18

The TMR2IRQ status bit, can be set by a match to T2CMP or T2PRIMECMP. If register bit TC2PRIME_EN=1 (PHY_CTRL register, address ZLL_BASE + 0x04), then TMR2IRQ becomes set by a match to T2PRIMECMP (16-bit compare); if TC2PRIME_EN=0, then TMR2IRQ becomes set by a match to T2CMP (24-bit compare).

Each timer comparator has a enable bit that enables or disables the compare function. The enable bit is written to a 1 to enable the corresponding comparator, and the default condition is the timer compare disabled (reset to 0):

1. TMR1CMP_EN (for T1CMP)
2. TMP2CMP_EN (for T2CMP and T2PRIMECMP)
3. TMR3CMP_EN (for T3CMP)
4. TMR4CMP_EN (for T4CMP)

If a timer comparator is enabled using its associated bit, the corresponding interrupt status bit (TMRxIRQ) will be set by a timer compare match to its respective TxCMP register. If a timer comparator is disabled using its associated bit, the corresponding interrupt status bit (TMRxIRQ) will not be set by a timer compare match to its respective TxCMP register.

When enabled, all four fields can be continuously compared to the current value of the Event Timer counter. When a match occurs, the following corresponding internal status flags assert:

1. TMR1IRQ (T1CMP match)
2. TMR2IRQ (T2CMP or T2PRIMECMP match)
3. TMR3IRQ (T3CMP match)
4. TMR4IRQ (T4CMP match)

These 4 status bits all reside in the IRQSTS register at address ZLL_BASE + 0x0. The 4 interrupt status bits are write-1-to-clear. The status bit remains set until a 1 is written to the bit location in the IRQSTS register. Writing 0 to the bit, or reading the bit, will not change its state.

When a comparator match occurs and the internal status flag asserts, the following interrupt masks can enable an interrupt on the ipi_int_zigbee line to the MCU:

1. TMR1MSK, applies to TMRQ1IRQ
2. TMR2MSK, applies to TMRQ2IRQ
3. TMR3MSK, applies to TMRQ3IRQ
4. TMR4MSK, applies to TMRQ4IRQ

If the interrupt mask is set to 1 (masked), the respective interrupt status bit, TMRxIRQ, will not cause an interrupt on the ipi_int_zigbee line. If the interrupt mask is set to 0 (enabled), the respective interrupt status bit, TMRxIRQ, will cause an interrupt on ipi_int_zigbee. Timer mask bits, TMRxMSK, have no effect on the state of the TMRxIRQ interrupt status bits; they only allow (or prevent) the respective TMRxIRQ bit from generating an interrupt to the MCU on ipi_int_zigbee.

## INTENDED EVENT TIMER USAGE

It is intended that the system utilize the current time value and the timer compare functions of the Event Timer to schedule system events, including:

1. Generating time-based interrupts
2. Aborting an RX operation
3. Triggering transceiver operations

## GENERATING TIME-BASED INTERRUPTS

Generating time-based interrupts is accomplished by setting timer compare values relative to the current time, allowing the Event Timer counter to increment until a timer compare match is generated, and using this match to generate an interrupt to the host. The general procedure is as follows:

1. Disable the timer compare: set TMRxCMP_EN=0
2. Enable the timer compare interrupt mask: set TMRxMSK=1
3. Read the current time value from EVENT_TMR register
4. Determine an offset to this value to equal desired future time.
5. Write to EVENT_TMR register with the EVENT_TMR field set to this offset, and set EVENT_TMR_ADD=1
6. Re-enable the timer compare: set TMRxCMP_EN=1
7. Unmask the timer compare interrupt mask: set TMRxMSK=0
8. Wait for TMRxIRQ
9. After TMRxIRQ, program the appropriate TMRxCMP_EN bit to disable the compare function. If this is not done, the compare function will continue to run and generate another interrupt every time the counter rolls over and again matches the comparator.

## USING T3CMP TO ABORT AN RX OPERATION

The Event Timer provides a timer-based mechanism to automatically abort an RX operation. The 802.15.4 link layer is put into an RX operation when XCVSEQ[2:0] (PHY_CTRL register, address ZLL_BASE + 0x4) is set to 0x01. An RX autosequence will commence, and will begin a search for preamble. Software can program in advance,

a hardware timeout to occur, such that the RX autosequence can be automatically aborted by hardware after a certain amount of time elapses without a packet being detected. The general procedure is as follows:

1. Read the current time value from EVENT_TMR register
2. Add an offset to this value to equal desired future time to abort the RX operation.
3. Program register T3CMP[23:0] to value future time.
4. Program TC3TMOUT=1, (PHY_CTRL Register, address ZLL_BASE + 0x4) to 1, to enable hardware aborting of the RX operation.
5. Program XCVSEQ=1, to start the RX autosequence
6. When current time equals T3CMP[23:0], the 802.15.4 Sequence Manager aborts the RX autosequence, and the Sequence Manager returns to SEQ_IDLE state. The SEQIRQ interrupt status bit (IRQSTS Register, address ZLL_BASE + 0x0) will be set, and the TMR3IRQ interrupt status bit will also be set. The RXIRQ interrupt status bit will not be set, since no packet was received before the timeout. In the SEQ_CTRL_STATUS register (address ZLL_BASE + 0x78), the TC3_ABORTED status bit will also be asserted to indicate that the type of abort that occurred was a TMR3 timeout. This bit will self-clear at the start of the next autosequence.

As mentioned above, TMR3 with TC3TMOUT=1 can be used to abort an RX operation. Such an RX operation can be entered by launching a Sequence R (XCVSEQ=1), and can also be entered by launching a Sequence TR (XCVSEQ=4); after the TX operation of a Sequence TR completes, an RX operation begins, and can be aborted using this method. In addition, a Continuous CCA autosequence (XCVSEQ=5) can be aborted using this method. When a TMR3 timeout occurs with TC3TMOUT=1 during transceiver operations other than those listed above, TMR3 will not abort the autosequence, and TMR3 becomes merely a software timer, capable of setting TMR3IRQ and interrupting the MCU. The table below shows the effect of TMR3 timeouts on the 802.15.4 autosequences:

**Table 64-47. TMR3 Hardware Abort Functions**

| SEQUENCE | TC3TMOUT=0 | TC3TMOUT=1 |
|---|---|---|
| I | S/W timer | S/W timer |
| R | S/W timer | Abort sequence |
| T | S/W timer | S/W timer |
| C | S/W timer | S/W timer |
| TR | S/W timer | Abort sequence, (during receive operation only) |
| CCCA | S/W timer | Abort sequence |

## USING T2CMP OR T2PRIMECMP TO TRIGGER TRANSCEIVER OPERATIONS

The Event Timer provides a timer-based mechanism to automatically launch a transceiver autosequence. Using this method, a sequence can be scheduled for time in the future, and the MCU can be put into a reduced power state, without the need to wake up at the appropriate time to start the autosequence. The general procedure is as follows.

1. Read the current time value from EVENT_TMR register.
2. Add an offset to this value to equal desired future time to launch the autosequence.
3. For a 24-bit compare, program register T2CMP[23:0] to value future time, and set TC2PRIME_EN=0. For a 16-bit compare, program register T2PRIMECMP[15:0] to value future time, and set TC2PRIME_EN=1. Set TMRTRIGEN=1 and program the desired autosequence into XCVSEQ[2:0]. The Sequence Manager will not launch the sequence immediately because it sees TMRTRIGEN=1.
4. When current time equals T2CMP[23:0] (for a 24-bit compare), or T2PRIMECMP[15:0] (for a 16-bit compare), the Sequence Manager will launch the scheduled autosequence in accordance with the XCVSEQ[2:0] field. The TMR2IRQ interrupt status bit will become set simultaneous with the autosequence launch.
5. The MCU can wake up at this point (if TMR2MSK=0), or can elect to continue in low-power state until the autosequence completes, at which time SEQIRQ will become set. If SEQMSK=0, then SEQIRQ will interrupt the MCU at the completion of the transceiver operation.

## EXTENDED PRECISION EVENT TIMER

For most 802.15.4 transceiver operations referencing the Event Timer, the 24-bit precision and the 62.5KHz update rate (16µs period) are sufficient. However, the required precision for maintaining the 802.15.4 timebase is tighter: 1 microsecond, instead of 16. As long as the 32MHz reference oscillator is running there is no difficulty maintaining that precision. However, when the link layer has to enter low-power state, and the reference oscillator must be switched off, and alternate clock source must be temporarily employed to keep the timebase running. This low power state is known as Deep Sleep Mode, or DSM. The alternate clock source, a 32KHz crystal oscillator, is equally as precise as the reference oscillator, but with a much longer period. At the instant the reference oscillator is turned off and the switchover to the 32KHz is made, the EVENT_TMR becomes frozen at its current state. After a period of DSM, when the precisely-programmed wakeup point is reached, the reference oscillator, which has been previously restarted, is re-engaged by the link layer, and the EVENT_TMR resumes counting where it left off. The precise duration of time for which the EVENT_TMR was frozen, is known to software, based on DSM programming parameters. The EVENT_TMR must be corrected for this "time slept", by adding to it an amount equal to "time slept", in microseconds (hence the requirement for microsecond precision). After the precise correction is made, the EVENT_TMR is now up-to-date, as if the DSM entry/exit had never happened, and all previously scheduled events will occur on time, within 1µs precision.

To realize the required microsecond precision, the actual Event Timer has been expanded to 28 bits, with 24 integer bits and 4 fractional bits, as shown in the diagram below.

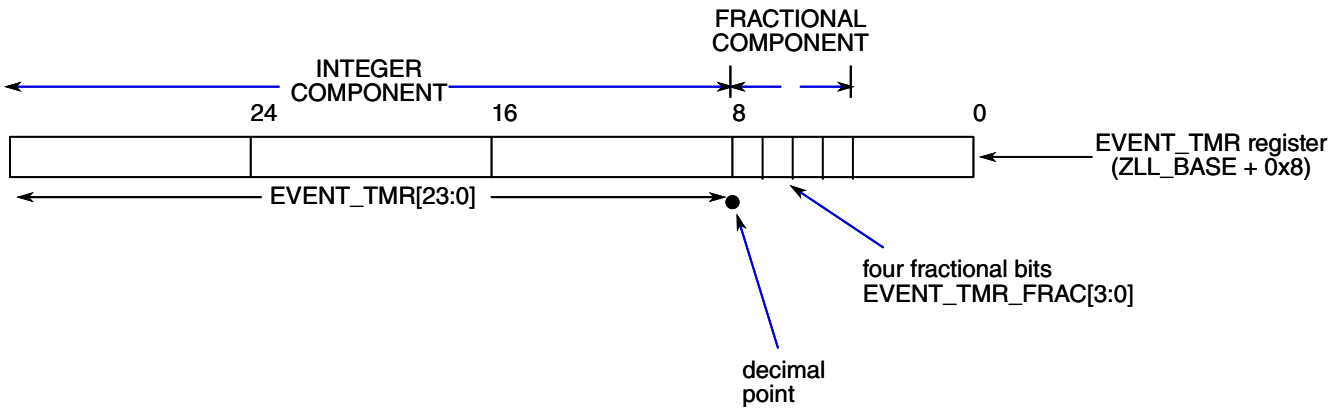



**Figure 64-150. Event Timer Integer and Fractional Components**

The 24-bit integer component is the EVENT_TMR value that has been referenced earlier in this section; most 802.15.4 operations require only the integer component, and the fractional component can be ignored. When the fractional bits are not required, writes to EVENT_TMR register (read-modify-write operations) with EVENT_TMR_LD=1 should leave the fractional bits unchanged. Transceiver operations which reference the EVENT_TMR (i.e., automatic sequence launching, and RX sequence aborting), reference only the integer component, as shown below.

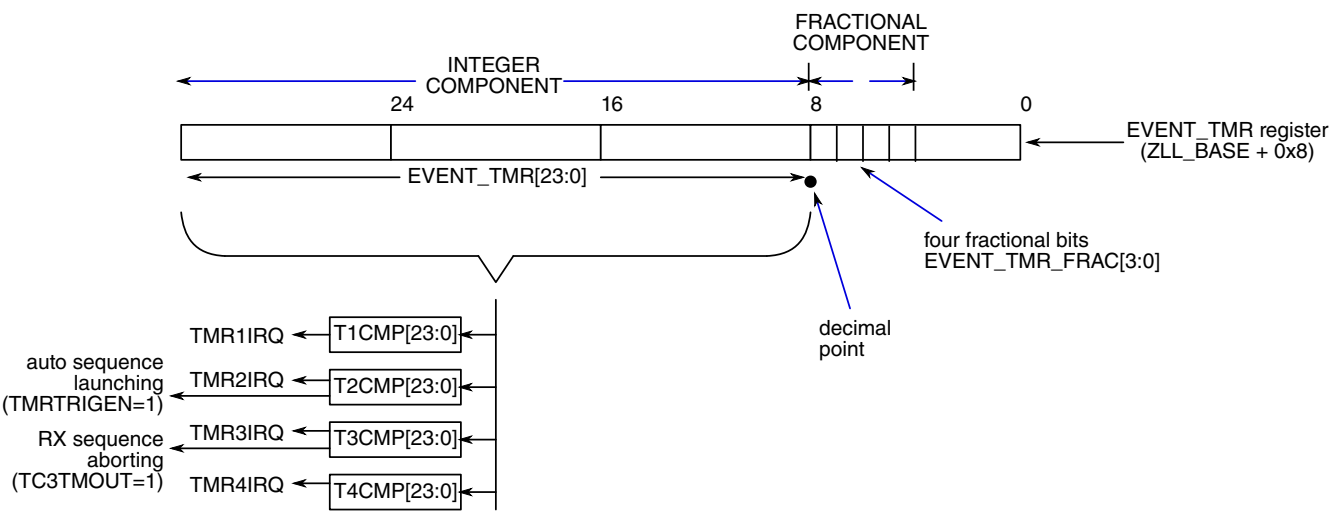



**Figure 64-151. Event Timer Hardware Referencing**

However, software EVENT_TMR updates to correct for "time slept" after a DSM cycle must make use of the full precision, including the fractional bits, in order to maintain microsecond accuracy. For example, say a DSM cycle was just completed in which the EVENT_TMR was frozen for exactly 163µs. The 163µs must be "added back" to the EVENT_TMR.

In this case the integer component is:

integer(163µs / 16µs) = **10**

And the fractional component is the remainder x 16:

(163µs / 16µs) - integer(163µs / 16µs) = 3/16 x 16 = **3**

So the value written to the EVENT_TMR with the EVENT_TMR_ADD bit set, accounting for the bit shifts to align to the integer and fractional components of the EVENT_TMR register, is:

(10 << 8) | (3 << 4), *or,*

EVENT_TMR = (10 << 8) | (3 << 4) | EVENT_TMR_ADD;

This operation adds 163µs to the EVENT_TMR current time, and after the update has occurred, the 802.15.4 timebase has been restored, as if no DSM cycle had occurred, within 1µs accuracy.

#### 64.5.2.4.6 Simultaneous CCA/RX

The 802.15.4 Link Layer controller features an autosequence that precedes a packet transmission with a CCA (Clear Channel Assessment), to ensure the channel is idle before transmitting. If the CCA indicates a busy channel, the autosequence ends after the CCA, without transmitting. In slotted mode, 2 CCA measurements take place, with the start points for the measurements 320us apart. Both must indicate an idle channel for transmission to proceed.

During intervals of severe network congestion, high node density, and packets which are heavily fragmented, the situation can arise where Device A is preparing to transmit a packet, or fragment, and performing pre-transmit CCA, and during the CCA, Device B attempts to transmit another packet or fragment to Device A. Previous NXP 802.15.4 transceivers were unable to receive a packet during CCA, so Device B's packet will be missed by Device A. For the Gen3 2.4GHz radio, the capability to receive-during-CCA has been added to 802.15.4. This capability is referred to as "Simultaneous CCA/RX".

To enable this capability, 3 enhancements have been made to the digital receiver and the 802.15.4 sequence manager:

1. CCA Mode 2 Algorithm Separated from 802.15.4 Demodulator
2. New 802.15.4 State Machine Transitions
3. New TSM Fast RX-to-TX Capability

**1. Separated CCA Mode 2 Algorithm**

One reason for the lack of prior support, is that the CCA Mode 2 algorithm has been tightly integrated with the 15.4 demodulator, so that both functions cannot simultaneously co-exist. The Gen3 digital receiver separates these functions, removing the CCA Mode 2 impediment to Simultaneous CCA/RX. See Chapter *Receiver Digital Module*, Section *IEEE 802.15.4 CCA mode 2*. (CCA Mode 1 does not utilize the demodulator, so such impediment exists.)

The legacy CCA Mode 2 hardware has been retained as a back-up. The control bit USE_DEMOD_CCA2 selects between legacy (1) and new CCA Mode 2 (0). This bit must be set to 0 to utilize Simultaneous CCA/RX. The following table summarizes the programmability associated with CCA Mode 2. The third column references table *IEEE 802.15.4 CCA mode 2 parameters* in Chapter *Receiver Digital Module*, Section *IEEE 802.15.4 CCA mode 2*

| REGISTER NAME | ADDRESS SPACE | ASSOCIATED CCA MODE 2 PARAMETER | DESCRIPTION |
|---|---|---|---|
| USE_DEMOD_CCA2 | XCVR | n/a | CCA Mode 2 hardware selector.<br>• 0: use new CCA Mode 2<br>• 1: use demodulator-based (legacy) CCA Mode 2 |
| CCA2_CORR_THRESH[7:0] | 802.15.4 | zb_cca2_thresh_corr[7:0] | Threshold used to compare correlation products. **Note:** this register is shared between the new and legacy CCA Mode 2 hardware |
| CCA2_CNT_THRESH[9:0] | XCVR | zb_cca2_thresh_cnt[9:0] | Threshold to compare count of correlation products exceeding CCA2_CORR_THRESH[7:0] |
| CCA2_SYM_THRESH[9:0] | XCVR | zb_cca2_thresh_sym[9:0] | Threshold used to detect repetition sequence addresses corresponding to correlation products that exceed CCA2_CORR_THRESH[7:0] |
| CCA2_REF_SEQ[7:0] | XCVR | zb_cca2_ref_seq[7:0] | Selection of the sequence addresses to be used. |
| CCA2_INTERVAL[1:0] | XCVR | zb_cca2_duration[1:0] | Specifies the CCA2 measurement window duration:<br>• 0x00 - 64 µs<br>• 0x01 - 128 µs<br>• 0x02 - 256 µs<br>• 0x03 - 512 µs |

**Note:** Set CCA2_DURATION=0x01 (128us) for 802.15.4 operation.

## 2. New 802.15.4 State Machine Transitions

Using Simultaneous CCA/RX, the 802.15.4 Link Layer controller needs to be able to anticipate preamble, and even SFD detection, during the states in which it is taking CCA measurement. Those states are RX_CCA1, for non-slotted, and RX_CCA2, for slotted operation. If preamble and/or SFD is detected in those states, the 802.15.4 sequence manager needs to transition to a packet reception state, and then receive the packet in its entirety as if the called sequence was Sequence R. To address this, these new state transitions have been added to the 802.15.4 Sequence Manager (ZSM) state machine:

- RX_CCA1 -> RX_PRE (preamble detected during 1st CCA but no SFD)
- RX_CCA1 -> RX_PKT (preamble + SFD detected during 1st CCA)
- RX_CCA2 -> RX_PRE (preamble detected during 2nd CCA but no SFD)
- RX_CCA2 -> RX_PKT (preamble + SFD detected during 1st CCA)

The following diagram highlights the new ZSM state transitions, in red:

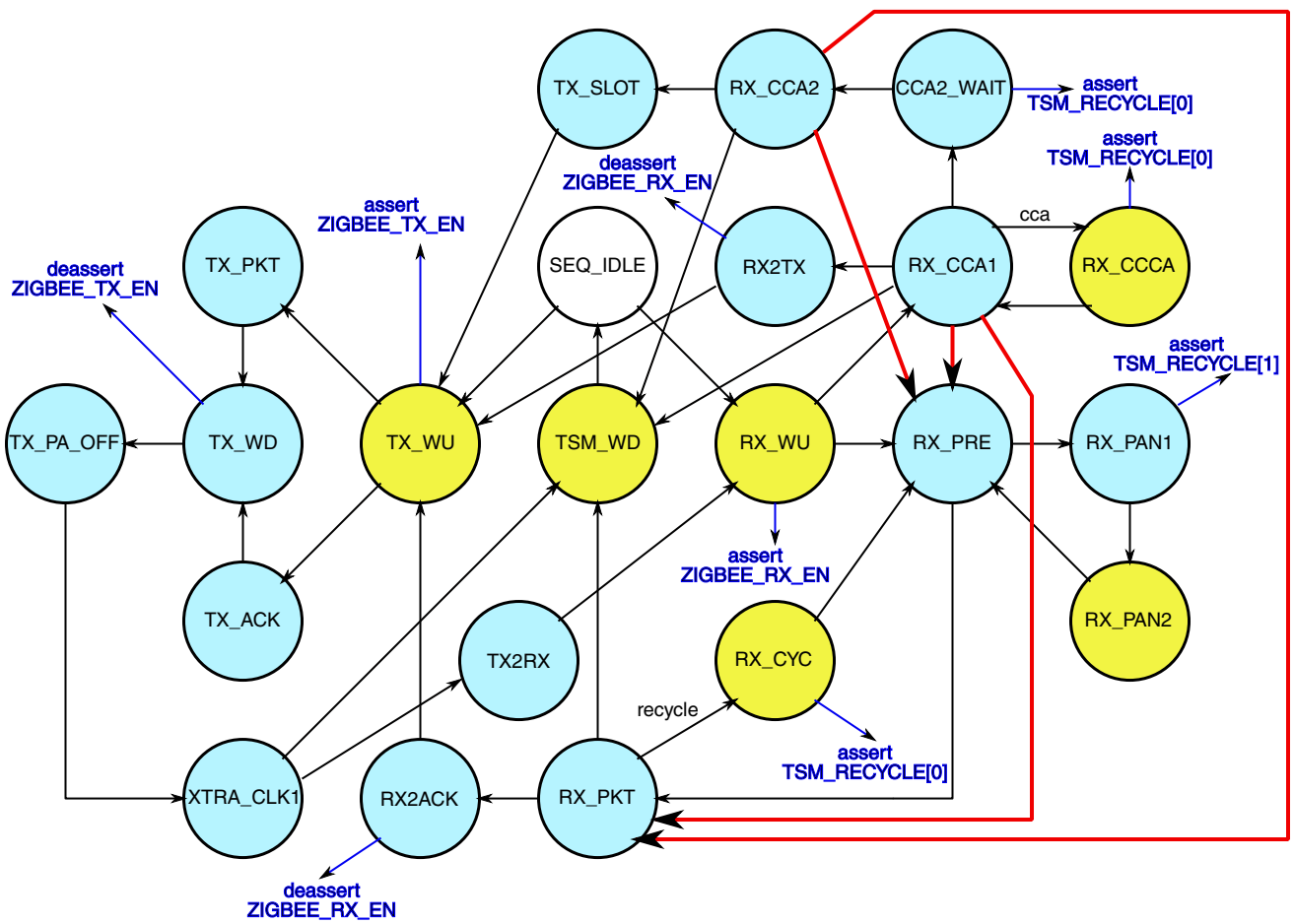



**Figure 64-152. NEW ZSM STATE TRANSITIONS FOR SIMUL CCA/RX**

False preambles may occur during the CCA states as well. False preambles occur when the 802.15.4 demodulator detects a sufficient number of consecutive 0 symbols to meet the preamble threshold, but the symbol which follows is neither a '0' nor a '7'. False preambles result in the 802.15.4 demodulator signal sfd_rst asserting either during, or after, the CCA state. If a false preamble is declared *within* the 128µs CCA window, then the fact that there was a false preamble has no bearing on the channel status, the CCA measurement determines channel status, as usual. If the false preamble is declared *after* the CCA window, the channel is assessed to be busy and the autosequence terminates without transmission, because the scheduled transmission time has been missed.

In defining the Simultaneous CCA/RX operation, 9 "scenarios", or cases, were developed to help ensure correct harwdare response to various permutations of preamble and SFD arrival times, preamble falsing, channel status, and RX packet goodness. These scenarios were used to guide the hardware design, for each of the 3 CCA modes, and for mode 3, both boolean combinations. The 9 scenarios are listed in the table below, with the correct hardware response indicated. The hardware response consists of a sequence of state transitions, as well as assertion (or non-assertion) of the various 802.15.4 interrupts. The final combination of asserted interrupts assists software in determining what transpired during the autosequence, and therefore what must be done next.

| SCENARIO | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 |
|---|---|---|---|---|---|---|---|---|---|
| CHANNEL IDLE @ 128us | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CHANNEL BUSY @ 128us | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PREAMBLE ASSERTED DURING CCA<br>SFD_RST OCCURS DURING CCA<br>CHANNEL OTHERWISE IDLE @ 128us | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PREAMBLE ASSERTED DURING CCA<br>SFD_RST OCCURS DURING CCA<br>CHANNEL OTHERWISE BUSY @ 128us | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PREAMBLE ASSERTED @ 128us<br>SFD NOT ASSERTED @ 128us<br>SFD_RST ASSERTS LATER (FALSE PRE) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PREAMBLE ASSERTED @ 128us<br>SFD NOT ASSERTED @ 128us<br>SFD ASSERTS LATER (BAD PKT) | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| PREAMBLE ASSERTED @ 128us<br>SFD ASSERTED @ 128us (BAD PKT) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| PREAMBLE ASSERTED @ 128us<br>SFD NOT ASSERTED @ 128us<br>SFD ASSERTS LATER (GOOD PKT) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PREAMBLE ASSERTED @ 128us<br>SFD ASSERTED @ 128us (GOOD PKT) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| OUTCOME: | | | | | | | | | |
| AUTOSEQUENCE ENDS AFTER: | TX | CCA | TX | CCA | SFD_RST | RX | RX | RX | RX |
| MODE SWITCH TO SFD SEARCH OCCURS AT: | -- | -- | -- | -- | 128us | 128us | -- | 128us | -- |
| MODE SWITCH TO PKT RECEPTION OCCURS AT: | -- | -- | -- | -- | -- | -- | SFD DET | -- | SFD DET |
| CCAIRQ? | yes (CCA=0) | yes (CCA=1) | yes (CCA=0) | yes (CCA=1) | yes (CCA=1) | yes (CCA=1) | yes (CCA=1) | yes (CCA=1) | yes (CCA=1) |
| RXIRQ? | no | no | no | no | no | no | no | yes | yes |
| TXIRQ? | yes | no | yes | no | no | no | no | no | no |

**Figure 64-153. SIMUL CCA/RX SCENARIOS**

A single register bit enables Simultaneous CCA/RX mode: SIMUL_CCA_RX. This bit resides in the CCA_CTRL register in 802.15.4 address space. By default SIMUL_CCA_RX=0, meaning that the new ZSM transitions are inhibited. Set this bit to 1, to enable the new transitions. Simultaneous CCA/RX works with any CCA Mode (1, 2, or 3), and works for both non-slotted (single CCA before TX) and slotted (2 CCA before TX) modes.

Note that to use the Simultaneous CCA/RX feature, software (not just hardware) needs to anticipate packet reception during a Sequence T (not just a Sequeunce TR). This may not exist today, since prior to Gen3, the 802.15.4 controller did not have this capability.

## 3. New TSM Fast RX-to-TX Capability

The 802.15.4 Sequence Manager (ZSM), makes use of the radio's all-protocol Transceiver Sequence Manager (TSM) to execute TX and RX warmup and warmdown sequences; TSM handles low-level commanding, with precise, programmable timing, of the analog and digital blocks in the transmit and receive chains, absolving the higher-level sequence managers from this responsibibility. TSM sequences have, in the past, been discrete TX or RX operations, with a required software command to initiate the sequence, and a return-to-idle after the sequence is complete, before a new sequence-start command can be issued. For 802.15.4 Simultaneous CCA/RX mode, it is highly desirable to be able to jump directly from the end of a CCA measurement, which is essentially an RX operation, to a packet transmit operation, without returning the TSM to IDLE in between, if CCA deems the channel to be free. A new TSM transition called Fast RX2TX accomplishes this. The 802.15.4 ZSM will signal to TSM when to perform such a transition, and TSM will respond by jumping from the RX ON phase, to a programmable point within the TX warmup. See Chapter *Transceiver Sequence Manager*, Section *FAST TSM Warmup and FAST RX2TX Transition* for more details on this TSM feature, and a description of the ZSM -> TSM signalling required to execute such a transition.

A single register bit enables TSM Fast RX-to-TX: FAST_RX2TX_EN. This bit resides in the TSM_FAST_CTRL1 register in XCVR address space. By default FAST_RX2TX_EN=0, meaning that this TSM capability is inhibited, and RX-to-TX transitions remain discrete RX and TX sequences (legacy).

Once enabled with FAST_RX2TX_EN=1, the ZSM will request a TSM Fast RX-to-TX warmup, by asserting fast_rx2tx_wu, when all of the the following conditions are met:

- 802.15.4 autosequence is T or TR
- Pre-TX CCA is enabled with CCABFRTX=1
- ZSM state is RX_CCA1 (non-slotted or slotted) or RX_CCA2 (slotted mode only)
- 128µs measurement timeout expires
- Neither preamble-detect nor SFD-detect are *currently* asserted
- Channel is deemed idle (CCA=0)

The TSM will respond to fast_rx2tx_wu by jumping from the "ON" phase of the RX operation to the "WARMUP" phase of the TX operation at the programmed point. Simultaneously, ZSM must jump to its TX_WU state to wait for the TX warmup to complete. This requires 2 new state transitions, to enable the Simultaneous CCA/RX feature, shown in red in the diagram below:

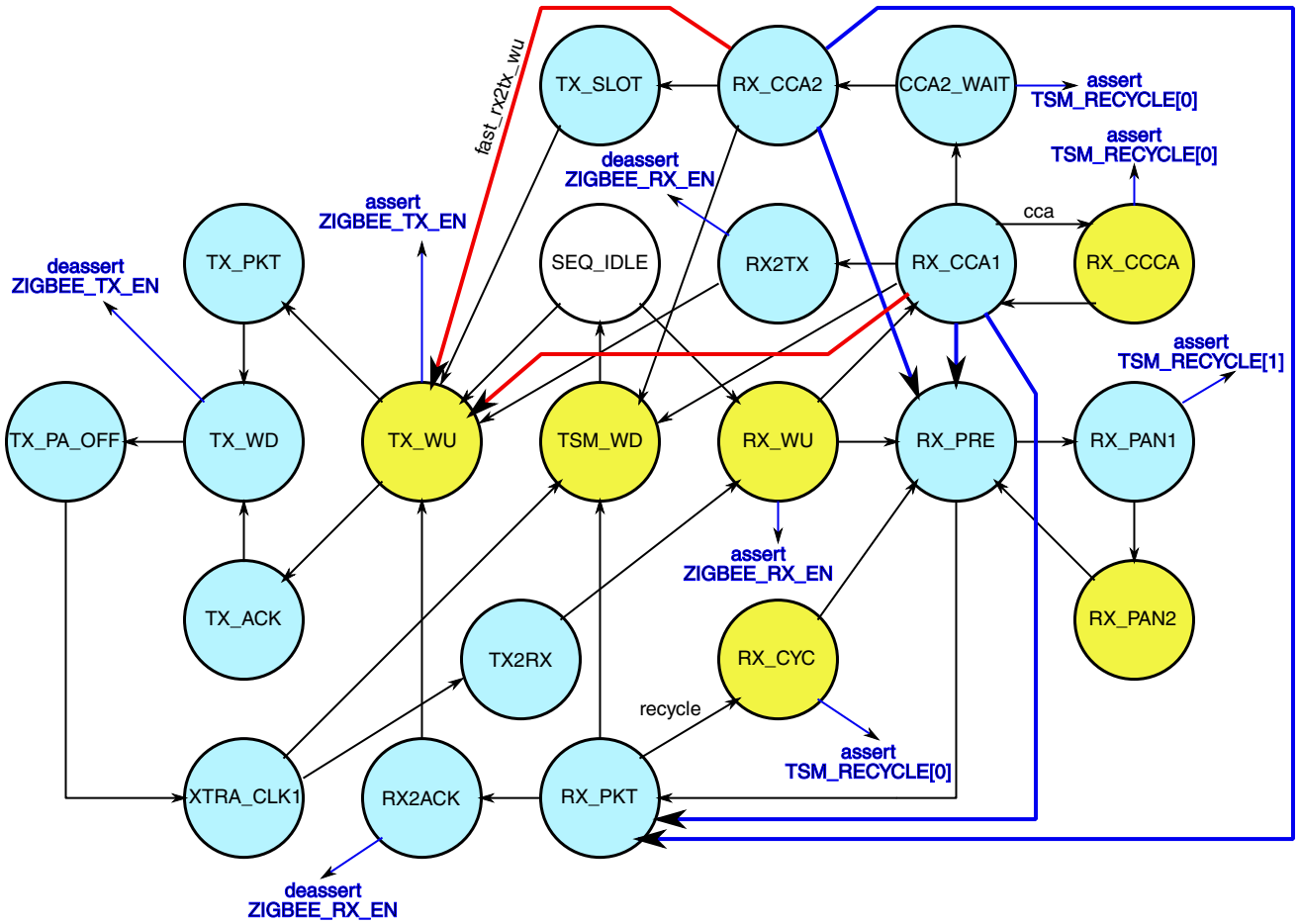



**Figure 64-154. ZSM STATE TRANSITIONS FOR FAST RX2TX**

Once the TX warmup completes, the 802.15.4 controller will transmit the packet as per the Sequence T command.

### 64.5.2.4.7 Manual DSM

When the 802.15.4 Link Layer anticipates long periods of inactivity, the Link Layer controller can be put into a Deep Sleep Mode (DSM), where all of its clocks are turned off. In addition, the 802.15.4 Link Layer can be configured to optionally turn off the RF Oscillator, if it is not otherwise needed by the SoC. Deep Sleep Mode results in dramatic power savings. The procedure described below, is called Manual DSM.

To make Manual DSM possible, the transceiver incorporates a low-frequency, high-precision Deep Sleep Timer, DSM_TIMER. This timer is clocked by a crystal-referenced 32.768KHz oscillator. The timer is 24-bit wide, yielding a 8.5 minute rollover (the maximum length of a deep sleep period).

The hardware to implement DSM within the SoC, is partitioned across several modules, and several power domains. This is to allow the maximum achievable power savings, by retaining full voltage for only those modules which need to be active in DSM, and placing the remaining modules in “state retention” mode. The diagram below depicts the partitioning of DSM hardware across the various modules, including the 802.15.4 Link Layer Controller.

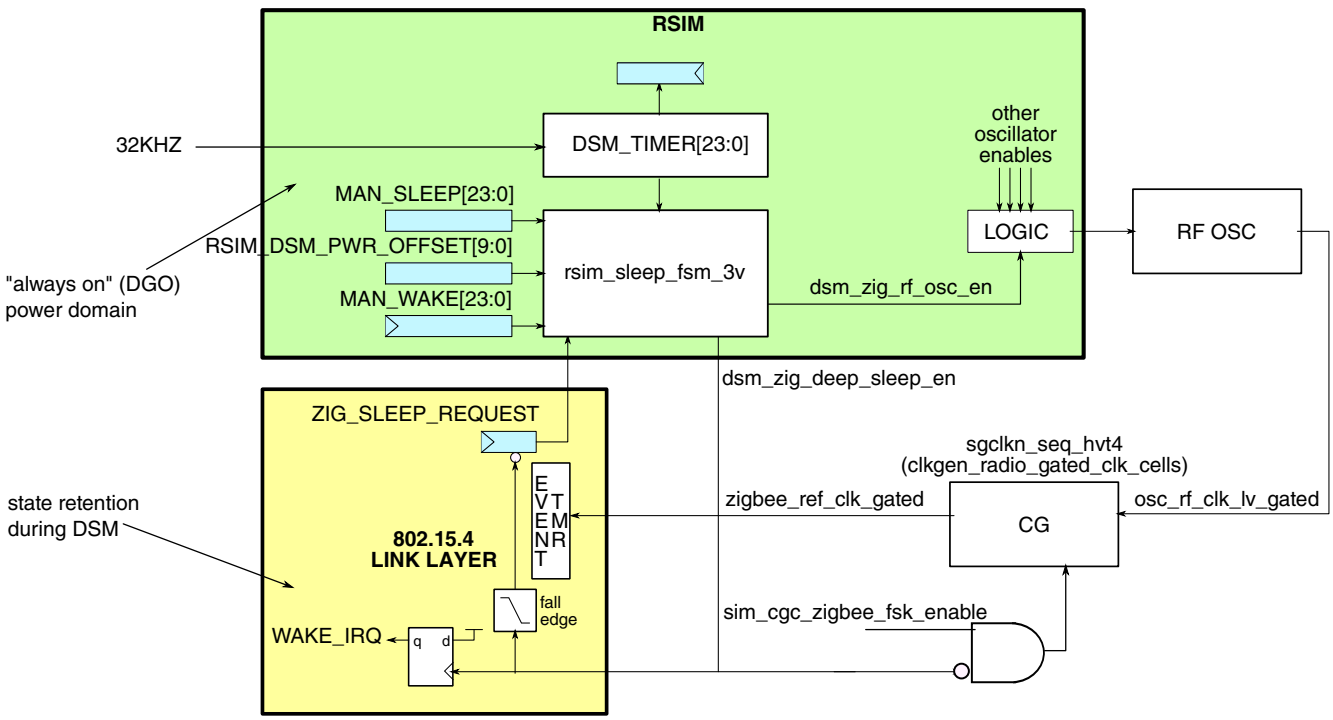



**Figure 64-155. MANUAL DSM BLOCK DIAGRAM**

The concept behind Manual DSM, as it is implemented for 802.15.4, is to allow the 802.15.4 timebase to be maintained for a period of time, while the RF Oscillator is turned off and much of the SoC is placed in state-retention. During DSM, the timebase management is temporarily transferred from the 802.15.4 EVENT_TMR to the low-power DSM_TIMER, and after DSM exit, timebase management is transferred back to the EVENT_TMR, under software control. During DSM, the EVENT_TMR is frozen. The total time spent in DSM is known to software, so that upon exiting DSM, the EVENT_TMR can be updated to correct for the precise amount of time it was frozen.

DSM entry and exit, and RF Oscillator re-start during Manual DSM, are governed by 5 registers, as described in the table below.

| Field | R/W | Description |
|---|---|---|
| MAN_SLEEP[23:0] | rw | If ZIG_SLEEP_REQUEST=1, enter DSM & freeze EVENT_TMR when MAN_SLEEP matches DSM_TIMER. |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | **NOTE:** This register resides in RSIM Address Space |
| RSIM_DSM_PWR_OFFSET[9:0] | rw | Determines the power and RF oscillator re-start times relative to MAN_WAKE<br>**NOTE:** This register resides in RSIM Address Space |
| MAN_WAKE[23:0] | rw | Exit Deep Sleep Mode, and resume EVENT_TMR, when MAN_WAKE matches DSM_TIMER<br>**NOTE:** This register resides in RSIM Address Space |
| ZIG_SLEEP_REQUEST | w | Enable a match on MAN_SLEEP[23:0] to DSM_TIMER[23:0], to enter Deep Sleep Mode, by writing a 1 to this bit. Cancel a pending sleep request by writing a 0 to it, if already set, at least 2 32.768KHz clock cycles prior to MAN_SLEEP[23:0] |
| DSM_TIMER[23:0] | r | Current State of the 32KHz Sleep Timer<br>**NOTE:** This register resides in RSIM Address Space |

## Procedure

The procedure to schedule a Manual DSM cycle (entry/exit) as well as to program the desired RF Oscillator re-start time, and the calculation to restore the 802.15.4 EVENT_TMR after a DSM cycle, is described below.

1. Software determines the future time at which it would like to enter DSM, by reading the DSM_TIMER, computing the number of 32KHz clock cycles remaining until the desired DSM start-time, and writing this value into MAN_SLEEP register. The value programmed into MAN_SLEEP should be no fewer than 4 clocks greater (in the future) than the current time as read from DSM_TIMER.
2. Software determines the future time at which it would like to exit DSM, by computing the number of 32KHz clock cycles remaining until the desired DSM exit-time, and writing this value into MAN_WAKE register.
3. Software writes a 1 to DSM_CTRL[ZIG_SLEEP_REQUEST]. This bit resides in 802.15.4 address space. This bit is write-only and always reads back 0. Writing this bit to 1 enables a DSM cycle to commence using the values programmed into MAN_SLEEP and MAN_WAKE.
4. MCU can go into a low power state (e.g., STOP, VLPS, or VLLSx)
5. When DSM_TIMER = MAN_SLEEP, the EVENT_TMR will be clock-gated so that it will remain frozen at its current count. The RF Oscillator will be turned off (unless overridden due to a non-802.15.4-related SoC requirement)

6. All hardware not in the low-voltage domain will be placed into state-retention, or other low-power state.
7. When DSM_TIMER = MAN_WAKE, clocking of the 802.15.4 Link Layer Controller will resume and the EVENT_TMR will be ungated (allowed to resume counting).
8. Software computes the time that the SoC was in DSM (and hence the time the EVENT_TMR was frozen), in microseconds, with the equation:
   **(MAN_WAKE – MAN_SLEEP) / 32768 * 1000000**
9. Software increments the 802.15.4 EVENT_TMR by this amount, by writing this value to the EVENT_TMR register with EVENT_TMR_ADD=1. **Note:** The 802.15.4 EVENT_TMR has an integer component (24 bits) and a fractional component (4 bits). The fractional component provides the microsecond precision to the Event Timer, so both components need to be accounted for when compensating for the DSM "time slept".
10. On the next 1MHz clock edge, the EVENT_TMR has been restored to where it would have been, had no DSM occurred, with 1.0-microsecond accuracy.

Note that the 1.0-microsecond accuracy quoted above, assumes an ideal 32.768KHz source. Any ppm error in the 32.768KHz source will degrade the microsecond accuracy proportionally to the magnitude of the error.

Once ZIG_SLEEP_REQUEST has been set to 1 to request Manual DSM, the request can be cancelled by writing a 0 to the bit. The cancellation must be performed at least 2 32.768KHz before the MAN_SLEEP[23:0] match to DSM_TIMER[23:0] to avoid a race condition to enter DSM.

The following timing diagram depicts an 802.15.4-driven DSM entry/exit cycle.

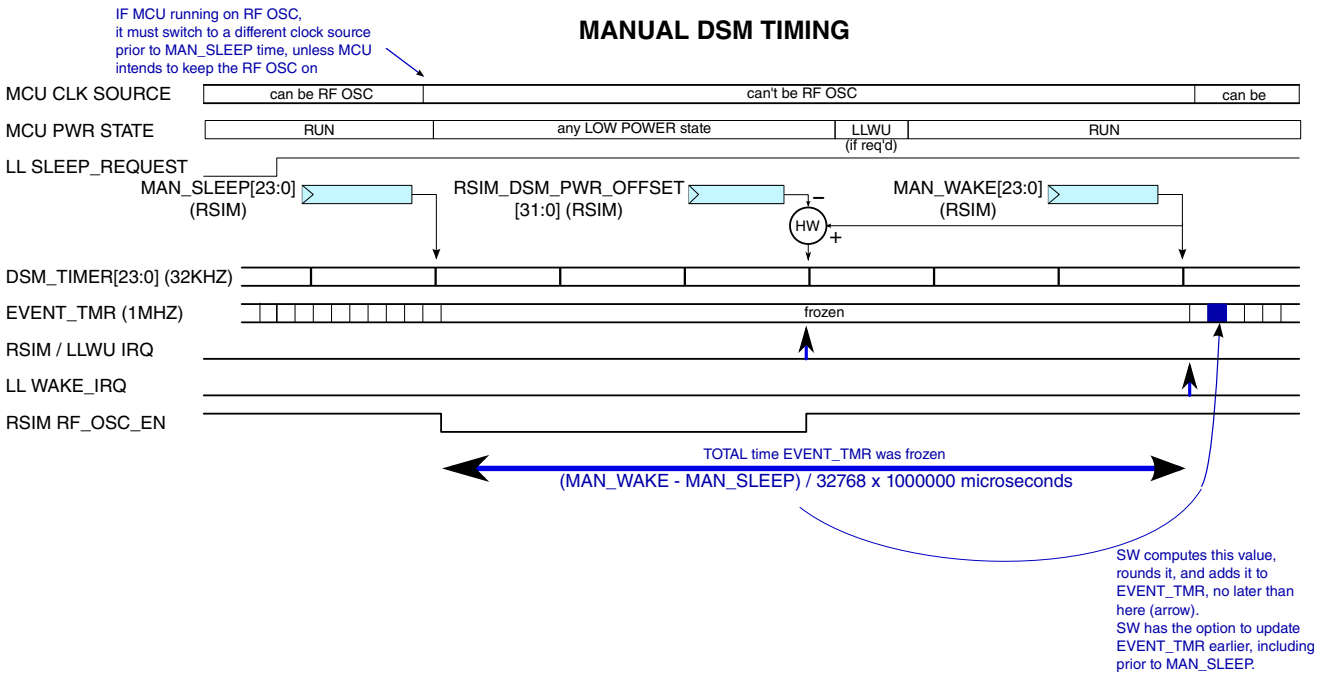



**Figure 64-156. 802.15.4 DEEP SLEEP TIMING**

## DSM Early Exit

Circumstances may arise where the Manual DSM cycle may occasionally need to be cut short, due to circumstances beyond the control of the radio. When this happens, the SoC, starting with the MCU, is awakened out of its low power state, long before the RSIM DSM_TIMER reaches MAN_WAKE. Therefore, software handling the event which triggered the wakeup, will determine if the radio needs to be awakened as well. If this is the case, software will advance the MAN_WAKE setting to a point in the near future, but no fewer than 4 32KHz clock cycles beyond the current time ascertained by reading DSM_TIMER. So, to execute a Manual DSM Early Exit, software shall read DSM_TIMER, add +4 to the value, and write the sum to the RSIM MAN_WAKE register. Software shall then wait for the 802.15.4 WAKE_IRQ interrupt, which will assert once the full radio wakeup from the low power state has completed. Software should then take whatever steps are necessary to configure the radio to respond to the conditions which triggered the early exit. Servicing of a DSM early exit interrupt is beyond the scope of this document.

### 64.5.2.4.8 Wake-On-Radio

The 802.15.4 Link Layer can take advantage of the automated sequence- and DSM-scheduling provided by Wake-On-Radio. Wake-On-Radio is one of two DSM control schemes that can be selected for 802.15.4, as an alternative to Manual DSM. See Chapter *Wake-On-Radio*.

Several changes to the 802.15.4 Link Layer Controller have been made to support Wake-On-Radio.

Interrupt WAKE_IRQ has been made dual-purpose. If Wake-On-Radio is disabled, i.e., WOR_CTRL[WOR_EN]=0, or Wake-On-Radio not assigned to 802.15.4, WAKE_IRQ performs its legacy function for Manual DSM. (See Section Manual DSM). If Wake-On-Radio is enabled, i.e., WOR_EN=1, and assigned to 802.15.4, this interrupt becomes WOR_IRQ, and has new functionality specific to Wake-On-Radio. See Chapter *Wake-On-Radio*, Section *Interrupt Management*, for more details on the meaning, and handling of WOR_IRQ.

In addition to WOR_IRQ, numerous interface changes have been made to support Wake-On-Radio. The changes allow, for example, WOR to manage 802.15.4's RF interrupts, inhibit timestamp updates, etc. The added port signals are listed in the table below, along with a description or each:

| WAKE-ON-RADIO INTERFACE SIGNAL | DIRECTION | DESCRIPTION |
|---|---|---|
| wor_event_tmr_add[27:0] | WOR -> 802.15.4 | The Event Timer "Addback" value computed by WOR to compensate for the amount of time Event Timer was frozen during DSM |
| wor_addback | WOR -> 802.15.4 | A 1µs-long pulse commanding the 802.15.4 Event Timer to auto-increment by the the amount appearing on wor_event_tmr_add[27:0] |
| wor_force_wake_irq | WOR -> 802.15.4 | Force WOR_IRQ to assert, due to an occurrence of one of the 3 WOR_IRQ interrupt triggers |
| wor_block_wake_irq | WOR -> 802.15.4 | Inhibit the legacy WAKE_IRQ assertion when WOR_EN=1 (this interrupt becomes WOR_IRQ) |
| tx_start_ext | WOR -> 802.15.4 | A 1µs-long pulse to the 802.15.4 command decoder to initiate a WOR-driven TX sequence |
| rx_start_ext | WOR -> 802.15.4 | A 1µs-long pulse to the 802.15.4 command decoder to initiate a WOR-driven RX sequence |
| seq_stop_ext | WOR -> 802.15.4 | A 1µs-long pulse to the 802.15.4 command decoder to stop (abort) a WOR-driven RX sequence |
| wor_allow_irq | WOR -> 802.15.4 | Inhibits RF interrupts, so that interrupt assertions can be managed based on WOR descriptor settings. Controlled dynamically by the WOR state machine. |
| timestamp_inhibit | WOR -> 802.15.4 | In Wake-On-Radio, the legacy 802.15.4 TIMESTAMP register becomes WOR TIMESTAMP0 (timestamp for SLOT0). |

*Table continues on the next page...*

| WAKE-ON-RADIO INTERFACE SIGNAL | DIRECTION | DESCRIPTION |
|---|---|---|
| | | This signal blocks this register from updating on SLOTS 1-3 |
| wake_irq_trig | WOR <- 802.15.4 | WOR uses this signal to cancel ZIG_SLEEP_REQUEST and transition from IN_DSM -> COMP_ADDBACK state |
| event_tmr[27:0] | WOR <- 802.15.4 | 802.15.4 Event Timer, used by WOR for timestamp capture |
| event_tmr_plus_1[27:0] | WOR <- 802.15.4 | 802.15.4 Event Timer + 1, used by WOR to schedule RF operation starts and stops |
| tx_irq_trig | WOR <- 802.15.4 | TXIRQ trigger, indicates the completion of a packet transmission, used by WOR to transition out of WAKE_SEQ_END state |
| rx_irq_trig | WOR <- 802.15.4 | RXIRQ trigger, indicates the completion of a good packet reception, used by WOR to transition out of WAKE_SEQ_END state |
| rx_status_for_wor[2:0] | WOR <- 802.15.4 | Internal RX status signals used by WOR to formulate TIMESTAMPx_STS[2:0] |
| timestamp_clk_en | WOR <- CGM <- 802.15.4 | Timestamp clock gate enable, the resultant clock is used by WOR to update the TIMESTAMP1,2, and 3 registers |

## 64.5.2.4.9 CCA/ED/LQI

### 64.5.2.4.9.1 Introduction

CCA/ED/LQI block resides inside 802.15.4 demodulator top. The purpose of this block is to perform the following operations: 1) Clear Channel Assessment: Modes 1, 2 and 3, 2) Energy Detection and 3) Link Quality Indicator.

This block takes RSSI value, as an input, for computations related to CCA modes 1 and 3, Energy Detection and LQI. For CCA2, this block takes output of 802.15.4 correlators for detecting an 802.15.4 signal in the channel of interest.

At the output, it provides a signal called (cca) to indicate if the channel of interest is busy (cca=1) or idle (cca=0), the final computed energy value (cca1_ed_fnl) and Link Quality Indicator (LQI) value. The cca signal is further used to generate cca interrupt, according to the set interrupt mask. The cca1_ed_fnl is a signed 8 bit value that can be read back as a register. LQI is an unsigned 8 bit value that is written at the end of each valid packet that is received and is available to be read in the packet buffer.

#### 64.5.2.4.9.1.1 Block diagram

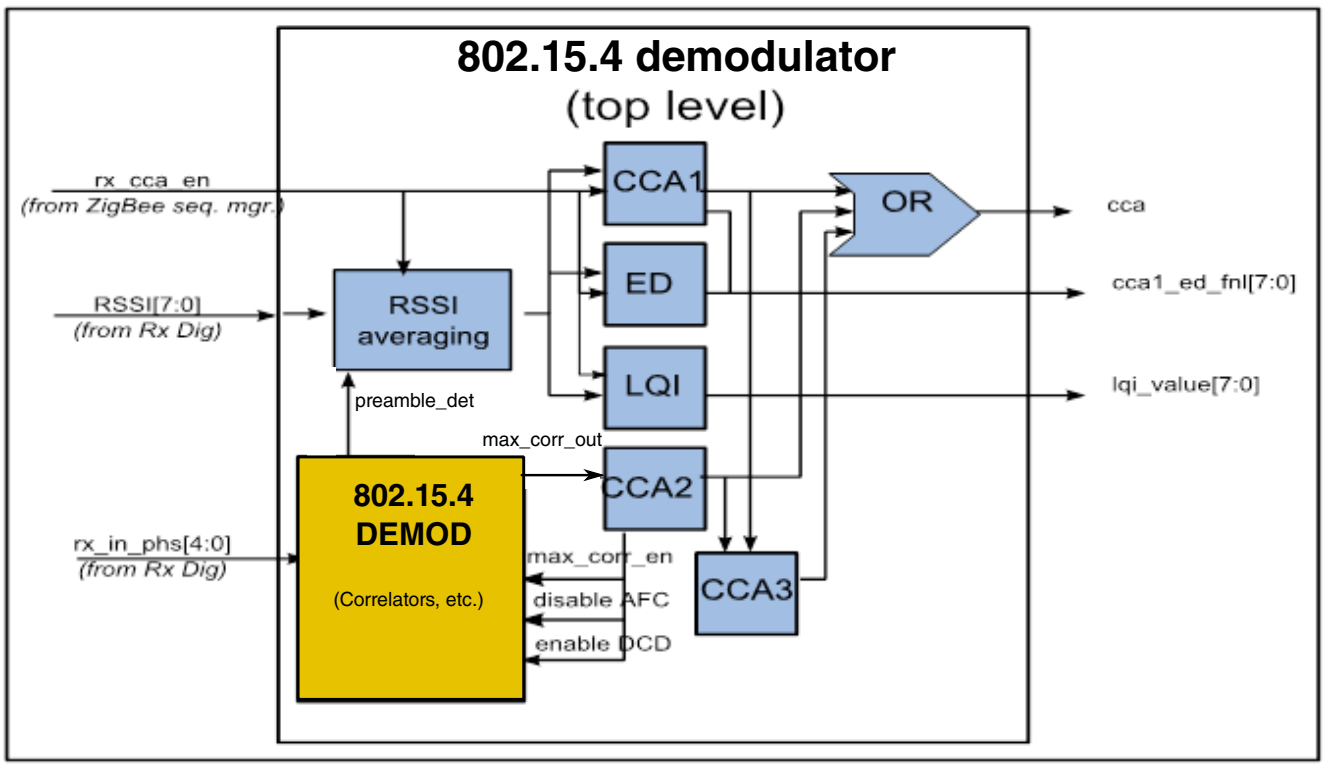



**Figure 64-157. Block diagram**

### 64.5.2.4.9.2 Memory Map and register definition

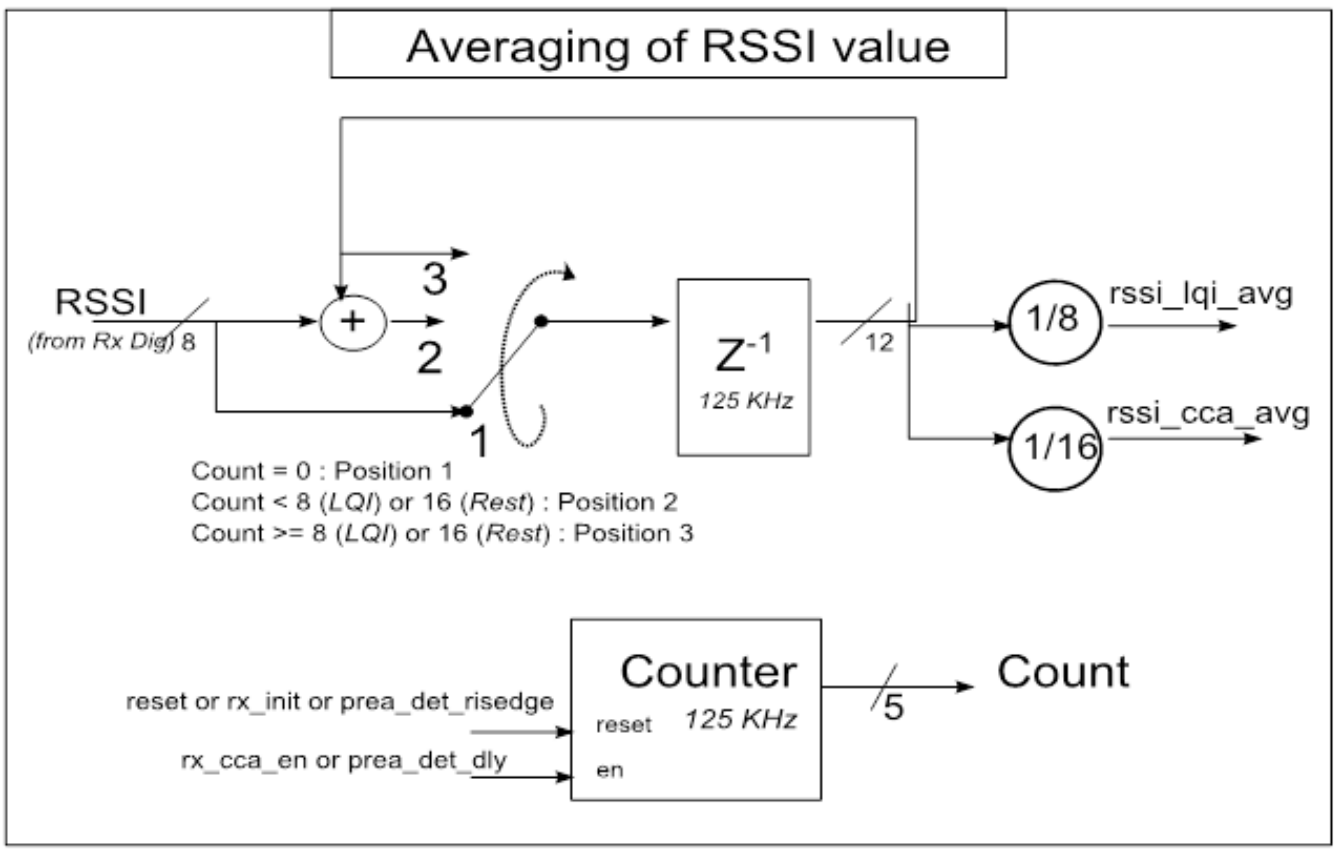
| Field | R/W | Description |
|---|---|---|
| LQI_VALUE[7:0] | r | 8-bit LQI value of each Rx packet |
| CCA1_ED_FNL[7:0] | r | 8-bit signed value to indicate last energy detected |
| CCA1_THRESH[7:0] | rw | 8-bit signed value to be compared against averaged signed RSSI value from Rx DIG block, to determine channel busy or idle |
| LQI_OFFSET_COMP[7:0] | rw | 8-bit value to offset average of signed RSSI value coming from Rx DIG, in order to compute 8-bit LQI value. |
| CCA3_AND_NOT_OR | rw | This register bit controls how CCA1 and CCA2 are combined to assert CCA3 mode:<br>1 : CCA1 AND CCA2<br>0: CCA1 OR CCA2 |
| CCA2_MIN_NUM_CORR_THRESH[2:0] | rw | 3-bit value to compare against number of correlation peaks within CCA2 time. |
| CCA2_NUM_CORR_PEAKS[3:0] | r | 4-bit value indicating number of correlation peaks detected in last CCA2 run |
| CCA2_CORR_THRESH[7:0] | rw | 8-bit value to be compared against correlation peak value during CCA2 operation. |
| LQI_START_AT_SFD | rw | Defines starting point for LQI computation during packet reception |

| Field | R/W | Description |
|---|---|---|
| | | 1: Begin computing LQI at SFD detect (new for KW41)<br>0: Begin computing LQI at preamble detect (legacy) |

### 64.5.2.4.9.3 Functional description

The following sections describe functional details of the module.

#### *64.5.2.4.9.3.1 RSSI Averaging*




The above picture shows how RSSI is averaged. 8-bit signed RSSI value from Rx Dig block is averaged for 128 us for CCA1/2/3/ED sequences. For LQI, it is averaged for 64 us. In both cases, the RSSI value is sampled at 125 KHz rate.

#### *64.5.2.4.9.3.2 CCA1*

When the 802.15.4 link layer is set up to perform CCA1 operation, rx_cca_en signal is asserted at the end of Rx warm up by the 802.15.4 Sequence Manager (ZSM). After rx_cca_en is asserted high, at the end of 128 us, i.e. count = 16, rssi_cca_avg is compared against 8-bit signed register cca1_thresh to assess CCA as follows:

If rssi_cca_avg >= cca1_thresh, CCA1 = 1, i.e. channel is busy.

Else CCA1 = 0, i.e. channel is idle.

#### *64.5.2.4.9.3.3 CCA2*

When the 802.15.4 link layer is set up to perform CCA2 operation, rx_cca_en signal is asserted at the end of Rx warm up by the 802.15.4 Sequence Manager (ZSM). During CCA2 operation, differential chip detection is enabled and automatic frequency correction is disabled, in order to receive correlation peaks related to all available 16 802.1.5.4 symbols. The signal max_corr_en is held high, in order to refresh maximum correlator value after each symbol period. 128 us after rx_cca_en is asserted high, i.e. count = 16, the number of maximum correlation peaks are counted which exceeded the value cca2_corr_thresh register. If the number of correlation peaks exceeds minimum desired number of correlation peaks (programmable register: cca2_num_corr_peaks), cca2 is asserted high to indicate the channel is busy and low otherwise, to indicate channel is idle.

#### *64.5.2.4.9.3.4 CCA3*

CCA mode 3 is a combination of CCA1 and CCA2. There are 2 combinations in which CCA mode 3 operates, based on the register setting of CCA3_AND_NOT_OR: 1) CCA1 OR CCA2 2) CCA1 AND CCA2. When the 802.15.4 link layer is set to perform CCA3, both CCA1 and CCA2 operations, as mentioned above and decides if the channel is busy or idle, according to the desired combination (i.e. AND/OR).

#### *64.5.2.4.9.3.5 Energy Detection*

During Energy Detection operation, the demodulator is expected to average RSSI value (from Rx_DIG) for 128 us, as mentioned above in the block diagram. At the end of 128 us, the register CCA1_ED_FNL is updated with this average energy value. Energy detection is similar to CCA1. Unlike CCA1, busy/idle signal is not generated.

#### *64.5.2.4.9.3.6 Link Quality Indicator*

LQI maps RSSI value (from Rx_DIG) to values between 0x00 and 0xFF, as shown in the following block diagram. Upon preamble detect, RSSI value is averaged for 64 us and mapped to LQI value. This latched LQI value is available in the register LQI_VALUE. Packet processor also appends this LQI value at the end of packet buffer.

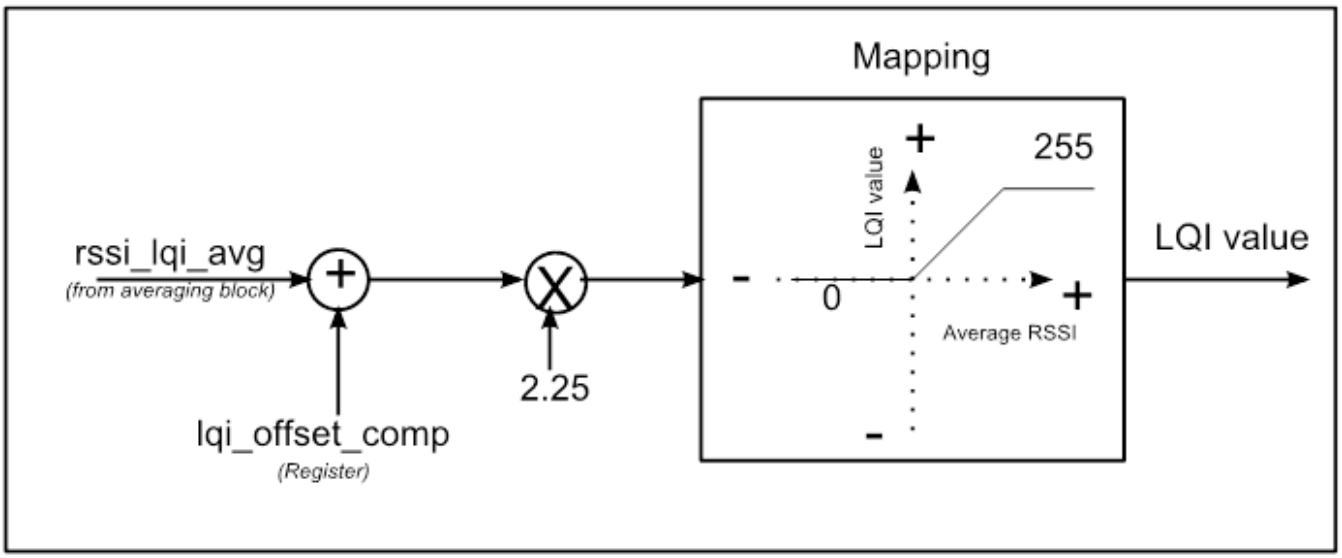



**Figure 64-158. LQI Computation**

The following picture shows an example of mapped LQI, when the register LQI_OFFSET_COMP is programmed to decimal value of 102.

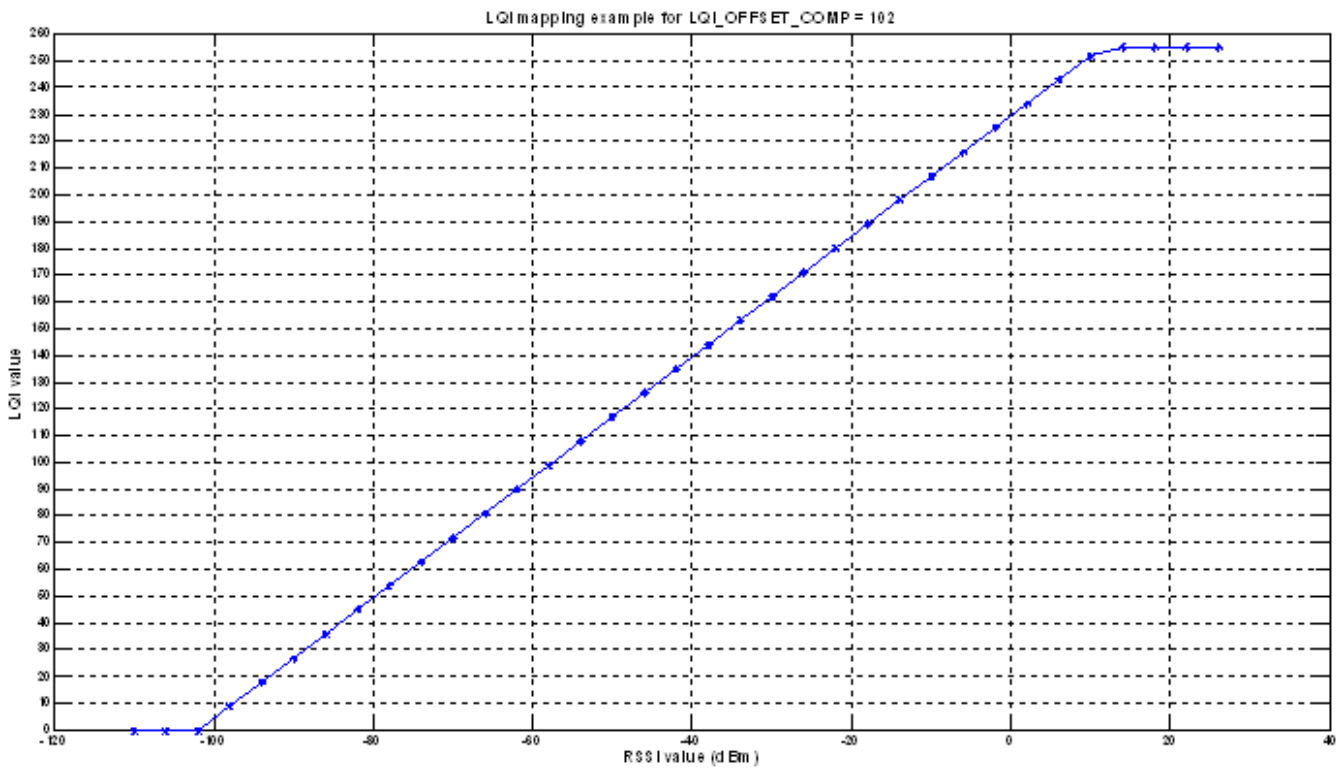



**Figure 64-159. Plot of LQI vs. RSSI**

#### *64.5.2.4.9.3.7 Clocks*

The CCA/ED/LQI module is a fully synchronous module. It has a single clock input **clk32m**. This clock input is driven by the RF Oscillator. The CCA/ED/LQI module has no asynchronous interfaces or clock domain crossings.

#### *64.5.2.4.9.3.8 Reset*

The CCA/ED/LQI module has a single, active-low, asynchronous reset input: **resetb** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

#### *64.5.2.4.9.3.9 Interrupts*

The CCA/ED/LQI module does not generate any interrupts.

### 64.5.2.4.10 Fast Antenna Diversity (FAD)

**FAD ALGORITHM**

The Fast Antenna Diversity (FAD) mode, when enabled, will allow the choice between two antennae to be selected during the preamble search. By continually monitoring the received signal the FAD block will select the first antenna on which the received signal has a correlation factor above a predefined, programmable, threshold. This threshold is register programmable. The FAD accomplishes the antenna selection by sequentially switching between the two antenna testing for the presence of a suitably strong s0 symbol, the first antenna to reach this condition is then selected for the reception of the packet.

The first antenna is monitored for a period equal to 1 symbol, $T_s$ = 16µs, then the antenna monitoring is switched to the second antenna, $T_a$ = 8µs. $T_a$ is required to allow for the external pin diodes to turn on/off to select the antenna. $T_s + T_a$ = 24µs which allows enough time to be able to test both antenna within the first 4 preamble symbols, $T_{fad} = 3 \times T_a + 2 \times T_s$ = 56µs, thus $T_{fad} < 4 \times T_s < 64$µs.

The FAD will continually switch between the two antenna until one is found which a sufficiently strong s0 detection. By virtue of the fact that the FAD operation covers less than four s0 symbols before the antenna is selected still allows the symbol demodulator to then detect at least four s0 symbols before declaring "Preamble Detect".

The symbol correlator runs during the $T_a$ period to calculate the max correlation factor for the antenna Cx. However since $T_c$ is not guaranteed to coincide with 1 unique symbol $T_s$ the correlation must be able to work across symbols. To accomplish this the correlator will rotate the s0 PN chip code through its 32 combinations in order to find the s0 symbol split across two adjacent symbols.

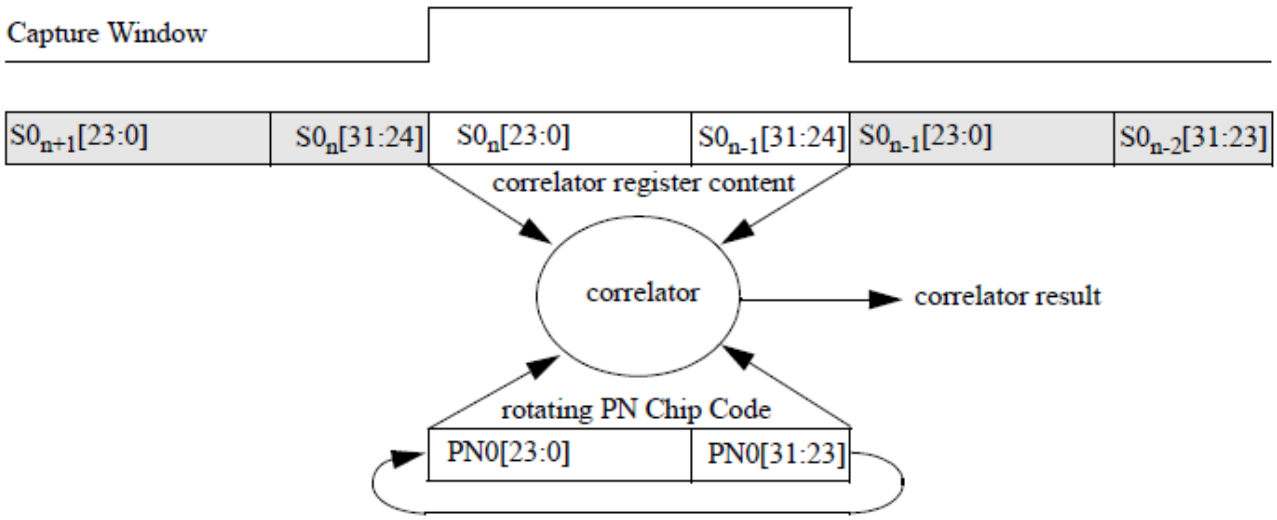



**Figure 64-160. S0 Symbol Search**

When the S0 PN Chip code becomes aligned to the Symbol data within the correlator register a correlation peak is produced that is used to determine the Antenna has a sufficiently strong signal to be selected.

It is not possible to choose between Antenna with the highest correlation factor, since indeed 1 antenna could receive no signal and thus is not distinguishable from a clear channel. This being the case, the first antenna that meets the requirements, correlation factor above the defined threshold, is selected.

When a correlator peak is detected, a second check is done ½ symbol later. A new peak should be detected with the rotation equal to ½ symbols ±2 chips. If it is the case the channel is validated and the classical *CORR_NVAL* consecutive S0 is used to certify a correct preamble, where *CORR_NVAL* is the programmable number of consecutive S0 symbols required to declare preamble detected.

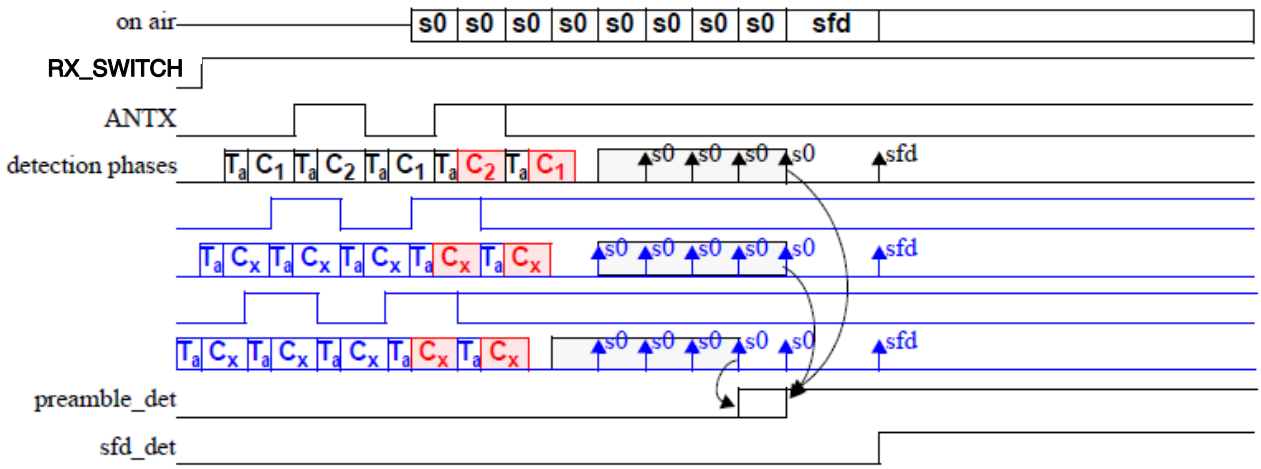
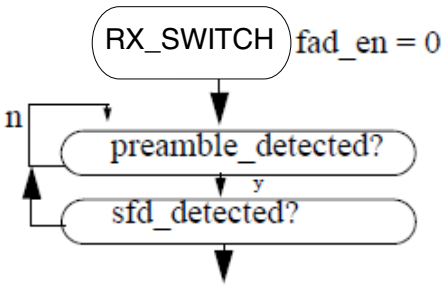
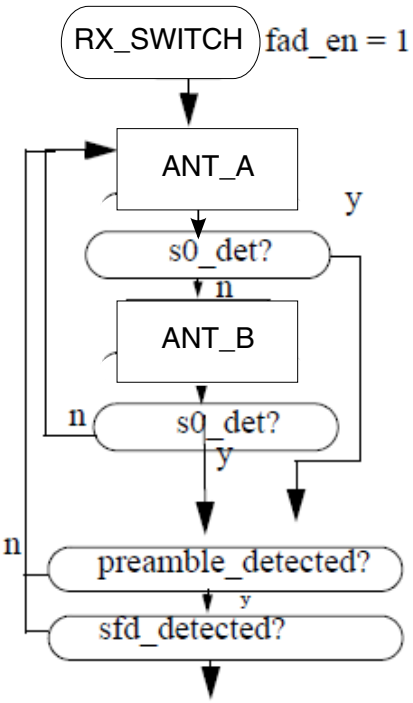



**Figure 64-161. FAD Switching**


RX_SWITCH
fad_en = 0
n
preamble_detected?
y
sfd_detected?


RX_SWITCH
fad_en = 1
ANT_A
y
s0_det?
n
ANT_B
n
s0_det?
y
n
preamble_detected?
y
sfd_detected?

The choice of the antenna is also used for the transmit. Since the antx_out value (the last antenna selected by the FAD algorithm) is kept after the end of the receive, the following TX will use the last best antenna found during the last receive. This is true also for any ensuing CCA or ED RX frame; the antenna selected will be the one selected by the FAD algorithm during the last successful preamble search. So a CCA will search for an empty channel on the best antenna chosen at the last RX. When FAD is enabled (FAD_CTRL[FAD_EN]=1), the state of the antenna can be ascertained at any time by reading the FAD_CTRL[ANTX] bit: 0 = ANT_A, 1 = ANT_B. When FAD_CTRL[FAD_EN]=0, antenna selection is placed under direct software control, and the value written to the FAD_CTRL[ANTX] bit becomes the selected antenna: 0 = ANT_A, 1 = ANT_B. The FAD_CTRL register resides in XCVR space.

**FAD PIN INTERFACE**

This content has been moved to: Chapter *Transceiver Sequence Manager*, Section *Fast Antenna Diversity (FAD)*

## 64.5.2.4.11 Interrupts

### 64.5.2.4.11.1 Introduction

This section describes the 802.15.4 Interrupt Architecture.

| Revision | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 04-28-2014 | D. Brown | Copy-over from Coconino |
| 0.2 | 10-24-2015 | D. Brown | Updated for KW41 |

The 802.15.4 Link Layer has a single, active high, interrupt line (ipi_int_zigbee) to the MCU. Internally, the Link Layer has 13 interrupt sources, as shown in the following table:

| 802.15.4 Interrupt Sources |
|---|
| SEQIRQ |
| TXIRQ |
| RXIRQ |
| CCAIRQ |
| RXWTRMRKIRQ |
| FILTERFAIL_IRQ |
| PLL_UNLOCK_IRQ |
| TMR1IRQ |
| TMR2IRQ |
| TMR3IRQ |

*Table continues on the next page...*

| 802.15.4 Interrupt Sources |
|---|
| TMR4IRQ |
| WAKE_IRQ |
| TSM_IRQ |

Any or all of the interrupt sources, can be enabled to cause an assertion on ipi_int_zigbee. Each interrupt source has its own interrupt status bit in 802.15.4 register space. Each interrupt source is individually maskable (each has its own MASK bit), so that each interrupt source can be individual enabled to trigger an assertion on ipi_int_zigbee. There is also a global interrupt mask, TRCV_MSK, which can enable/disable all ipi_int_zigbee assertions by programming a single masking bit. All interrupt status bits (except TSM_IRQ) use a write-1-to-clear protocol. Interrupt status bits are not affected by reads. The ipi_int_zigbee pin is a level-sensitive interrupt indicator to the MCU; on an interrupt, ipi_int_zigbee will remain asserted until all active interrupt sources are cleared or masked.

#### *64.5.2.4.11.1.1 Features*

- 13 interrupt sources (interrupt status bits)
- each interrupt source individually maskable
- each interrupt source is individually write-1-to-clear.
- interrupts remain asserted until cleared.

### 64.5.2.4.11.2 Memory Map and register definition

Numerous register bits are provided to control 802.15.4 interrupt behaviour. The registers referred to below (IRQSTS1, IRQSTS2, IRQSTS3, PHY_CTRL2, PHY_CTRL3, and PHY_CTRL4) all reside in the 802.15.4 Address space.

| Field | R/W | Description | Default |
|---|---|---|---|
| SEQIRQ | r/w1tc | 1: A Sequencer Interrupt has occurred<br>0: A Sequencer Interrupt has not occurred | - |
| TXIRQ | r/w1tc | 1: A TX Interrupt has occurred<br>0: A TX Interrupt has not occurred | - |
| RXIRQ | r/w1tc | 1: A RX Interrupt has occurred<br>0: A RX Interrupt has not occurred | - |
| CCAIRQ | r/w1tc | 1: A CCA Interrupt has occurred<br>0: A CCA Interrupt has not occurred | - |
| RXWTRMRKIRQ | r/w1tc | 1: A RX Watermark Interrupt has occurred<br>0: A RX Watermark Interrupt has not occurred | - |
| FILTERFAIL_IRQ | r/w1tc | 1: A Filter Fail Interrupt has occurred | - |

*Table continues on the next page...*

| Field | R/W | Description | Default |
|---|---|---|---|
| | | 0: A Filter Fail Interrupt has not occurred | |
| PLL_UNLOCK_IRQ | r/w1tc | 1: A PLL Unlock Interrupt has occurred<br>0: A PLL Unlock Interrupt has not occurred | - |
| TMR1IRQ | r/w1tc | 1: A TMR1 Interrupt has occurred<br>0: A TMR1 Interrupt has not occurred | - |
| TMR2IRQ | r/w1tc | 1: A TMR2 Interrupt has occurred<br>0: A TMR2 Interrupt has not occurred | - |
| TMR3IRQ | r/w1tc | 1: A TMR3 Interrupt has occurred<br>0: A TMR3 Interrupt has not occurred | - |
| TMR4IRQ | r/w1tc | 1: A TMR4 Interrupt has occurred<br>0: A TMR4 Interrupt has not occurred | - |
| WAKE_IRQ | r | 1: A Wake Interrupt has occurred<br>0: A Wake Interrupt has not occurred | - |
| TSM_IRQ | r | 1: A TSM Interrupt has occurred<br>0: A TSM Interrupt has not occurred | - |
| TMR1MSK | rw | 1: Mask TMR1 Interrupt from asserting ipi_int_zigbee<br>0: Enable TMR1 Interrupt to assert ipi_int_zigbee | 1 |
| TMR2MSK | rw | 1: Mask TMR2 Interrupt from asserting ipi_int_zigbee<br>0: Enable TMR2 Interrupt to assert ipi_int_zigbee | 1 |
| TMR3MSK | rw | 1: Mask TMR3 Interrupt from asserting ipi_int_zigbee<br>0: Enable TMR3 Interrupt to assert ipi_int_zigbee | 1 |
| TMR4MSK | rw | 1: Mask TMR4 Interrupt from asserting ipi_int_zigbee<br>0: Enable TMR4 Interrupt to assert ipi_int_zigbee | 1 |
| SEQMSK | rw | 1: Mask Sequencer Interrupt from asserting ipi_int_zigbee<br>0: Enable Sequencer Interrupt to assert ipi_int_zigbee | 1 |
| TXMSK | rw | 1: Mask TX Interrupt from asserting ipi_int_zigbee<br>0: Enable TX Interrupt to assert ipi_int_zigbee | 1 |
| RXMSK | rw | 1: Mask RX Interrupt from asserting ipi_int_zigbee<br>0: Enable RX Interrupt to assert ipi_int_zigbee | 1 |
| CCAMSK | rw | 1: Mask CCA Interrupt from asserting ipi_int_zigbee<br>0: Enable CCA Interrupt to assert ipi_int_zigbee | 1 |
| RX_WMRK_MSK | rw | 1: Mask RX Watermark Interrupt from asserting ipi_int_zigbee<br>0: Enable RX Watermark Interrupt to assert ipi_int_zigbee | 1 |
| FILTERFAIL_MSK | rw | 1: Mask Filter Fail Interrupt from asserting ipi_int_zigbee<br>0: Enable Filter Fail Interrupt to assert ipi_int_zigbee | 1 |
| PLL_UNLOCK_MSK | rw | 1: Mask PLL Unlock Interrupt from asserting ipi_int_zigbee<br>0: Enable PLL Unlock Interrupt to assert ipi_int_zigbee | 1 |
| WAKE_MSK | rw | 1: Mask Wake Interrupt from asserting ipi_int_zigbee<br>0: Enable Wake Interrupt to assert ipi_int_zigbee | 1 |
| TSM_MSK | rw | 1: Mask TSM Interrupt from asserting ipi_int_zigbee | 1 |

*Table continues on the next page...*

| Field | R/W | Description | Default |
|---|---|---|---|
| | | 0: Enable TSM Interrupt to assert ipi_int_zigbee | |
| TRCV_MSK | rw | 1: Mask all Interrupts from asserting ipi_int_zigbee<br>0: Enable any Interrupt to assert ipi_int_zigbee | 0 |
| T1CMP[23:0] | rw | TMR1 compare value. If TMR1CMP_EN=1 and the Event Timer matches this value, TMR1IRQ is set. | 0xFFF |
| T2CMP[23:0] | rw | TMR2 compare value. If TMR2CMP_EN=1 and the Event Timer matches this value, TMR2IRQ is set. | 0xFFF |
| T2PRIMECMP[15:0] | rw | TMR2-prime compare value. If TMR2CMP_EN=1 and TC2PRIME_EN=1 and the lower 16 bits of Event Timer matches this value, TMR2IRQ is set. | 0xFF |
| T3CMP[23:0] | rw | TMR3 compare value. If TMR3CMP_EN=1 and the Event Timer matches this value, TMR3IRQ is set. | 0xFFF |
| T4CMP[23:0] | rw | TMR4 compare value. If TMR4CMP_EN=1 and the Event Timer matches this value, TMR4IRQ is set. | 0xFFF |
| TMR1CMP_EN | rw | 1: Allow an Event Timer Match to T1CMP to set TMR1IRQ<br>0: Don't allow an Event Timer Match to T1CMP to set TMR1IRQ | 0 |
| TMR2CMP_EN | rw | 1: Allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ<br>0: Don't allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ | 0 |
| TMR3CMP_EN | rw | 1: Allow an Event Timer Match to T3CMP to set TMR3IRQ<br>0: Don't allow an Event Timer Match to T3CMP to set TMR3IRQ | 0 |
| TMR4CMP_EN | rw | 1: Allow an Event Timer Match to T4CMP to set TMR4IRQ<br>0: Don't allow an Event Timer Match to T4CMP to set TMR4IRQ | 0 |
| TC2PRIME_EN | rw | 1: Allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ<br>0: Don't allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ | 0 |
| RX_WTR_MARK[7:0] | rw | Receive byte count (octets) needed to trigger a RXWTRMRKIRQ interrupt . A setting of 0 generates an interrupt at end of the Frame Length field (first byte after SFD). A setting of 1 generates an interrupt after the first byte of Frame Control Field, etc. | 0xFF |

## 64.5.2.4.11.3 Functional description

### *64.5.2.4.11.3.1 Interrupt Status Bit Structure*

The 802.15.4 Link Layer has 13 interrupt sources, each represented in the register map by an interrupt status bit. All interrupt status bits share a common, generic structure. If a particular interrupt source is enabled, a "TRIGGERING EVENT" for that interrupt source, will always set the status bit. A write-1-to-clear input from the IPS bus, will clear the status bit, but only if there is not a simultaneous triggering event. The triggering event prevails over a software clear attempt, if both events coincide. The clock gate guarantees that the status bit only sees a clock when either a triggering event occurs, or a write-1-to-

clear pulse arrives from the IPS bus. This reduces interrupt status bit power consumption to an absolute minimum. The common interrupt status bit structure is shown in the following diagram.

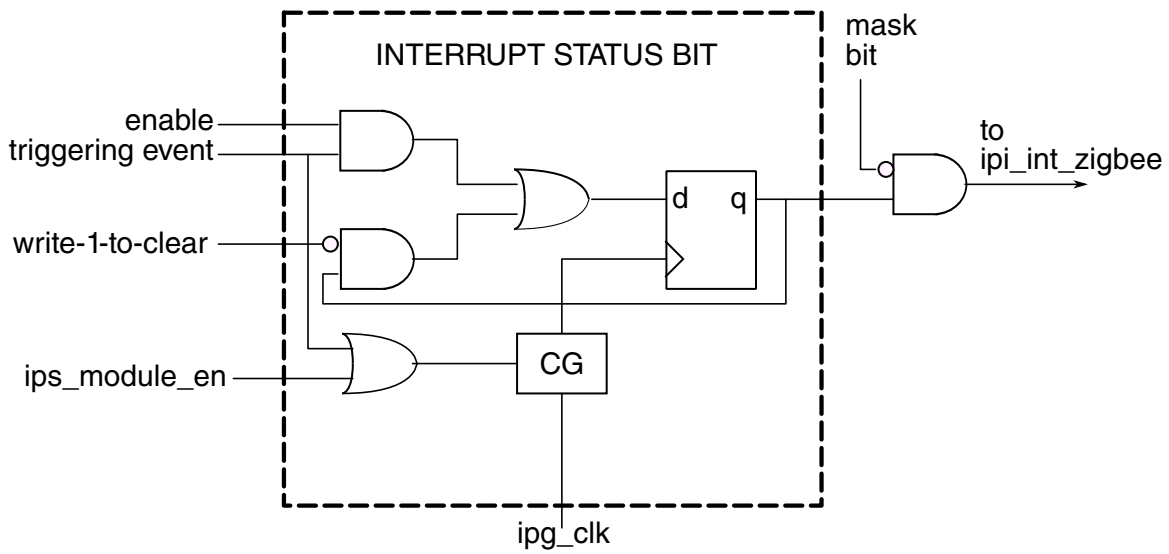



**Figure 64-163. Interrupt Status Bit**

Note that for any 802.15.4 interrupt source, if the triggering event occurs, the Interrupt Status Bit will be set regardless of the state of the corresponding MASK bit. If any of the 13 interrupts is to be ignored, software should set the corresponding MASK bit, and apply the appropriate bit mask when reading the IRQSTS1, IRQSTS2, or IRQSTS3 register, to mask out the unwanted status bit.

#### *64.5.2.4.11.3.2 802.15.4 Interrupt Architecture*

The 13 802.15.4 interrupt sources (status bits) are combined with their individual MASK bits, and then logically OR'ed together, in sum-of-products fashion, to generate single line to the MCU: ipi_int_zigbee. A global mask bit, TRCV_MSK, can enable or disable ipi_int_zigbee altogether. There is no prioritization of interrupt sources, they all have equal weight. The ipi_int_zigbee output is a level-sensitive indicator and will remain asserted until all interrupt status bits are cleared (or the corresponding mask bits set). The following diagram depicts the 802.15.4 Interrupt Architecture.

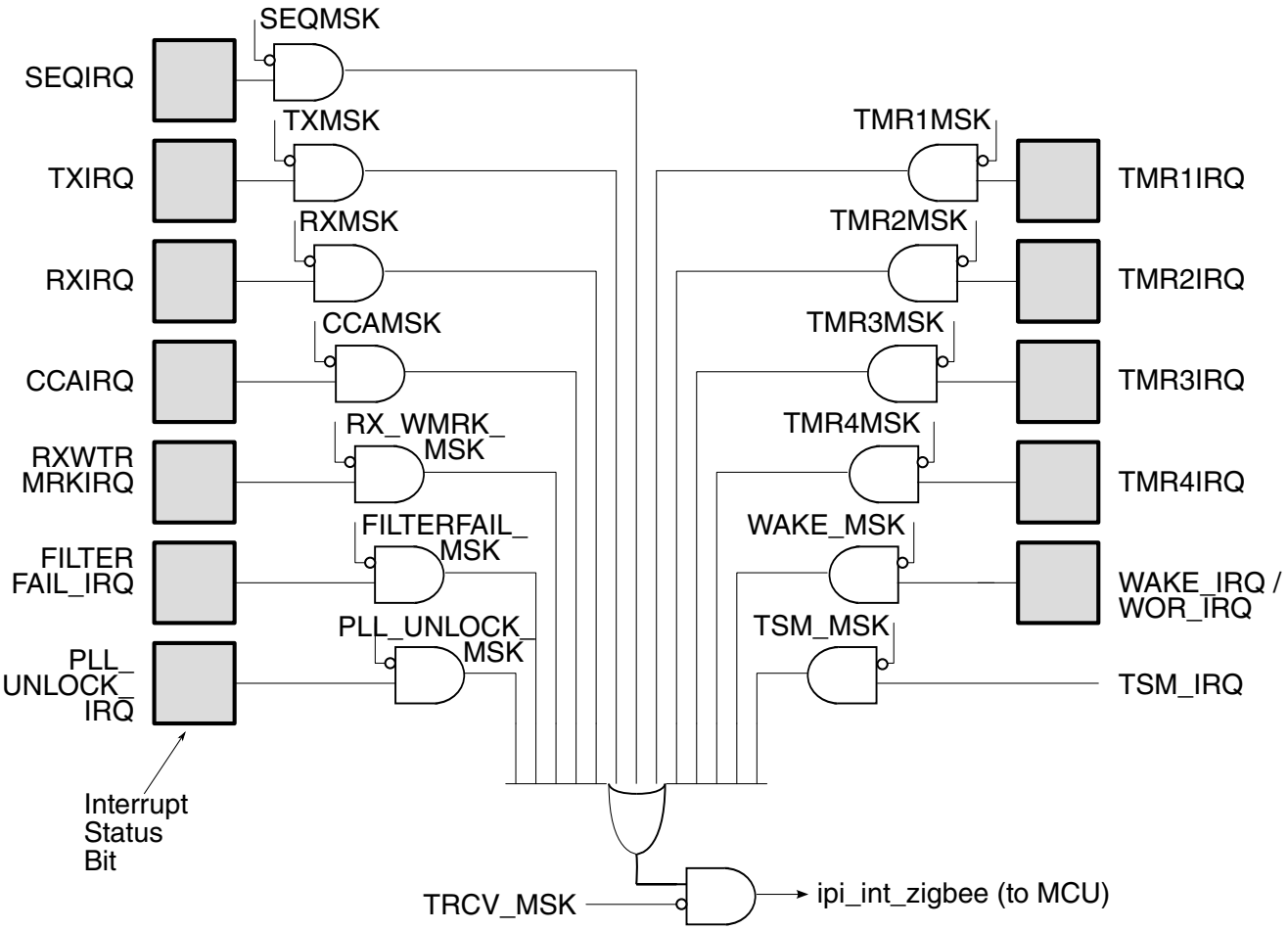



**Figure 64-164. 802.15.4 Interrupt Architecture**

#### *64.5.2.4.11.3.3 Clearing Interrupts*

All interrupt status bit use a write-1-to-clear protocol. Writing a ‘1’ to the interrupt status bit, in the IRQSTS1, IRQSTS2, or IRQSTS3 register, clears the offending interrupt. Writing a ‘0’ to an interrupt status bit has no effect on the bit. Interrupt status bits are not affected by reads.

**Note:** The TSM interrupt is an exception from the interrupt-clearing rules stated above. See Section TSM Interrupt below.

#### *64.5.2.4.11.3.4 Timer Interrupts*

The 802.15.4 Link Layer features a 24-bit Event Timer (see Section Event Timer). The Link Layer has 4 Timer interrupts (TMR1IRQ, TMR2IRQ, TMR3IRQ, and TMR4IRQ), each with its own 24-bit compare register (T1CMP, T2CMP, T3CMP, and T4CMP), and each with its own compare enable (TMR1CMP_EN, TMR2CMP_EN, TMR3CMP_EN, and TMR4CMP_EN). For each timer compare enable, if the bit is set, a match on the

respective 24-bit compare value to the Event Timer will cause the corresponding interrupt status bit to become set. If the compare enable is low, Event Timer matches won't cause the corresponding interrupt status bit to become set.

In addition, a 16-bit T2PRIMECMP compare value is provided, along with a compare-enable bit: TC2PRIME_EN. When TC2PRIME_EN is set (and TMR2CMP_EN is also set), a match on T2PRIMECMP with the *lower 16 bits of Event Timer*, will cause TMR2IRQ to become set, rather than a full 24-bit compare.

#### *64.5.2.4.11.3.5 PLL Unlock Interrupt*

When an PLL unlock event occurs during a 802.15.4 autosequence, the PLL_UNLOCK_IRQ status bit will become set. The ZSM Sequence Manager will begin monitoring for PLL unlock only after a complete transceiver warmup has occurred; unlocks which occur during warmup will not cause a PLL_UNLOCK_IRQ. A PLL unlock which occurs after the warmup period, can be enabled to cause a sequence abort. (See Section Sequence Aborting for more details).

#### *64.5.2.4.11.3.6 Filterfail Interrupt*

Filterfail interrupt will occur during packet reception, when a packet fails filtering rules. On a Filterfail interrupt, the FILTERFAIL_CODE register can be read to ascertain more information about the nature of the filter failure. Filterfail interrupt is not expected to be widely used in mission modes; it has been provided primarily for debug purposes.

#### *64.5.2.4.11.3.7 RX Watermark Interrupt*

RX Watermark interrupt will occur during packet reception, when the number of received bytes matches the contents of the RX_WTR_MARK register, minus 1. For the purpose of defining RX_WTR_MARK, the first byte received is the Frame Length field (PHR). The second byte received is the least-significant byte of Frame Control Field, etc. For example, to cause an RX Watermark Interrupt to occur after the 10th received byte, set RX_WTR_MARK=11.

#### *64.5.2.4.11.3.8 CCA Interrupt*

CCA interrupts occur at the end of CCA and ED measurement intervals. The presence, and timing, of CCA interrupts varies based on which autosequence is engaged. See Chapter ZSM Sequence Manager, Section Supported Sequences for more details on when to expect a CCAIRQ for each of the relevant autosequences (Sequences C, T, TR, and CCCA).

#### 64.5.2.4.11.3.9 RX Interrupt

RX interrupts occur at the end of an RX operation. An RX operation can be a standalone Sequence R, or the “R” portion of a Sequence TR. An RXIRQ, in combination with a SEQIRQ, is considered a “Data Indication”. RX interrupts are not generated on packets which fail FCS (CRC check), or fail packet filtering. Either of these two conditions results in an RX recycle back to the preamble-search state. To prevent RX recycling on packets which fail filtering or FCS, set the NO_RX_RECYCLE bit of the SEQ_MGR_CTRL register to 1. To receive a Data Indication (including RXIRQ) on packets which fail CRC, clear the CRC_MSK bit of the PHY_CTR2 register to 0. To inhibit packet filtering and enable Data Indication on packets which would otherwise fail packet filtering rules, set the PROMISCUOUS bit of the PHY_CTRL4 register to 1.

#### 64.5.2.4.11.3.10 TX Interrupt

TX interrupts occur at the end of a TX operation. An TX operation can be a standalone Sequence T, the “T” portion of a Sequence TR, or the auto-TXACK portion of a Sequence R with AUTOACK=1.

#### 64.5.2.4.11.3.11 Sequencer Interrupt

The Sequencer Interrupt (SEQIRQ), indicates that an autosequence has completed, and the 802.15.4 Sequence Manager has returned to its idle state. A SEQIRQ will *always* occur at the end of an autosequence, even if the autosequence terminated abnormally (such as a Software Abort, a TC3 Timeout, or a PLL Unlock Abort). **Note:** the ABORT_STS register can be read to determine which of these abnormal terminations occurred, if any. The SEQIRQ always occurs whenever the Sequence Manager transitions from non-idle to idle state. When SEQIRQ occurs, software can be sure that the Sequence Manager is in its idle state, and a new sequence can be programmed immediately.

#### 64.5.2.4.11.3.12 TSM Interrupt

TSM_IRQ is a debug feature, enabling the Transceiver Sequence Manager (TSM) to generate an interrupt at any point in a TX or RX Warmup. TSM is a multipurpose hardware resource shared by all of the protocol engines in the SoC. Thus, TSM does not have its own interrupts; its interrupts are assigned to whichever link layer controller is currently executing an RF operation. The 802.15.4 interrupt TSM_IRQ is asserted when the TSM interrupt is enabled in the transceiver's TSM_CTRL register, and the TSM interrupt asserts during an 802.15.4 TX or RX Warmup. The TSM Interrupt status bit in the IRQSTS register of 802.15.4 space is read-only. It is a logical OR of the 2 TSM_IRQ

interrupt status bits of the XCVR_STATUS register in XCVR address space. and the asserted status bit(s) should be cleared there. There is no intended mission-mode use for TSM_IRQ. See the TSM Chapter for more details.

#### *64.5.2.4.11.3.13 Wake Interrupt*

**For Manual DSM:** This is WAKE_IRQ. A WAKE_IRQ will be triggered when the 802.15.4 Link Layer Controller has awoken from a Manual DSM (Deep Sleep Mode) cycle. WAKE_IRQ indicates that the RF Oscillator has been restarted, and the 802.15.4 EVENT_TMR has resumed counting.

**For Wake-On-Radio:**: This is WOR_IRQ. This interrupt will become set when 802.15.4 has been assigned to Wake-On-Radio, and one or more of the following Wake-On-Radio conditions occurs:

1. A Wake-On-Radio slot of type "NO_RF" has completed, and MCU wakeup is requested in the descriptor
2. A Wake-On-Radio maximum slot count has been reached
3. A Wake-On-Radio Early Exit condition has been encountered

See Chapter *Wake-On-Radio* for more details

## 64.5.2.4.12 Bit Streaming Mode

### 64.5.2.4.12.1 Introduction

The 802.15.4 Link Layer features an optional Bit Streaming Mode. When activated, this feature allows all 802.15.4 packet data, received or transmitted, to be serialized and shifted out to external hardware for further processing.

#### *64.5.2.4.12.1.1 Overview*

Bit Streaming Mode (BSM), when activated, allows all 802.15.4 packet data, received or transmitted, to be serialized and shifted out to external hardware for further processing. A simple development system can be crafted to consume the BSM outputs and generate packet trace data for all 802.15.4 traffic appearing on a network, within range of the device. This enables PAN-level monitoring and debugging. BSM uses a simple, synchronous 3-wire interface, consisting of BSM_CLK, BSM_DATA, and BSM_FRAME outputs. Packet data is shifted out serially, at the 802.15.4 bit rate (250 kHz). Signalling is provided on BSM_FRAME to indicate start-of-packet and end-of-packet conditions, and to distinguish between TX and RX packet types. BSM_DATA and BSM_FRAME are synchronous to BSM_CLK. BSM_DATA and BSM_FRAME are shifted out on falling BSM_CLK, and are intended to be captured on rising BSM_CLK. A single control bit activates or deactivates BSM. Aside from controlling this bit, BSM

requires no software support, and there is no software insight into BSM status while the mode is engaged. The BSM outputs are multiplexed with GPIO, so that the pins are available for general purpose use when BSM is disabled. BSM does not interfere with 802.15.4 packet processing, or transmit data handling, in any way; it is merely a passive monitoring and debugging tool. BSM, when engaged, will not measurably increase current consumption, since the BSM module operates at a 500 kHz clock rate.

#### *64.5.2.4.12.1.2 Features*

BSM includes the following features:

- Enables monitoring and debugging of all network traffic within range of the device
- Simple, synchronous 3-wire interface facilitates external hardware development
- Low power consumption
- Non-intrusive design.
- Minimal software support required

#### *64.5.2.4.12.1.3 Block Diagram*

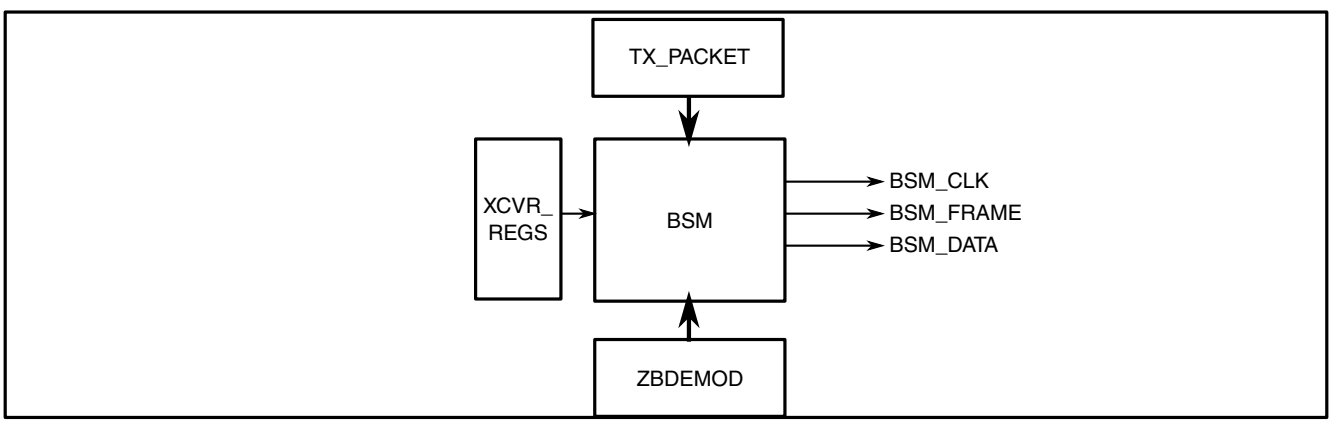



**Figure 64-165. Block diagram**

### 64.5.2.4.12.2 Memory Map and register definition

A single control bit, BSM_EN activates and deactivates BSM.

The control bit BSM_EN, is in the BSM Control Register (BSM_CTRL), which resides in 802.15.4 address space.

BSM_EN activates Bit Streaming Mode. Default is disabled. BSM mode, when enabled, allows all TX and RX packet data to appear on the 3-wire BSM interface as described in the following section.

### 64.5.2.4.12.3 Functional description

Bit Streaming Mode is enabled by setting BSM_EN=1 in the BSM_CTRL register. Once engaged, BSM will begin tracking sequence manager state, to determine when a packet is to be transmitted or received. If there is no sequence underway, or the sequence manager is in a state that does not involve transmission or reception (e.g., CCA), there will be no activity on the BSM outputs; they will be held low by hardware. Once a packet transmission or reception begins, the interface will automatically activate. The BSM_CLK will start and a single high pulse (1 BSM_CLK long) on BSM_FRAME, with BSM_DATA also high, will indicate a start-of-packet condition. The state of BSM_DATA, following the BSM_FRAME assertion, will indicate whether the packet which follows, is a TX or RX packet (1=TX, 0 = RX). The packet contents follow, starting with the next BSM_CLK period. The 802.15.4 preamble and SFD (start-of-frame delimiter), are not shifted out on BSM, since they are the same for every packet. A start-of-packet condition implies that preamble and SFD have been transmitted (for TX packets), or correctly received (for RX packets). After the start-of-packet bit and the TX/RX indicator bit, BSM packet shifting begins with the Frame Length byte (802.15.4 PHR), followed by the packet payload (802.15.4 PSDU). The MAC frame structure is shown here for reference.

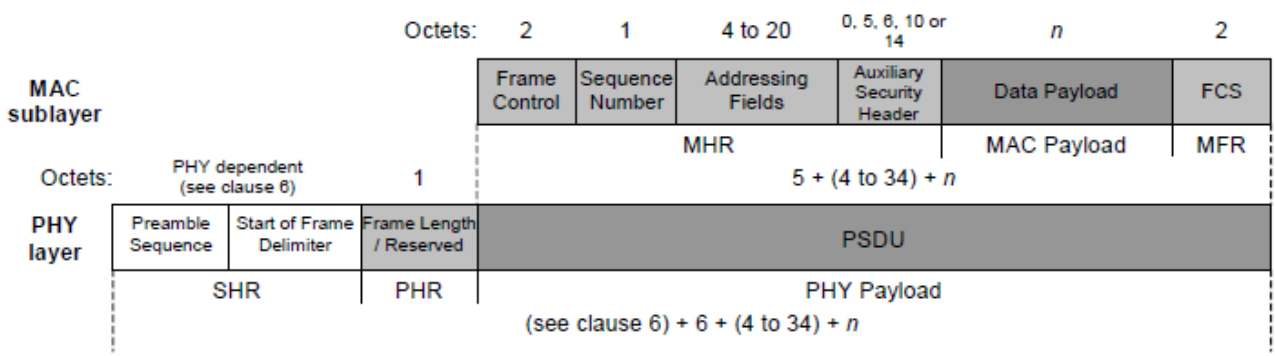


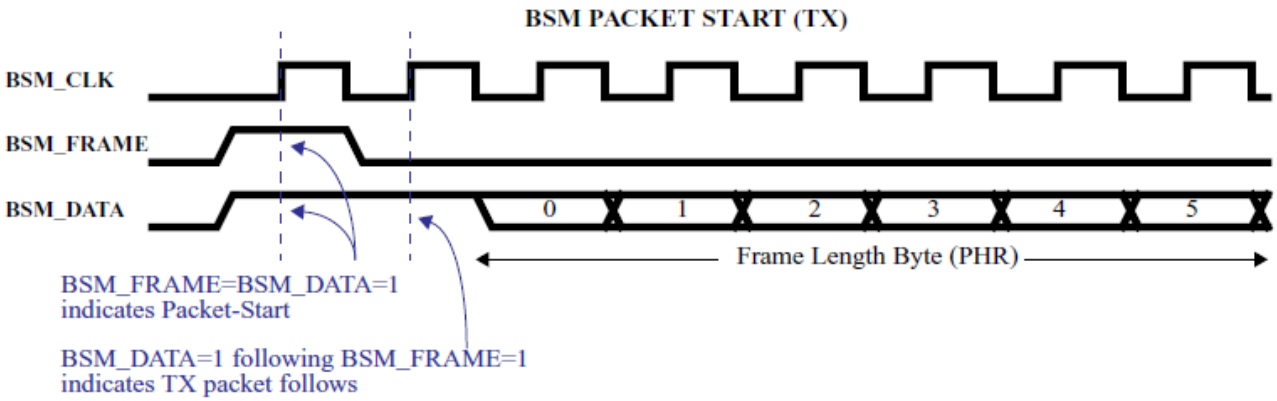

**Figure 64-166. MAC Frame Structure**

#### *64.5.2.4.12.3.1 Functional timing*

**BSM Packet Start (TX)**

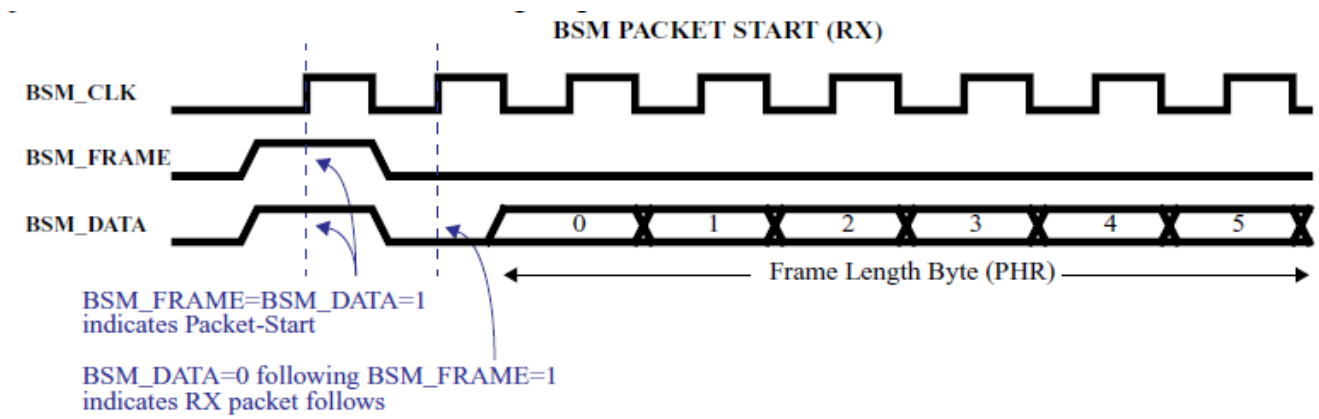
A simple protocol on BSM_FRAME and BSM_DATA are used to indicate a BSM Packet Start. If BSM_FRAME and BSM_DATA are both high at rising edge of BSM_CLK, this indicates a Packet Start condition. The state of BSM_DATA during the

bit period following the BSM_FRAME assertion, is used to indicate whether the ensuing packet is TX or RX. If BSM_DATA is high during this bit period, a TX packet follows, as shown in the following diagram.




## BSM Packet Start (RX)

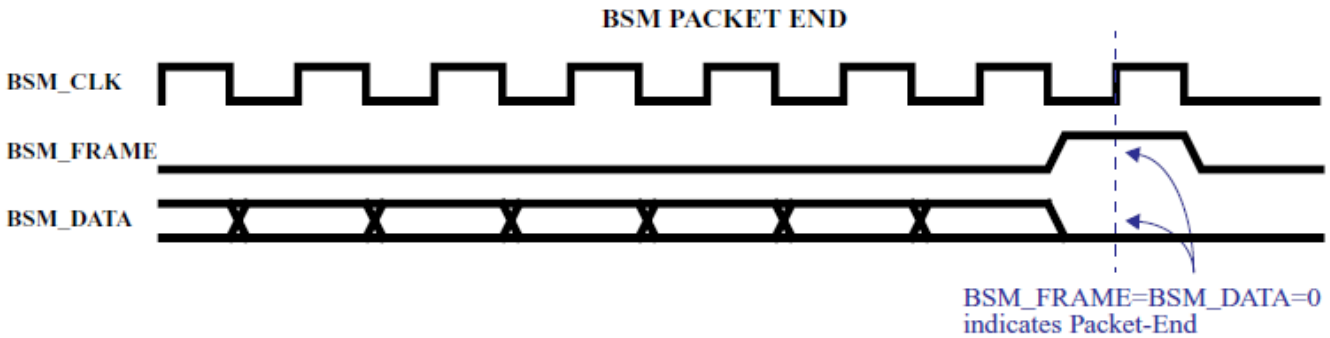
A simple protocol on BSM_FRAME and BSM_DATA is used to indicate a BSM Packet Start. If BSM_FRAME and BSM_DATA are both high at rising edge of BSM_CLK, this indicates a Packet Start condition. The state of BSM_DATA during the bit period following the BSM_FRAME assertion, is used to indicate whether the ensuing packet is TX or RX. If BSM_DATA is low during this bit period, an RX packet follows, as shown in the following diagram.




## BSM Packet End

A simple protocol on BSM_FRAME and BSM_DATA is used to indicate a BSM Packet End. If BSM_FRAME is high and BSM_DATA is low at the rising edge of BSM_SCLK, this indicates a Packet End condition. Nominally, external hardware can predict the point at which the packet should end, based on the Frame Length (first BSM byte shifted out).

However, because a TX or RX sequence could end abnormally, (e.g., a software abort or a hardware timeout), this BSM packet end indicator is provided to convey that information to the external development system. How the external hardware opts to handle an abnormally-terminated sequence, is beyond the scope of this document.




#### *64.5.2.4.12.3.2 Clocks*

BSM is a fully synchronous module. It has a single clock input **bsm_500khz_clk**. This clock is generated in the CGM (Clock Generation Module), and is only active when BSM is enabled. This input clock is divided by 2 to generate the BSM output BSM_CLK, whenever an active TX or RX operation is in progress. BSM has no asynchronous interfaces.

#### *64.5.2.4.12.3.3 Reset*

The BSM module has a single, active-low, asynchronous reset input: **rb** . At integration, this reset should be tied to the master reset for the entire transceiver. There are no special reset requirements.

#### *64.5.2.4.12.3.4 Interrupts*

BSM generates no interrupts.

### 64.5.2.4.13 Wifi Coexistence

Provisions have been made to allow for the 2.4GHz transceiver to coexist in the same space as a WiFi transceiver IC, which shares the same frequency band. The coexistence scheme designates the WiFi transceiver as the master, and the 2.4GHz transceiver as the

slave. The objective of the coexistence strategy is to prevent both the WiFi and 2.4GHz transceivers transmitting simultaneously; a configuration option exists to also prevent the 2.4GHz transceiver from receiving when the WiFi transceiver owns the RF channel.

A very basic mechanism is used in cases where 2.4GHz radio does not generate requests, but instead defers to the WiFi transceiver to grant permission to use the medium. The WiFi IC generates a signal, RF_NOT_ALLOWED. If this signal is asserted, then 2.4GHz radio does not perform any communication. When this signal is de-asserted, then 2.4GHz radio is free to perform communications. In case the signal RF_NOT_ALLOWED is de-asserted when 2.4GHz radio has already initiated the transmission/reception of a packet then the 2.4GHz radio must stop its activity immediately .

When RF_NOT_ALLOWED aborting is enabled, the 802.15.4 Link Layer hardware must respond to RF_NOT_ALLOWED assertions, by aborting any autosequence which is currently underway, and Link Layer software must not initiate any new autosequences until RF_NOT_ALLOWED has been deasserted. The hardware response must be handled autonomously by the Link Layer, with no MCU intervention required, because the RF_NOT_ALLOWED assertions can occur during a Wake-On-Radio timeslot which has been designated as "no-MCU-wakeup" in the slot's descriptor. In this scenario, the aborting of the autosequence occurs without any assistance from, or notification of, the MCU.

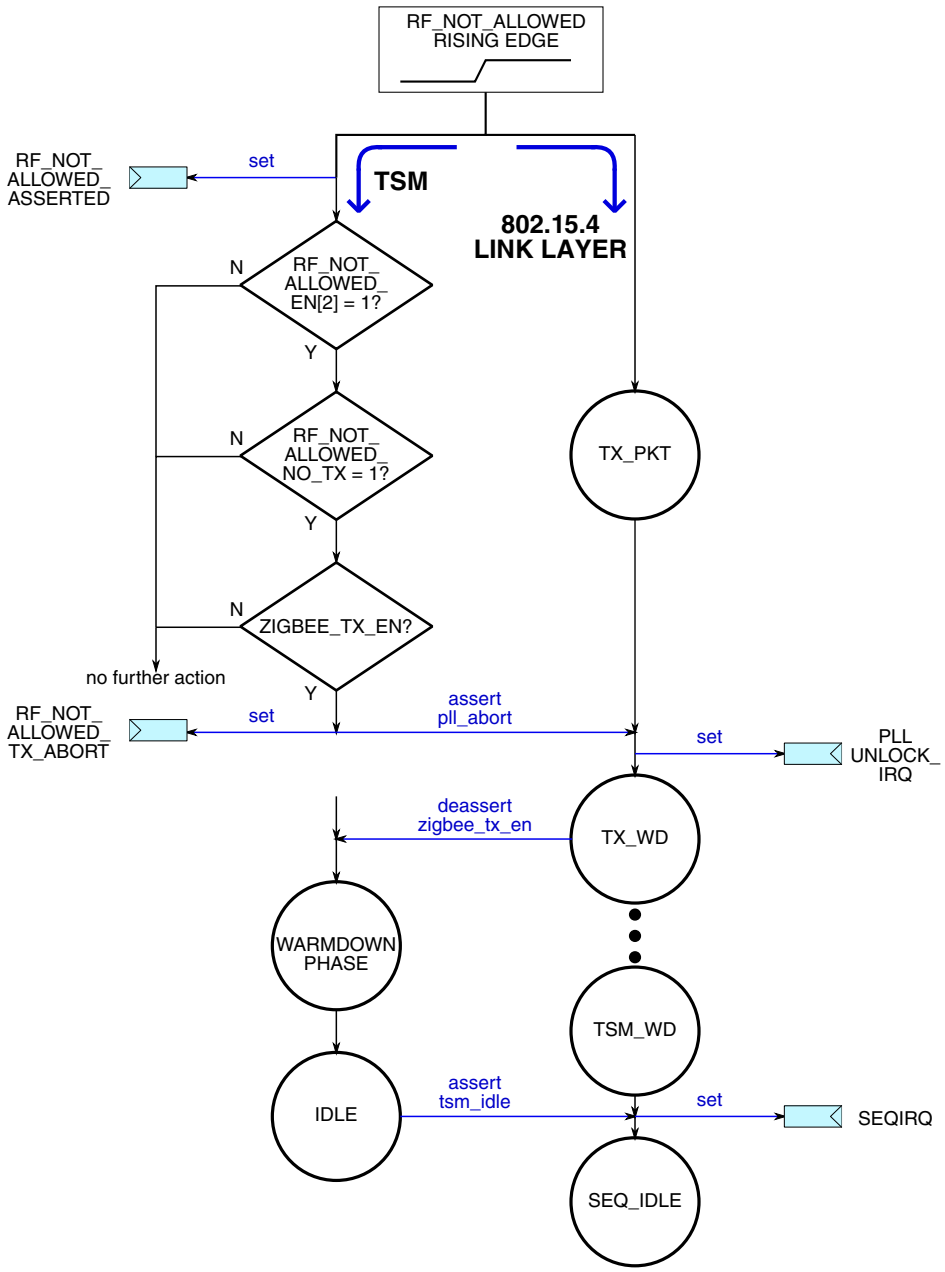
In the multi-protocol 2.4GHz radio, RF_NOT_ALLOWED aborting can be individually enabled/disabled for each protocol engine. For 802.15.4, RF_NOT_ALLOWED_EN[2] is the associated enable control bit. This bit resides in the COEX_CTRL register in XCVR address space. When this bit is 0, transitions on RF_NOT_ALLOWED are ignored by the 802.15.4 Link Layer hardware; when this bit is 1, the 802.15.4 Link Layer hardware will monitor RF_NOT_ALLOWED at all times, and abort any active autosequence which is underway when an assertion on the pin occurs. The complete hardware response to RF_NOT_ALLOWED assertions is described below.

Additional control over RF_NOT_ALLOWED aborting is provided by the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits. If RF_NOT_ALLOWED_NO_TX=1, then an RF_NOT_ALLOWED abort will occur only if an 802.15.4 autosequence is active, and currently executing a TX operation (zigbee_tx_en=1); 802.15.4 autosequence TX operations will not be aborted if RF_NOT_ALLOWED_NO_TX=0. If RF_NOT_ALLOWED_NO_RX=1, then an

RF_NOT_ALLOWED abort will occur only if an 802.15.4 autosequence is active, and currently executing an RX operation (zigbee_rx_en=1); 802.15.4 autosequence RX operatons will not be aborted if RF_NOT_ALLOWED_NO_RX=0. The RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits reside in the COEX_CTRL register.

For the purposes of triggering a hardware abort, the *pll_abort* input to the 802.15.4 Sequence Manager (ZSM) is used. This is because the hardware response to the RF_NOT_ALLOWED assertion is identical to that of a PLL unlock event. This also means that, when RF_NOT_ALLOWED aborting is enabled for 802.15.4, the PLL_UNLOCK_IRQ interrupt status bit, will be dual purpose: it will not only indicate a PLL unlock condition, but also a RF_NOT_ALLOWED abort. PLL aborting and RF_NOT_ALLOWED aborting are enabled separately, with the COEX_CTRL register maintaining the control bits required for the latter. The status bits for RF_NOT_ALLOWED aborting are also available in COEX_CTRL, so that software will be able to distinguish the source of the PLL_UNLOCK_IRQ, in case both PLL and RF_NOT_ALLOWED aborting are enabled.

The sequence of events which results in a hardware abort of an 802.15.4 TX operation triggered by an assertion on RF_NOT_ALLOWED, is a collaboration between the TSM (Transceiver Sequence Manager) and the 802.15.4 Sequence Manager (ZSM), as shown in the following diagram.



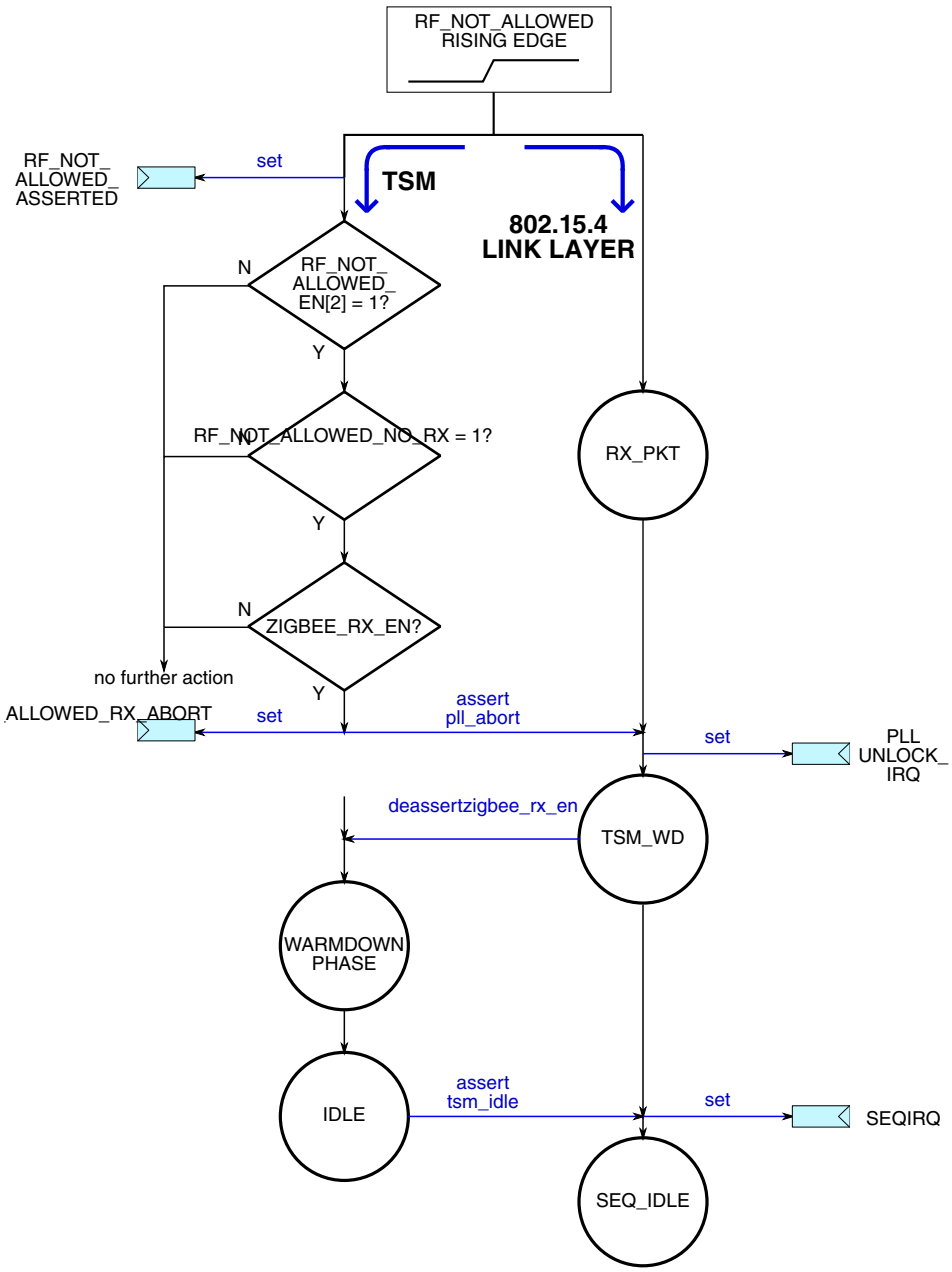

Upon assertion on RF_NOT_ALLOWED, the TSM sets the RF_NOT_ALLOWED_ASSERTED status bit in COEX_CTRL, then checks that the conditions for a hardware abort are all met:

- RF_NOT_ALLOWED_EN[2] = 1, which enables 802.15.4 to respond to RF_NOT_ALLOWED events
- RF_NOT_ALLOWED_NO_TX = 1, which enables TX operations to be aborted
- zigbee_tx_en = 1, TX request to TSM from 802.15.4 ZSM, indicating TX operation in progress

If all conditions are not met, no further action is taken. Otherwise, the TSM sets COEX_CTRL[RF_NOT_ALLOWED_TX_ABORT], and also asserts *pll_unlock* to the 802.15.4 ZSM Sequence Manager. The ZSM responds by asserting IRQSTS1[PLL_UNLOCK_IRQ] in 802.15.4 space. ZSM enters TX_WD state, which deasserts zigbee_tx_en to the TSM. This initiates the TSM TX warmdown, while the ZSM simultaneously steps through its TX warmdown sequence of states. ZSM enters TSM_WD to wait for TSM to return to idle. Once this occurs, ZSM asserts IRQSTS1[SEQIRQ] and returns to its SEQ_IDLE state. Three status bits are now set to indicate to software that the source of the abort was an RF_NOT_ALLOWED assertion:

- IRQSTS1[PLL_UNLOCK_IRQ]
- COEX_CTRL[RF_NOT_ALLOWED_ASSERTED]
- COEX_CTRL[RF_NOT_ALLOWED_TX_ABORT]

The sequence of events which results in a hardware abort of an 802.15.4 RX operation triggered by an assertion on RF_NOT_ALLOWED, is shown in the following diagram.




Upon assertion on RF_NOT_ALLOWED, the TSM sets the RF_NOT_ALLOWED_ASSERTED status bit in COEX_CTRL, then checks that the conditions for a hardware abort are all met:

- RF_NOT_ALLOWED_EN[2] = 1, which enables 802.15.4 to respond to RF_NOT_ALLOWED events
- RF_NOT_ALLOWED_NO_RX = 1, which enables RX operations to be aborted
- zigbee_rx_en = 1, RX request to TSM from 802.15.4 ZSM, indicating RX operation in progress

If all conditions are not met, no further action is taken. Otherwise, the TSM sets COEX_CTRL[RF_NOT_ALLOWED_RX_ABORT], and also asserts *pll_unlock* to the 802.15.4 ZSM Sequence Manager. The ZSM responds by asserting IRQSTS1[PLL_UNLOCK_IRQ] in 802.15.4 space. ZSM enters TSM_WD state, which deasserts zigbee_rx_en to the TSM. This initiates the TSM RX warmdown. ZSM waits in the TSM_WD to wait for TSM to return to idle. Once this occurs, ZSM asserts IRQSTS1[SEQIRQ] and returns to its SEQ_IDLE state. Three status bits are now set to indicate to software that the source of the abort was an RF_NOT_ALLOWED assertion:

- IRQSTS1[PLL_UNLOCK_IRQ]
- COEX_CTRL[RF_NOT_ALLOWED_ASSERTED]
- COEX_CTRL[RF_NOT_ALLOWED_RX_ABORT]

The complete set of control and status bits in the COEX_CTRL register is described in the following table:

| Field | R/W | Description |
|---|---|---|
| RF_NOT_ALLOWED_EN | RW | The coexistence input RF_NOT_ALLOWED can be enabled to selectively abort TX or RX sequences, with individual enables for each supported protocol, subject also to the state of the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits, according to the following table:<br>**xxx1**: RF_NOT_ALLOWED assertions are enabled to abort BLE TX and RX sequences<br>**xxx0**: RF_NOT_ALLOWED assertions are not enabled to abort BLE TX and RX sequences<br>**x1xx**: RF_NOT_ALLOWED assertions are enabled to abort 802.15.4 TX and RX sequences<br>**x0xx**: RF_NOT_ALLOWED assertions are not enabled to abort 802.15.4 TX and RX sequences<br>**1xxx**: RF_NOT_ALLOWED assertions are enabled to abort GENERIC_FSK TX and RX sequences<br>**0xxx**: RF_NOT_ALLOWED assertions are not enabled to abort GENERIC_FSK TX and RX sequences |
| RF_NOT_ALLOWED_NO_TX | RW | TX Sequences will be aborted by RF_NOT_ALLOWED assertions, if the appropriate RF_NOT_ALLOWED_EN[x] bit is set. |
| RF_NOT_ALLOWED_NO_RX | RW | RX Sequences will be aborted by RF_NOT_ALLOWED assertions, if the appropriate RF_NOT_ALLOWED_EN[x] bit is set. |
| RF_NOT_ALLOWED_ASSERTED | R/W1TC | RF_NOT_ALLOWED has gone from not asserted, to asserted, since the last time this bit was cleared |
| RF_NOT_ALLOWED_TX_ABORT | R/W1TC | An TX sequence has been aborted since the last time this bit was cleared |
| RF_NOT_ALLOWED_TX_ABORT | R/W1TC | An TX sequence has been aborted since the last time this bit was cleared |
| RF_NOT_ALLOWED | R | Reflects the instantaneous state of the RF_NOT_ALLOWED pin, |

# 64.5.2.5 ZLL register descriptions

## 64.5.2.5.1 ZLL register descriptions

### 64.5.2.5.1.1 ZLL Memory map

ZLL base address: 4103_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | INTERRUPT REQUEST STATUS (IRQSTS) | 32 | RW | See description. |
| 4h | PHY CONTROL (PHY_CTRL) | 32 | RW | 0805_FF00h |
| 8h | EVENT TIMER (EVENT_TMR) | 32 | RW | See description. |
| Ch | TIMESTAMP (TIMESTAMP) | 32 | RO | See description. |
| 10h | T1 COMPARE (T1CMP) | 32 | RW | 00FF_FFFFh |
| 14h | T2 COMPARE (T2CMP) | 32 | RW | 00FF_FFFFh |
| 18h | T2 PRIME COMPARE (T2PRIMECMP) | 32 | RW | 0000_FFFFh |
| 1Ch | T3 COMPARE (T3CMP) | 32 | RW | 00FF_FFFFh |
| 20h | T4 COMPARE (T4CMP) | 32 | RW | 00FF_FFFFh |
| 24h | PA POWER (PA_PWR) | 32 | RW | 0000_0000h |
| 28h | CHANNEL NUMBER 0 (CHANNEL_NUM0) | 32 | RW | 0000_0012h |
| 2Ch | LQI AND RSSI (LQI_AND_RSSI) | 32 | RO | See description. |
| 30h | MAC SHORT ADDRESS 0 (MACSHORTADDRS0) | 32 | RW | FFFF_FFFFh |
| 34h | MAC LONG ADDRESS 0 LSB (MACLONGADDRS0_LSB) | 32 | RW | FFFF_FFFFh |
| 38h | MAC LONG ADDRESS 0 MSB (MACLONGADDRS0_MSB) | 32 | RW | FFFF_FFFFh |
| 3Ch | RECEIVE FRAME FILTER (RX_FRAME_FILTER) | 32 | RW | See description. |
| 40h | CCA AND LQI CONTROL (CCA_LQI_CTRL) | 32 | RW | 0866_004Bh |
| 44h | CCA2 CONTROL (CCA2_CTRL) | 32 | RW | See description. |
| 4Ch | DSM CONTROL (DSM_CTRL) | 32 | RW | 0000_0000h |
| 50h | BSM CONTROL (BSM_CTRL) | 32 | RW | 0000_0000h |
| 54h | MAC SHORT ADDRESS FOR PAN1 (MACSHORTADDRS1) | 32 | RW | FFFF_FFFFh |
| 58h | MAC LONG ADDRESS 1 LSB (MACLONGADDRS1_LSB) | 32 | RW | FFFF_FFFFh |
| 5Ch | MAC LONG ADDRESS 1 MSB (MACLONGADDRS1_MSB) | 32 | RW | FFFF_FFFFh |
| 60h | DUAL PAN CONTROL (DUAL_PAN_CTRL) | 32 | RW | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 64h | CHANNEL NUMBER 1 (CHANNEL_NUM1) | 32 | RW | 0000_007Fh |
| 68h | SAM CONTROL (SAM_CTRL) | 32 | RW | 8080_4000h |
| 6Ch | SOURCE ADDRESS MANAGEMENT TABLE (SAM_TABLE) | 32 | RW | See description. |
| 70h | SOURCE ADDRESS MANAGEMENT MATCH (SAM_MATCH) | 32 | RO | See description. |
| 74h | SAM FREE INDEX (SAM_FREE_IDX) | 32 | RO | See description. |
| 78h | SEQUENCE CONTROL AND STATUS (SEQ_CTRL_STS) | 32 | RW | See description. |
| 7Ch | ACK DELAY (ACKDELAY) | 32 | RW | 0000_0007h |
| 80h | FILTER FAIL CODE (FILTERFAIL_CODE) | 32 | RW | See description. |
| 84h | RECEIVE WATER MARK (RX_WTR_MARK) | 32 | RW | 0000_00FFh |
| 8Ch | SLOT PRELOAD (SLOT_PRELOAD) | 32 | RW | 0000_0074h |
| 90h | 802.15.4 SEQUENCE STATE (SEQ_STATE) | 32 | RO | See description. |
| 94h | TIMER PRESCALER (TMR_PRESCALE) | 32 | RW | 0000_0005h |
| 98h | LENIENCY LSB (LENIENCY_LSB) | 32 | RW | 0000_0000h |
| 9Ch | LENIENCY MSB (LENIENCY_MSB) | 32 | RW | 0000_0000h |
| A0h | PART ID (PART_ID) | 32 | RO | 0000_0003h |
| 100h - 17Eh | Packet Buffer TX (PKT_BUFFER_TX0 - PKT_BUFFER_TX63) | 16 | RW | See description. |
| 180h - 1FEh | Packet Buffer RX (PKT_BUFFER_RX0 - PKT_BUFFER_RX63) | 16 | RW | See description. |

## 64.5.2.5.1.2 INTERRUPT REQUEST STATUS (IRQSTS)

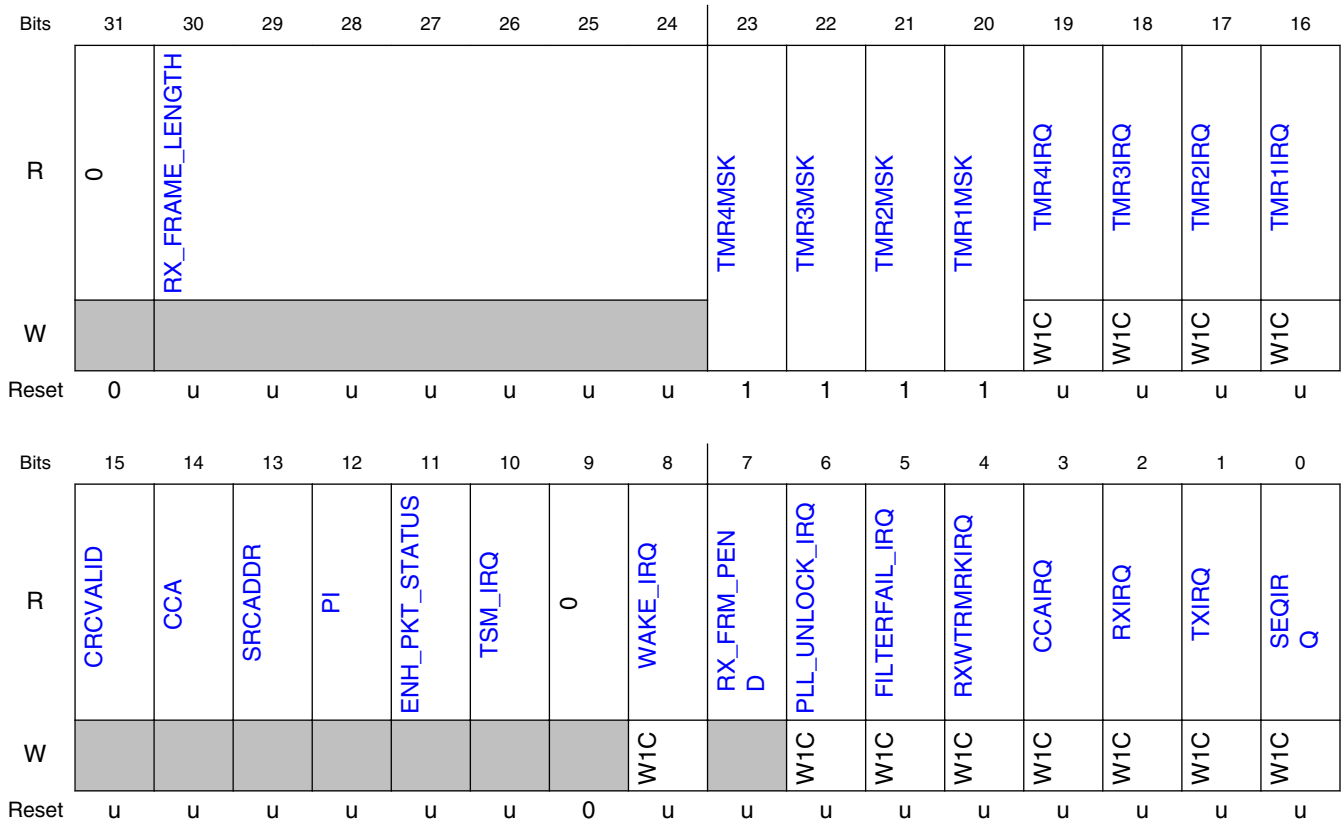
### *64.5.2.5.1.2.1 Offset*

| Register | Offset |
|---|---|
| IRQSTS | 0h |

### *64.5.2.5.1.2.2 Function*

The IRQSTS register indicates the status of interrupt requests in the 802.15.4 module.

### 64.5.2.5.1.2.3 Diagram




### 64.5.2.5.1.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>RX_FRAME_LENGTH | Receive Frame Length<br>Contents of the PHR (PHY header), or FrameLength field, of the most recently received packet. Read-only. |
| 23<br>TMR4MSK | Timer Comperator 4 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR4IRQ flag can be set |
| 22<br>TMR3MSK | Timer Comperator 3 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR3IRQ flag can be set |
| 21<br>TMR2MSK | Timer Comperator 2 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR2IRQ flag can be set |
| 20<br>TMR1MSK | Timer Comperator 1 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR1IRQ flag can be set |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19<br>TMR4IRQ | Timer 4 IRQ<br>Timer Comparator 4 Interrupt Status bit: Indiates T4CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 18<br>TMR3IRQ | Timer 3 IRQ<br>Timer Comparator 3 Interrupt Status bit: Indiates T3CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 17<br>TMR2IRQ | Timer 2 IRQ<br>Timer Comparator 2 Interrupt Status bit: Indiates comparator value matched event timer counter. This flag is shared between the T2CMP (24-bit) and T2PRIMECMP (16-bit) compare registers. This is write '1' to clear bit |
| 16<br>TMR1IRQ | Timer 1 IRQ<br>Timer Comparator 1 Interrupt Status bit: Indiates T1CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 15<br>CRCVALID | CRC Valid Status<br>Code Redundancy Check Valid: This flag indicates the compare result between the FCS field, in the most-recently received frame, and the internally calculated CRC value. This flag is cleared at next receiver warm up.<br>0b - Rx FCS != calculated CRC (incorrect)<br>1b - Rx FCS = calculated CRC (correct) |
| 14<br>CCA | CCA Status<br>Channel IDLE/BUSY indicator. This indicator is valid at CCAIRQ and also at SEQIRQ. This flag is cleared at next receiver warm up.<br>0b - IDLE<br>1b - BUSY |
| 13<br>SRCADDR | Source Address Match Status<br>If Source Address Management is engaged, meaning at least one of the following bits is set:<br>SAP0_EN<br>SAA0_EN<br>SAP1_EN<br>SAA1_EN<br><br>Then SRCADDR will be set to 1 if the packet just received is a poll request (PI=1), *and* at least one of the following conditions is met:<br>SAP0_EN and SAP0_ADDR_PRESENT<br>SAA0_EN and SAA0_ADDR_ABSENT<br>SAP1_EN and SAP1_ADDR_PRESENT<br>SAA1_EN and SAA1_ADDR_ABSENT<br><br>If SRCADDR=1, this indicates to SW that the Packet Processor has determined that an auto-TxACK frame must be transmitted with the FramePending subfield of the FrameControlField set to 1. HW will assemble and transmit this Ack packet. If the above conditions are not met, SRCADDR will be cleared to 0. |
| 12<br>PI | Poll Indication<br>0b - the received packet was not a data request<br>1b - the received packet was a data request, regardless of whether a Source Address table match occurred, or whether Source Address Management is enabled or not |
| 11<br>ENH_PKT_STATUS | Enhanced Packet Status<br>0b - The last packet received was neither 4e- nor 2015-compliant<br>1b - The last packet received was 4e- or 2015-compliant (RX_FRAME_FILTER register should be queried for additional status bits) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>TSM_IRQ | TSM IRQ<br>TSM Interrupt Request<br>0b - A TSM Interrupt has not occurred<br>1b - A TSM Interrupt has occurred |
| 9<br>— | Reserved. |
| 8<br>WAKE_IRQ | WAKE Interrupt Request<br>Wake Interrrupt. The RSIM SLEEP_TMR has matched the RSIM ZIGBEE_WAKE register, and DSM has exited. The 802.15.4 EVENT_TMR has resumed counting.<br>0b - A Wake Interrupt has not occurred<br>1b - A Wake Interrupt has occurred |
| 7<br>RX_FRM_PEND | RX Frame Pending<br>Status of the frame pending bit of the frame control field for the most-recently received packet. Read-only. |
| 6<br>PLL_UNLOCK_IRQ | PLL Unlock IRQ<br>PLL Un-lock Interrupt Status bit. A '1' indicates an unlock event has occurred in the PLL. This is write a '1' to clear bit.<br>0b - A PLL Unlock Interrupt has not occurred<br>1b - A PLL Unlock Interrupt has occurred |
| 5<br>FILTERFAIL_IRQ | Filter Fail IRQ<br>Receiver Packet Filter Fail Interrupt Status bit. A '1' indicates that the most-recently received packet has been rejected due to elements within the packet. This is write a '1' to clear bit.<br>In Dual PAN mode, FILTERFAIL_IRQ applies to either or both networks, as follows:<br>**A:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=0, FILTERFAIL_IRQ applies to PAN0.<br>**B:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=1, FILTERFAIL_IRQ applies to PAN1.<br>**C:** If PAN0 and PAN1 occupy the same channel, FILTERFAIL_IRQ is the logical 'AND' of the individual PANs' Filter Fail status.<br>0b - A Filter Fail Interrupt has not occurred<br>1b - A Filter Fail Interrupt has occurred |
| 4<br>RXWTRMRKIRQ | Receive Watermark IRQ<br>Receiver Byte Count Water Mark Interrupt Status bit. A '1' indicates that the number of bytes specified in the RX_WTR_MARK register has been reached. This is write a '1' to clear bit.<br>0b - A Receive Watermark Interrupt has not occurred<br>1b - A Receive Watermark Interrupt has occurred |
| 3<br>CCAIRQ | CCA IRQ<br>Clear Channel Assessment Interrupt Status bit. A '1' indicates completion of CCA operation. This is write '1' to clear bit.<br>0b - A CCA Interrupt has not occurred<br>1b - A CCA Interrupt has occurred |
| 2<br>RXIRQ | RX IRQ<br>Receiver Interrupt Status bit. A '1' indicates the completion of a receive operation. This is write a '1' to clear bit.<br>0b - A RX Interrupt has not occurred<br>1b - A RX Interrupt has occurred |
| 1 | TX IRQ |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TXIRQ | Transmitter Interrupt Status bit. A '1' indicates the completion of a transmit operation. This is write a '1' to clear bit.<br>0b - A TX Interrupt has not occurred<br>1b - A TX Interrupt has occurred |
| 0<br>SEQIRQ | Sequencer IRQ<br>Sequence-end Interrupt Status bit. A '1' indicates the completion of an autosequence. This interrupt will assert whenever the Sequence Manager transitions from non-idle to idle state, for any reason. This is write a '1' to clear bit.<br>0b - A Sequencer Interrupt has not occurred<br>1b - A Sequencer Interrupt has occurred |

## 64.5.2.5.1.3 PHY CONTROL (PHY_CTRL)

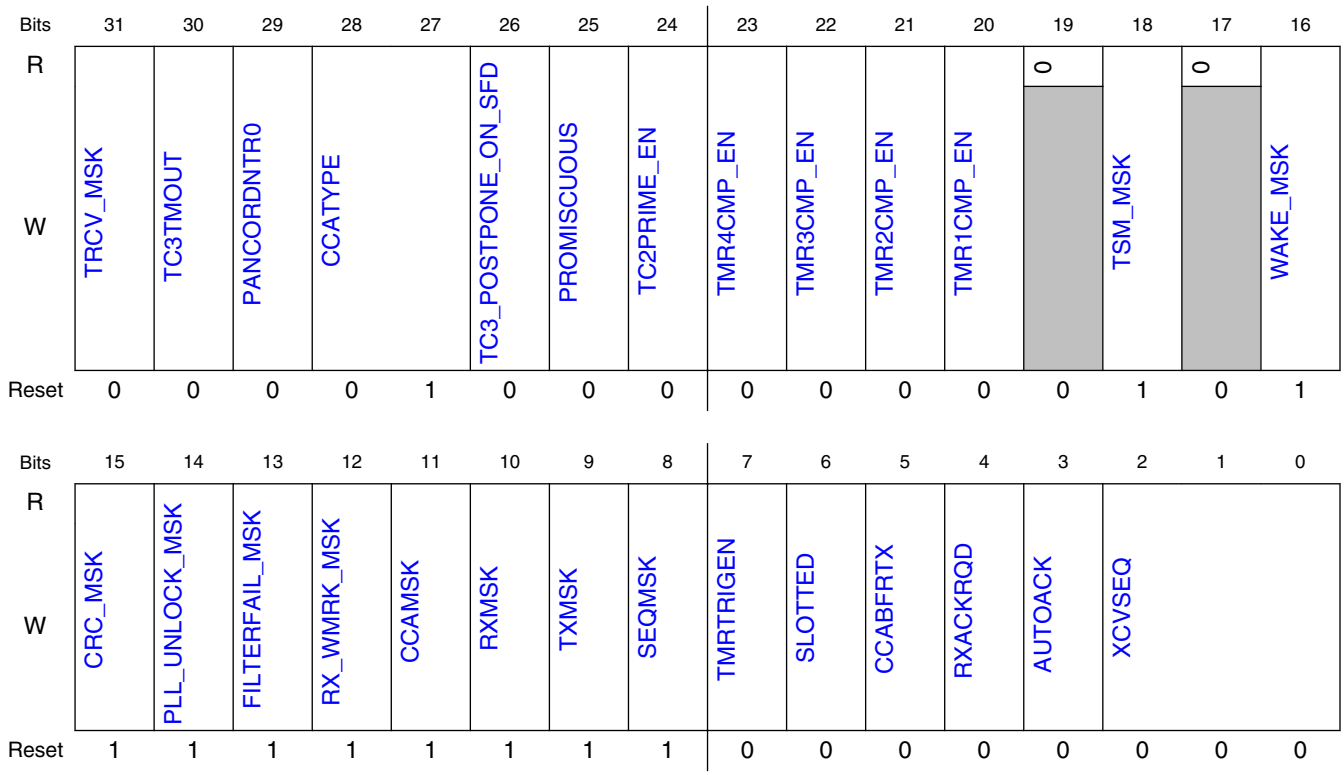
### *64.5.2.5.1.3.1 Offset*

| Register | Offset |
|---|---|
| PHY_CTRL | 4h |

### *64.5.2.5.1.3.2 Function*

PHY Control Register

### 64.5.2.5.1.3.3 Diagram




### 64.5.2.5.1.3.4 Fields

| Field | Function |
|---|---|
| 31<br>TRCV_MSK | Transceiver Global Interrupt Mask<br>Transceiver Global Interrupt Mask<br>0b - Enable any unmasked interrupt source to assert zigbee interrupt<br>1b - Mask all interrupt sources from asserting zigbee interrupt |
| 30<br>TC3TMOUT | TMR3 Timeout Enable<br>TMR3 Timeout Enable<br>0b - TMR3 is a software timer only<br>1b - Enable TMR3 to abort Rx or CCCA operations. |
| 29<br>PANCORDNTR0 | Device is a PAN Coordinator on PAN0<br>Device is a PAN Coordinator on PAN0. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| 28-27<br>CCATYPE | Clear Channel Assessment Type<br>Clear Channel Assessment Type. Selects one of four possible functions for CCA or Energy Detect, per below.<br>00b - ENERGY DETECT<br>01b - CCA MODE 1<br>10b - CCA MODE 2<br>11b - CCA MODE 3 |
| 26 | Postpone TC3 Timeout On SFD Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TC3_POSTPONE_ON_SFD | If this bit is set, if a TMR3 timeout occurs while a packet is being received (*rx_sfd_detect* = 1), the timeout-related abort will be deferred untl the packet reception is completed. If the packet is good, HW will signal Data Indication; otherwise the 802.15.4 ZSM (Sequence Manager) will return to SEQ_IDLE and await further commanding.<br>0b - TC3 Abort will occur on TMR3 timeout, regardless of rx_sfd_detect<br>1b - TC3 Abort will be deferred on TMR3 timeout if rx_sfd_detect is asserted; otherwise the TC3 Abort will occur immediately |
| 25<br>PROMISCUOUS | Promiscuous Mode Enable<br>Bypasses most packet filtering.<br>0b - normal mode<br>1b - all packet filtering except frame length checking (FrameLength>=5 and FrameLength<=127) is bypassed. |
| 24<br>TC2PRIME_EN | Timer 2 Prime Compare Enable<br>0b - Don't allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ<br>1b - Allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ |
| 23<br>TMR4CMP_EN | Timer 4 Compare Enable<br>0b - Don't allow an Event Timer Match to T4CMP to set TMR4IRQ<br>1b - Allow an Event Timer Match to T4CMP to set TMR4IRQ |
| 22<br>TMR3CMP_EN | Timer 3 Compare Enable<br>0b - Don't allow an Event Timer Match to T3CMP to set TMR3IRQ<br>1b - Allow an Event Timer Match to T3CMP to set TMR3IRQ |
| 21<br>TMR2CMP_EN | Timer 2 Compare Enable<br>0b - Don't allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ<br>1b - Allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ |
| 20<br>TMR1CMP_EN | Timer 1 Compare Enable<br>0b - Don't allow an Event Timer Match to T1CMP to set TMR1IRQ<br>1b - Allow an Event Timer Match to T1CMP to set TMR1IRQ |
| 19<br>— | Reserved. |
| 18<br>TSM_MSK | Reserved.<br>0b - allows assertion of a TSM interrupt to generate a zigbee interrupt<br>1b - Assertion of a TSM interrupt will set the TSM_IRQ status bit, but a zigbee interrupt is not generated |
| 17<br>— | Reserved. |
| 16<br>WAKE_MSK | Reserved.<br>0b - Allows a wakeup from DSM to generate a zigbee interrupt<br>1b - Wakeup from DSM will set the WAKE_IRQ status bit, but a zigbee interrupt is not generated |
| 15<br>CRC_MSK | CRC Mask<br>CRC Mask<br>0b - sequence manager ignores CRCVALID and considers the receive operation complete after the last octet of the frame has been received.<br>1b - sequence manager requires CRCVALID=1 at the end of the received frame in order for the receive operation to complete successfully; if CRCVALID=0, sequence manager will return to preamble-detect mode after the last octet of the frame has been received. |
| 14<br>PLL_UNLOCK_MSK | PLL Unlock Interrupt Mask<br>0b - allows PLL unlock event to generate a zigbee interrupt<br>1b - A PLL unlock event will set the PLL_UNLOCK_IRQ status bit, but a zigbee interrupt is not generated |
| 13 | FilterFail Interrupt Mask |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FILTERFAIL_MSK | FilterFail Interrupt Mask<br>0b - allows Packet Processor Filtering Failure to generate a zigbee interrupt<br>1b - A Packet Processor Filtering Failure will set the FILTERFAIL_IRQ status bit, but a zigbee interrupt is not generated |
| 12<br>RX_WMRK_MSK | RX Watermark Interrupt Mask<br>RX Watermark Interrupt Mask<br>0b - allows a Received Byte Count match to the RX_WTR_MARK threshold register to generate a zigbee interrupt<br>1b - A Received Byte Count match to the RX_WTR_MARK threshold register will set the RXWTRMRKIRQ status bit, but a zigbee interrupt is not generated |
| 11<br>CCAMSK | CCA Interrupt Mask<br>CCA Interrupt Mask<br>0b - allows completion of a CCA operation to generate a zigbee interrupt<br>1b - Completion of a CCA operation will set the CCA status bit, but a zigbee interrupt is not generated |
| 10<br>RXMSK | RX Interrupt Mask<br>0b - allows completion of a RX operation to generate a zigbee interrupt<br>1b - Completion of a RX operation will set the RXIRQ status bit, but a zigbee interrupt is not generated |
| 9<br>TXMSK | TX Interrupt Mask<br>0b - allows completion of a TX operation to generate a zigbee interrupt<br>1b - Completion of a TX operation will set the TXIRQ status bit, but a zigbee interrupt is not generated |
| 8<br>SEQMSK | Sequencer Interrupt Mask<br>0b - allows completion of an autosequence to generate a zigbee interrupt<br>1b - Completion of an autosequence will set the SEQIRQ status bit, but a zigbee interrupt is not generated |
| 7<br>TMRTRIGEN | Timer2 Trigger Enable<br>0b - programmed sequence initiates immediately upon write to XCVSEQ.<br>1b - allow timer TC2 (or TC2') to initiate a preprogrammed sequence (see XCVSEQ register). |
| 6<br>SLOTTED | Slotted Mode<br>Slotted Mode, for beacon-enabled networks. Applies only to Sequences T, TR, and R, ignored during all other sequences. Used, in concert with CCABFRTX, to determine how many CCA measurements are required prior to a transmit operation. Also used during R sequence to determine whether the ensuing transmit acknowledge frame (if any) needs to be synchronized to a backoff slot boundary. |
| 5<br>CCABFRTX | CCA Before TX<br>Applies only to Sequences T and TR, ignored during all other sequences.<br>0b - no CCA required, transmit operation begins immediately.<br>1b - at least one CCA measurement is required prior to the transmit operation (see also SLOTTED). |
| 4<br>RXACKRQD | Receive Acknowledge Frame required<br>Applies only to Sequence TR, ignored during all other sequences.<br>0b - An ordinary receive frame (any type of frame) follows the transmit frame.<br>1b - A receive Ack frame is expected to follow the transmit frame (non-Ack frames are rejected). |
| 3<br>AUTOACK | Auto Acknowledge Enable<br>Applies only to Sequence R and Sequence TR, ignored during other sequences<br>0b - sequence manager will not follow a receive frame with a Tx Ack frame, under any conditions; the autosequence will terminate after the receive frame.<br>1b - sequence manager will follow a receive frame with an automatic hardware-generated Tx Ack frame, assuming other necessary conditions are met. |

*Table continues on the next page...*

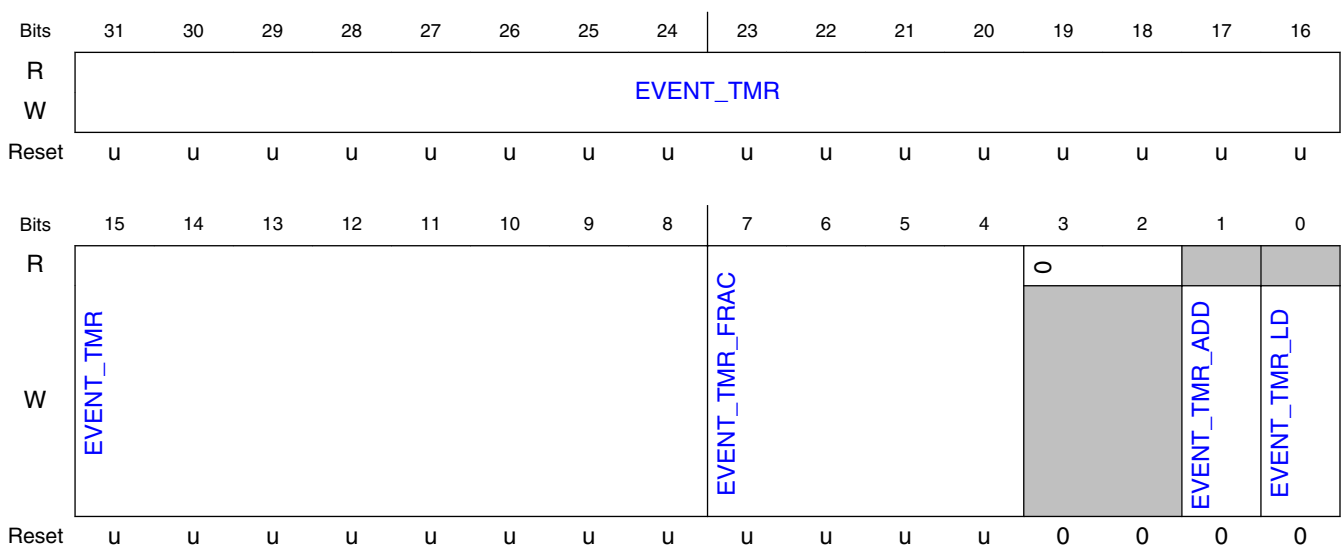
| Field | Function |
|---|---|
| 2-0<br>XCVSEQ | 802.15.4 Transceiver Sequence Selector<br>The Transceiver Sequence Selector register selects an autosequence for the sequence manager to execute. Sequence initiation can be immediate, or scheduled (see TMRTRIGEN). A write of XCVSEQ=IDLE will abort any ongoing sequence. A write of XCVSEQ=IDLE must always be performed after a sequence is complete, and before a new sequence is programmed. Any write to XCVSEQ other than XCVSEQ=IDLE during an ongoing sequence, shall be ignored. The mapping of XCVSEQ to sequence types is as follows:<br>000b - I (IDLE)<br>001b - R (RECEIVE)<br>010b - T (TRANSMIT)<br>011b - C (CCA)<br>100b - TR (TRANSMIT/RECEIVE)<br>101b - CCCA (CONTINUOUS CCA)<br>110b - Reserved<br>111b - Reserved |

## 64.5.2.5.1.4 EVENT TIMER (EVENT_TMR)

### *64.5.2.5.1.4.1 Offset*

| Register | Offset |
|---|---|
| EVENT_TMR | 8h |

### *64.5.2.5.1.4.2 Diagram*



#### *64.5.2.5.1.4.3 Fields*

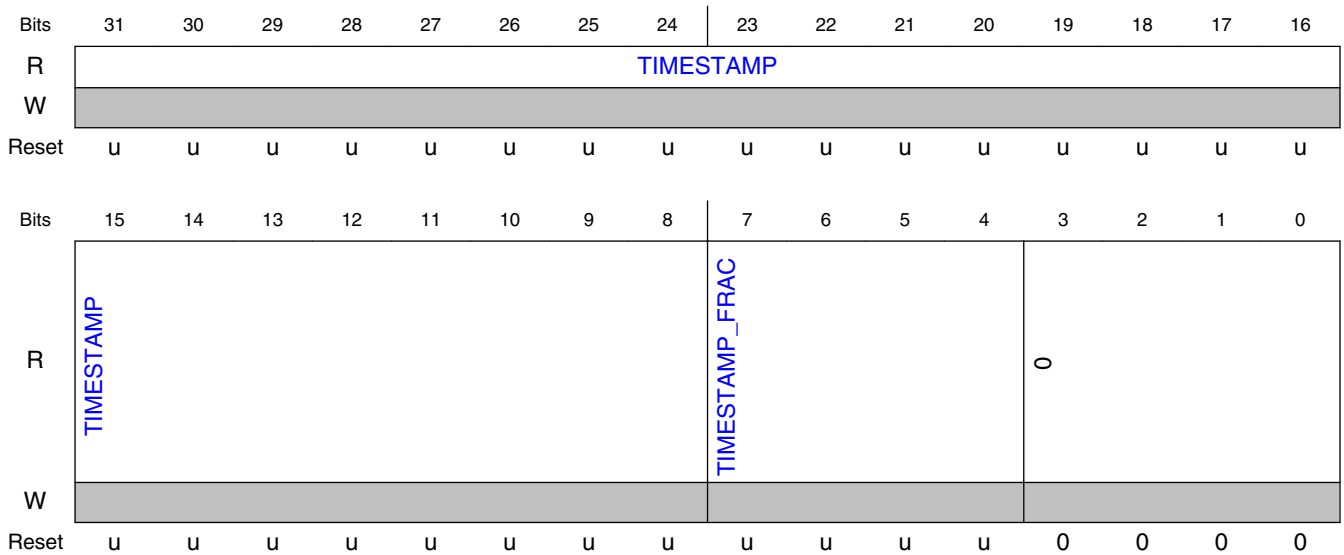
| Field | Function |
|---|---|
| 31-8<br>EVENT_TMR | Event Timer Integer Component<br>The EVENT_TMR represents a 28-bit number with 24 Integer bits and 4 Fractional bits. This field is the integer component, which appears to the left of the decimal point. Most 802.15.4 operations which relate to the Event Timer require only the integer component. |
| 7-4<br>EVENT_TMR_FRAC | Event Timer Fractional Component<br>The EVENT_TMR represents a 28-bit number with 24 Integer bits and 4 Fractional bits. This field is the fractional component, which appears to the right of the decimal point. The fractional component can be ignored for most 802.15.4 operations which relate to the Event Timer; it has been provided only to allow for microsecond accuracy when software updates the Event Timer after a long period of DSM (sleep) mode |
| 3-2<br>— | Reserved. |
| 1<br>EVENT_TMR_ADD | Event Timer Add Enable<br>Increments the Event Timer by the values written to the EVENT_TMR[23:0] and EVENT_TMR_FRAC[3:0] fields of this register. |
| 0<br>EVENT_TMR_LD | Event Timer Load Enable<br>Loads the Event Timer with the values written to the EVENT_TMR[23:0] and EVENT_TMR_FRAC[3:0] fields of this register. |

## 64.5.2.5.1.5 TIMESTAMP (TIMESTAMP)

#### *64.5.2.5.1.5.1 Offset*

| Register | Offset |
|---|---|
| TIMESTAMP | Ch |

#### 64.5.2.5.1.5.2 Diagram




#### 64.5.2.5.1.5.3 Fields

| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP | Timestamp<br>Holds the latched value of the integer portion of the Event Timer current time corresponding to the beginning of the most recently received packet, at the point of SFD detection |
| 7-4<br>TIMESTAMP_FRAC | Timestamp Fractional<br>Holds the latched value of the fractional portion of the Event Timer current time corresponding to the beginning of the most recently received packet, at the point of SFD detection |
| 3-0<br>— | Reserved. |

### 64.5.2.5.1.6 T1 COMPARE (T1CMP)

#### 64.5.2.5.1.6.1 Offset

| Register | Offset |
|---|---|
| T1CMP | 10h |

#### 64.5.2.5.1.6.2 Function

TMR1 Compare Value

#### 64.5.2.5.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | T1CMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T1CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T1CMP | TMR1 Compare Value<br>TMR1 compare value. If TMR1CMP_EN=1 and the Event Timer matches this value, TMR1IRQ is set. |

### 64.5.2.5.1.7 T2 COMPARE (T2CMP)

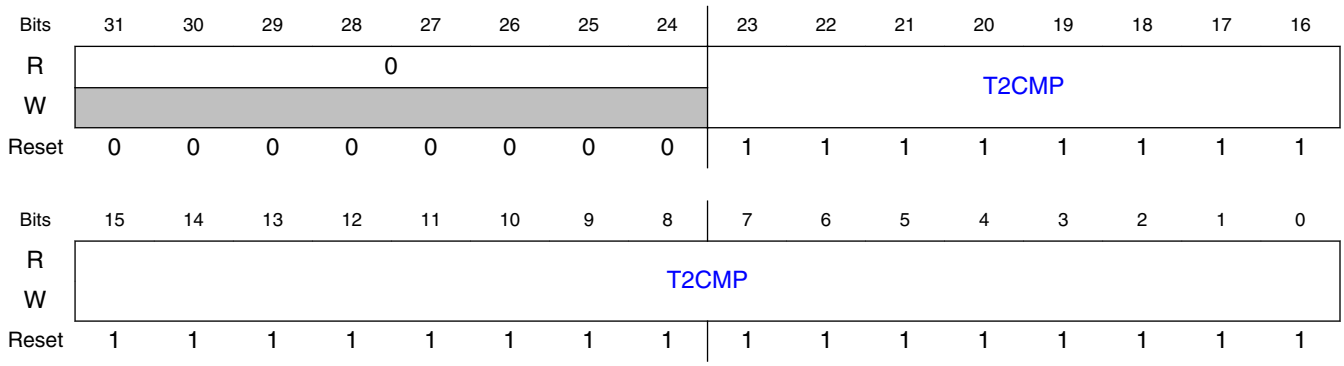
#### 64.5.2.5.1.7.1 Offset

| Register | Offset |
|---|---|
| T2CMP | 14h |

#### 64.5.2.5.1.7.2 Function

TMR2 Compare Value

#### 64.5.2.5.1.7.3 Diagram




#### 64.5.2.5.1.7.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T2CMP | TMR2 Compare Value<br>TMR2 compare value. If TMR2CMP_EN=1 and TC2PRIME_EN=0 and the Event Timer matches this value, TMR2IRQ is set. |

### 64.5.2.5.1.8 T2 PRIME COMPARE (T2PRIMECMP)

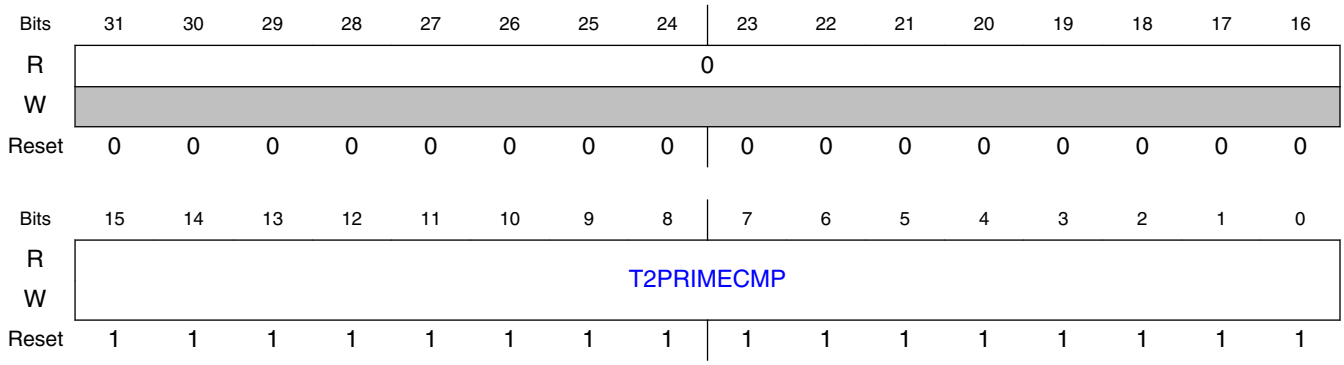
#### 64.5.2.5.1.8.1 Offset

| Register | Offset |
|---|---|
| T2PRIMECMP | 18h |

#### 64.5.2.5.1.8.2 Function

TMR2 Prime Compare Value

#### 64.5.2.5.1.8.3 Diagram




#### 64.5.2.5.1.8.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-0<br>T2PRIMECMP | TMR2 Prime Compare Value<br>TMR2 compare value. If TMR2CMP_EN=1 and TC2PRIME_EN=1 and the Event Timer matches this value, TMR2IRQ is set. |

### 64.5.2.5.1.9 T3 COMPARE (T3CMP)

#### 64.5.2.5.1.9.1 Offset

| Register | Offset |
|---|---|
| T3CMP | 1Ch |

#### 64.5.2.5.1.9.2 Function

TMR3 Compare Value

#### 64.5.2.5.1.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | T3CMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T3CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.9.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T3CMP | TMR3 Compare Value<br>TMR3 compare value. If TMR3CMP_EN=1 and the Event Timer matches this value, TMR3IRQ is set. |

### 64.5.2.5.1.10 T4 COMPARE (T4CMP)

#### 64.5.2.5.1.10.1 Offset

| Register | Offset |
|---|---|
| T4CMP | 20h |

#### 64.5.2.5.1.10.2 Function

TMR4 Compare Value

#### 64.5.2.5.1.10.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | T4CMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T4CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.10.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T4CMP | TMR4 Compare Value<br>TMR4 compare value. If TMR4CMP_EN=1 and the Event Timer matches this value, TMR4IRQ is set. |

### 64.5.2.5.1.11 PA POWER (PA_PWR)

#### 64.5.2.5.1.11.1 Offset

| Register | Offset |
|---|---|
| PA_PWR | 24h |

#### 64.5.2.5.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | PA_PWR | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.5.2.5.1.11.3 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>PA_PWR | PA Power |

### 64.5.2.5.1.12 CHANNEL NUMBER 0 (CHANNEL_NUM0)

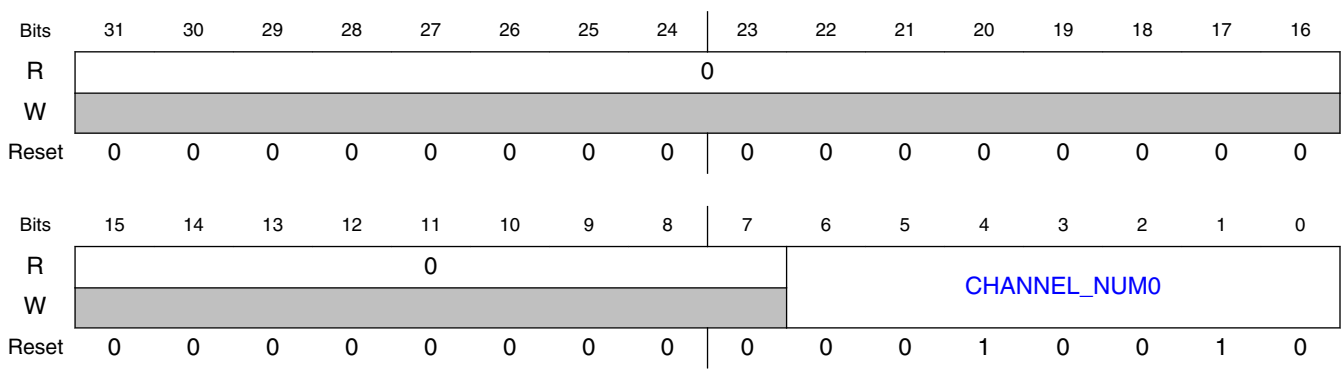
#### 64.5.2.5.1.12.1 Offset

| Register | Offset |
|---|---|
| CHANNEL_NUM0 | 28h |

#### 64.5.2.5.1.12.2 Function

Channel Number for PAN0

#### 64.5.2.5.1.12.3 Diagram




#### 64.5.2.5.1.12.4 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0 | Channel Number for PAN0 |

| Field | Function |
|---|---|
| CHANNEL_NUM0 | This is the mapped channel number used to transmit and receive Zigbee packets. If Dual PAN is engaged, this register applies to PAN0. CHANNEL_NUM0 should be in the range:<br>11 <= CHANNEL_NUM0 <= 26 |

## 64.5.2.5.1.13 LQI AND RSSI (LQI_AND_RSSI)

### *64.5.2.5.1.13.1 Offset*

| Register | Offset |
|---|---|
| LQI_AND_RSSI | 2Ch |

### *64.5.2.5.1.13.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CCA1_ED_FNL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RSSI | | | | | | | | LQI_VALUE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### *64.5.2.5.1.13.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-16<br>CCA1_ED_FNL | Final Result for CCA Mode 1 and Energy Detect<br>Output register to show final averaged RSSI value or compensated value of the same at the end of a CCA Mode1 or Energy Detect computation. |
| 15-8<br>RSSI | RSSI Value<br>Received Signal Strength Indicator, in dBm |
| 7-0<br>LQI_VALUE | LQI Value<br>Link Quality Indicator for the most recently received packet. (LQI is also available in the Packet Buffer, at the end of the received packet data) |

## 64.5.2.5.1.14 MAC SHORT ADDRESS 0 (MACSHORTADDRS0)

### *64.5.2.5.1.14.1 Offset*

| Register | Offset |
|---|---|
| MACSHORTADDRS0 | 30h |

### *64.5.2.5.1.14.2 Function*

MAC SHORT ADDRESS for PAN0

### *64.5.2.5.1.14.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACSHORTADDRS0 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACPANID0 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### *64.5.2.5.1.14.4 Fields*

| Field | Function |
|---|---|
| 31-16<br>MACSHORTADDRS0 | MAC SHORT ADDRESS FOR PAN0<br>MAC Short Address for PAN0, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| 15-0<br>MACPANID0 | MAC PAN ID for PAN0<br>MAC PAN ID for PAN0. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID against this register. Also, if PANCORDNTR0=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |

## 64.5.2.5.1.15 MAC LONG ADDRESS 0 LSB (MACLONGADDRS0_LSB)

#### 64.5.2.5.1.15.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS0_LSB | 34h |

#### 64.5.2.5.1.15.2 Function

MAC LONG ADDRESS for PAN0 LSB

#### 64.5.2.5.1.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.15.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS0_LSB | MAC LONG ADDRESS for PAN0 LSB<br>MAC Long Address for PAN0, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

### 64.5.2.5.1.16 MAC LONG ADDRESS 0 MSB (MACLONGADDRS0_MSB)

#### 64.5.2.5.1.16.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS0_MSB | 38h |

#### *64.5.2.5.1.16.2 Function*

MAC LONG ADDRESS for PAN0 MSB

#### *64.5.2.5.1.16.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### *64.5.2.5.1.16.4 Fields*

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS0_MSB | MAC LONG ADDRESS for PAN0 MSB<br>MAC Long Address for PAN0, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

### 64.5.2.5.1.17 RECEIVE FRAME FILTER (RX_FRAME_FILTER)

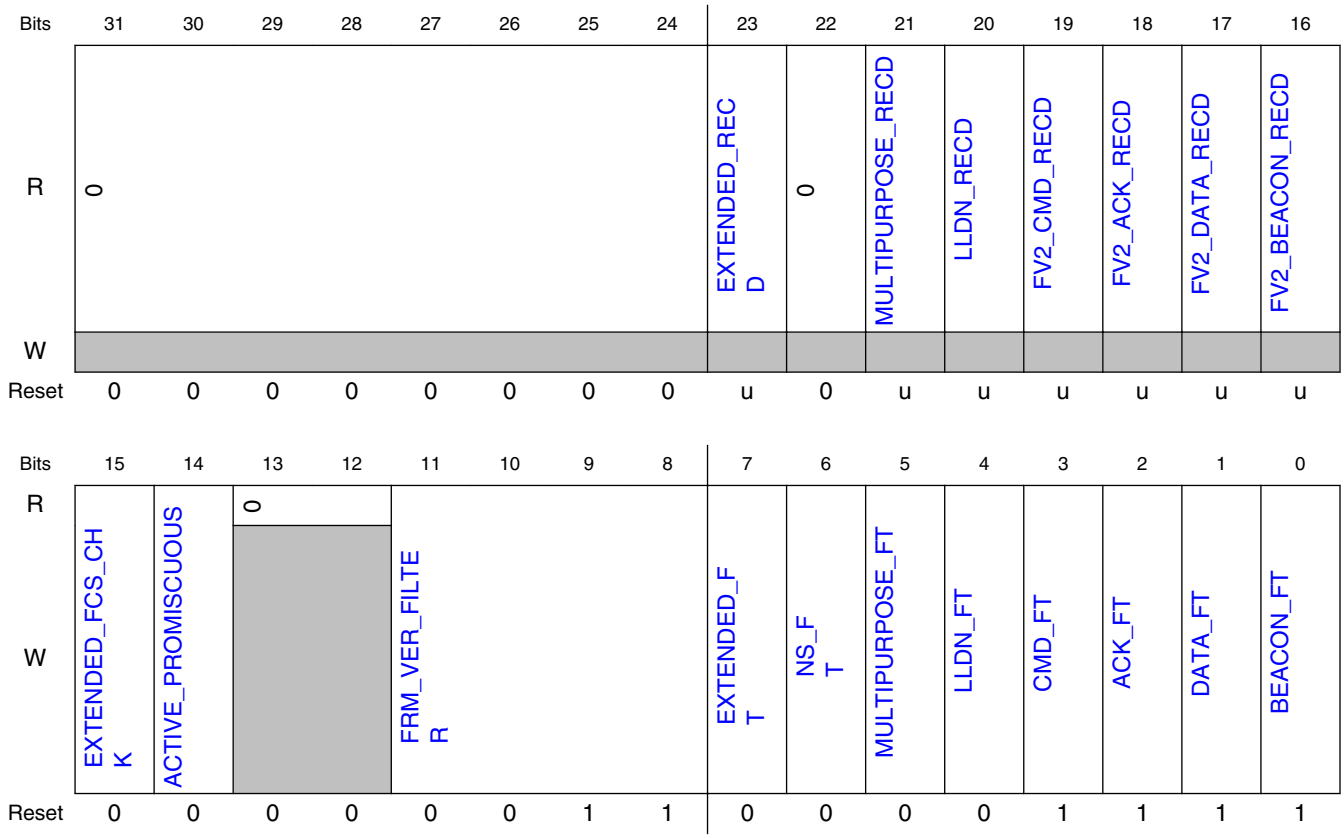
#### *64.5.2.5.1.17.1 Offset*

| Register | Offset |
|---|---|
| RX_FRAME_FILTER | 3Ch |

#### *64.5.2.5.1.17.2 Function*

RECEIVE FRAME FILTER

### 64.5.2.5.1.17.3 Diagram




### 64.5.2.5.1.17.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23<br>EXTENDED_RECD | Extended Packet Received<br>0b - The last packet received was not Frame Type EXTENDED<br>1b - The last packet received was Frame Type EXTENDED, and EXTENDED_FT=1 to allow such packets. |
| 22<br>— | Reserved. |
| 21<br>MULTIPURPOSE_RECD | Multipurpose Packet Received<br>0b - last packet received was not Frame Type MULTIPURPOSE<br>1b - The last packet received was Frame Type MULTIPURPOSE, and MULTIPURPOSE_FT=1 to allow such packets. |
| 20<br>LLDN_RECD | LLDN Packet Received<br>0b - The last packet received was not Frame Type LLDN<br>1b - The last packet received was Frame Type LLDN, and LLDN_FT=1 to allow such packets. |
| 19 | Frame Version 2 MAC Command Packet Received<br>0b - The last packet received was not Frame Type MAC Command with Frame Version 2 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FV2_CMD_RECD | 1b - The last packet received was Frame Type MAC Command with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 18<br>FV2_ACK_RECD | Frame Version 2 Acknowledge Packet Received<br>0b - The last packet received was not Frame Type Ack with Frame Version 2<br>1b - The last packet received was Frame Type Ack with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 17<br>FV2_DATA_RECD | Frame Version 2 Data Packet Received<br>0b - The last packet received was not Frame Type Data with Frame Version 2<br>1b - The last packet received was Frame Type Data with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 16<br>FV2_BEACON_RECD | Frame Version 2 Beacon Packet Received<br>0b - The last packet received was not Frame Type Beacon with Frame Version 2<br>1b - The last packet received was Frame Type Beacon with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 15<br>EXTENDED_FCS_CHK | Verify FCS on Frame Type Extended<br>0b - Packet Processor will not check FCS for Frame Type EXTENDED (default)<br>1b - Packet Processor will check FCS at end-of-packet based on packet length derived from PHR, for Frame Type EXTENDED |
| 14<br>ACTIVE_PROMISCUOUS | Active Promiscuous<br>0b - normal operation<br>1b - Provide Data Indication on all received packets under the same rules which apply in PROMISCUOUS mode, however acknowledge those packets under rules which apply in non-PROMISCUOUS mode |
| 13-12<br>— | Reserved. |
| 11-8<br>FRM_VER_FILTER | Frame Version selector.<br>The incoming packet's Frame Control Field is parsed to obtain the FrameVersion subfield, and that value is compared against this register, in accordance with the following:<br>xxx1: Accept received packets with FrameVersion=00<br>xx1x: Accept received packets with FrameVersion=01<br>x1xx: Accept received packets with FrameVersion=10<br>1xxx: Accept received packets with FrameVersion=11<br>These filtering rules apply to Beacon, Acknowledge, Data, and MAC Command Frame Types, since these frame types require a 2-octet Frame Control Field which embeds a 2-bit FrameVersion subfield. Later frame types introduced in the 802.15.4e addendum (LLDN, Multipurpose) don't guarantee a FrameVersion subfield with the original meaning, so these filtering rules do not apply to these frame types. See registers LLDN_FT and MULTIPURPOSE_FT.<br>For Acknowledge frames, FrameVersion bits are ignored by the Packet Processor, irrespective of FRM_VER_FILTER. |
| 7<br>EXTENDED_FT | Extended Frame Type Enable<br>0b - reject all Extended frames<br>1b - Extended frame type enabled (Frame Type 7). |
| 6<br>NS_FT | "Not Specified" Frame Type Enable<br>This bit enables reception of all Frame Types not covered by the other _FT bits in this register.<br>0b - reject all "Not Specified" frames<br>1b - Not-specified (reserved) frame type enabled. Applies to Frame Type 6. No packet filtering is performed, except for frame length checking (FrameLength>=5 and FrameLength<=127). No AUTOACK is transmitted for this Frame Type |
| 5 | Multipurpose Frame Type Enable<br>0b - reject all Multipurpose frames |

*Table continues on the next page...*

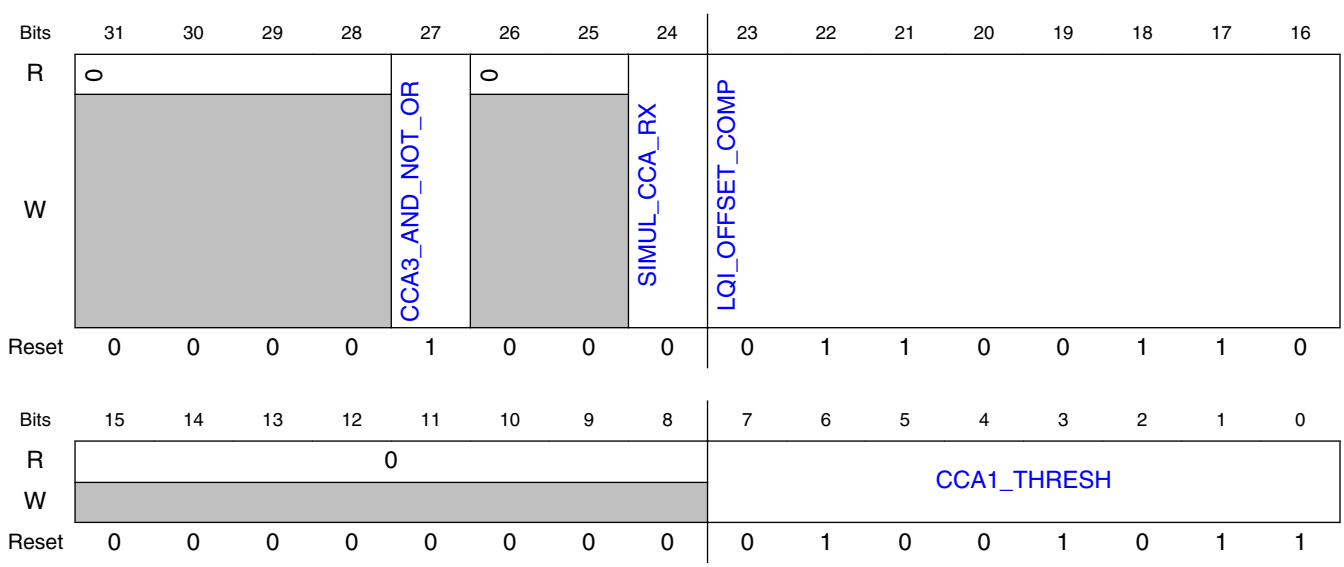
| Field | Function |
|---|---|
| MULTIPURPOSE_FT | 1b - Multipurpose frame type enabled (Frame Type 5). |
| 4<br>LLDN_FT | LLDN Frame Type Enable<br>0b - reject all LLDN frames<br>1b - LLDN frame type enabled (Frame Type 4). |
| 3<br>CMD_FT | MAC Command Frame Type Enable<br>0b - reject all MAC Command frames<br>1b - MAC Command frame type enabled. |
| 2<br>ACK_FT | Ack Frame Type Enable<br>0b - reject all Acknowledge frames<br>1b - Acknowledge frame type enabled. |
| 1<br>DATA_FT | Data Frame Type Enable<br>0b - reject all Beacon frames<br>1b - Data frame type enabled. |
| 0<br>BEACON_FT | Beacon Frame Type Enable<br>0b - reject all Beacon frames<br>1b - Beacon frame type enabled. |

## 64.5.2.5.1.18 CCA AND LQI CONTROL (CCA_LQI_CTRL)

### *64.5.2.5.1.18.1 Offset*

| Register | Offset |
|---|---|
| CCA_LQI_CTRL | 40h |

### *64.5.2.5.1.18.2 Diagram*



#### *64.5.2.5.1.18.3 Fields*

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>CCA3_AND_NOT_OR | CCA Mode 3 AND not OR<br>Determines the way CCA3 is required to be detected<br>0b - CCA1 or CCA2<br>1b - CCA1 and CCA2 |
| 26-25<br>— | Reserved. |
| 24<br>SIMUL_CCA_RX | Simultaneous CCA and Receive Enable<br>During autosequences of type Sequence T or Sequence TR with CCA before TX, this bit allows a packet to be received if preamble and SFD are detected during the CCA measurement window.<br>0b - Packets can't be received during CCA measurement<br>1b - Packet reception is enabled during CCA measurement if preamble and SFD are detected |
| 23-16<br>LQI_OFFSET_COMP | LQI Offset Compensation<br>Programmable amount to offset RSSI based LQI value |
| 15-8<br>— | Reserved. |
| 7-0<br>CCA1_THRESH | CCA Mode 1 Threshold<br>Programmable energy threshold register for CCA mode 1. |

## 64.5.2.5.1.19 CCA2 CONTROL (CCA2_CTRL)

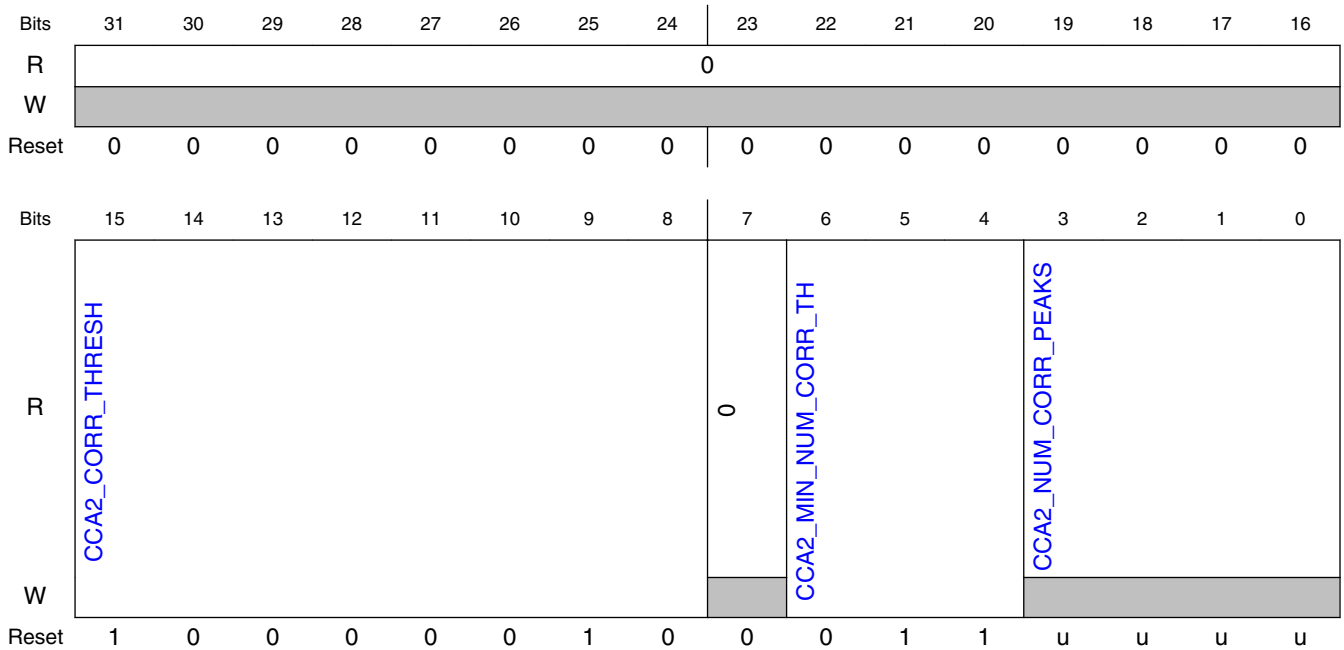
#### *64.5.2.5.1.19.1 Offset*

| Register | Offset |
|---|---|
| CCA2_CTRL | 44h |

#### *64.5.2.5.1.19.2 Function*

CCA Mode 2 Control

### 64.5.2.5.1.19.3 Diagram




### 64.5.2.5.1.19.4 Fields

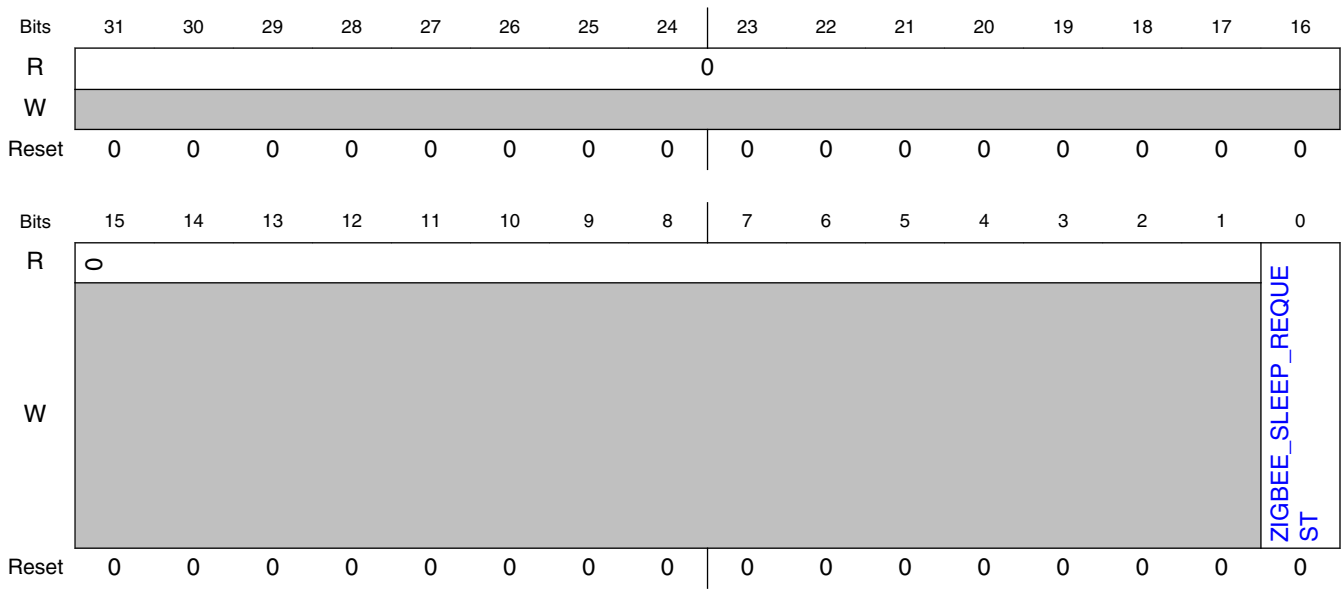
| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8<br>CCA2_CORR_THRESH | CCA Mode 2 Correlation Threshold<br>Correlator threshold used to qualify correlator peaks counted by CCA2_NUM_CORR_PEAKS. Correlator peaks exceeding this value increment CCA2_NUM_CORR_PEAKS. |
| 7<br>— | Reserved. |
| 6-4<br>CCA2_MIN_NUM_CORR_TH | CCA Mode 2 Threshold Number of Correlation Peaks<br>Programmable threshold to be compared against number of correlation peaks that exceeded CCA2_CORR_THRESH for detecting CCA mode 2. Number of peaks detected = CCA2_MIN_NUM_CORR_TH + 1; Example: If it is programmed to 3, CCA2 logic looks for at least 4 correlation peaks that crossed the threshold, to indicate channel is idle or busy. |
| 3-0<br>CCA2_NUM_CORR_PEAKS | CCA Mode 2 Number of Correlation Peaks Detected<br>Counts of number of peaks that crossed CCA2_CORR_THRESH in CCA Mode 2 operation |

## 64.5.2.5.1.20 DSM CONTROL (DSM_CTRL)

#### 64.5.2.5.1.20.1 Offset

| Register | Offset |
|---|---|
| DSM_CTRL | 4Ch |

#### 64.5.2.5.1.20.2 Diagram




#### 64.5.2.5.1.20.3 Fields

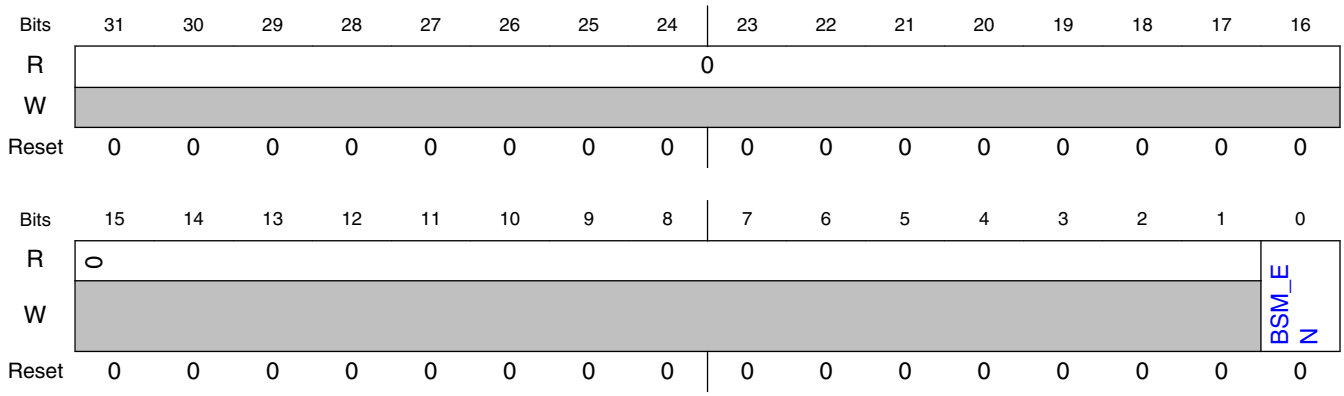
| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>ZIGBEE_SLEEP_REQUEST | 802.15.4 Deep Sleep Mode Request for Manual DSM<br>If 802.15.4 is assigned to Manual DSM (XCVR_CTRL[MAN_DSM_SEL] = 1), setting ZIGBEE_SLEEP_REQUEST to 1 enables Deep Sleep Mode (DSM) to be entered when the RSIM DSM_TIMER matches the RSIM MAN_SLEEP register. |

### 64.5.2.5.1.21 BSM CONTROL (BSM_CTRL)

#### 64.5.2.5.1.21.1 Offset

| Register | Offset |
|---|---|
| BSM_CTRL | 50h |

#### 64.5.2.5.1.21.2 Diagram




#### 64.5.2.5.1.21.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>BSM_EN | BSM Enable<br>0b - 802.15.4 Bit Streaming Mode Disabled<br>1b - 802.15.4 Bit Streaming Mode Enabled |

### 64.5.2.5.1.22 MAC SHORT ADDRESS FOR PAN1 (MACSHORTADDRS1)

#### 64.5.2.5.1.22.1 Offset

| Register | Offset |
|---|---|
| MACSHORTADDRS1 | 54h |

#### 64.5.2.5.1.22.2 Function

The MACSHORTADDRS1 register .

#### 64.5.2.5.1.22.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACSHORTADDRS1 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACPANID1 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.22.4 Fields

| Field | Function |
|---|---|
| 31-16<br>MACSHORTADDRS1 | MAC SHORT ADDRESS for PAN1<br>MAC Short Address for PAN1, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| 15-0<br>MACPANID1 | MAC PAN ID for PAN1<br>MAC PAN ID for PAN1. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID against this register. Also, if PANCORDNTR1=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |

## 64.5.2.5.1.23 MAC LONG ADDRESS 1 LSB (MACLONGADDRS1_LSB)

#### 64.5.2.5.1.23.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS1_LSB | 58h |

#### 64.5.2.5.1.23.2 Function

MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device.

#### 64.5.2.5.1.23.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.23.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS1_LSB | MAC LONG ADDRESS for PAN1 LSB<br>MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

### 64.5.2.5.1.24 MAC LONG ADDRESS 1 MSB (MACLONGADDRS1_MSB)

#### 64.5.2.5.1.24.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS1_MSB | 5Ch |

#### 64.5.2.5.1.24.2 Function

MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device.

#### 64.5.2.5.1.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.2.5.1.24.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS1_MSB | MAC LONG ADDRESS for PAN1 MSB<br>MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

## 64.5.2.5.1.25 DUAL PAN CONTROL (DUAL_PAN_CTRL)

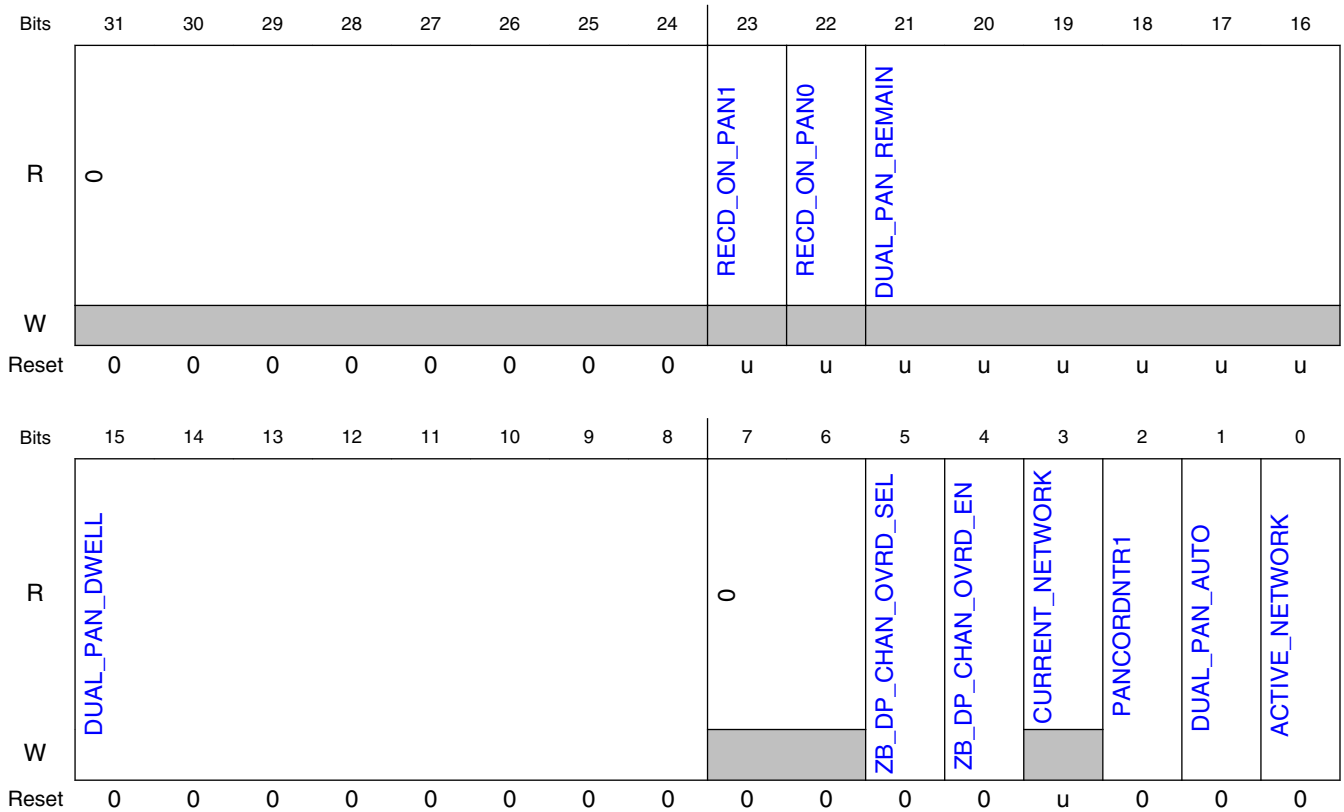
#### 64.5.2.5.1.25.1 Offset

| Register | Offset |
|---|---|
| DUAL_PAN_CTRL | 60h |

#### 64.5.2.5.1.25.2 Function

Dual PAN Control Register

### 64.5.2.5.1.25.3 Diagram




### 64.5.2.5.1.25.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23<br>RECD_ON_PAN1 | Last Packet was Received on PAN1<br>Indicates the packet which was just received, was received on PAN1. In Dual PAN mode operating on 2 different channels, RECD_ON_PAN1 will be set if CURRENT_NETWORK=1 when the packet was received, regardless of FILTERFAIL status. In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN1 will be set only if a valid packet was received on PAN1 (PAN1's FILTERFAIL_FLAG is deasserted). RECD_ON_PAN1 remains valid until the start of the next autoseqeuence. |
| 22<br>RECD_ON_PAN0 | Last Packet was Received on PAN0<br>Indicates the packet which was just received, was received on PAN0. In Dual PAN mode operating on 2 different channels, RECD_ON_PAN0 will be set if CURRENT_NETWORK=0 when the packet was received, regardless of FILTERFAIL status. In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN0 will be set only if a valid packet was received on PAN0 (PAN0's FILTERFAIL_FLAG is deasserted). RECD_ON_PAN0 remains valid until the start of the next autoseqeuence. |
| 21-16 | Time Remaining before next PAN switch in auto Dual PAN mode |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DUAL_PAN_REMAIN | This read-only register indicates time remaining before next PAN switch in auto Dual PAN mode. The units for this register, depend on the PRESCALER setting (bits [1:0]) in the DUAL_PAN_DWELL register. Tthe units are the same as the TIMEBASE determined by the prescaler, as per the following table:<br><br>DUAL_PAN_DWELL[1:0]=0b00 (TIMEBASE=0.5ms) DUAL_PAN_REMAIN UNITS=0.5ms<br>DUAL_PAN_DWELL[1:0]=0b01 (TIMEBASE=2.5ms) DUAL_PAN_REMAIN UNITS=2.5ms<br>DUAL_PAN_DWELL[1:0]=0b10 (TIMEBASE=10ms) DUAL_PAN_REMAIN UNITS=10ms<br>DUAL_PAN_DWELL[1:0]=0b11 (TIMEBASE=50ms) DUAL_PAN_REMAIN UNITS=50ms<br><br>The readback value indicates that between N-1 and N timebase units remain until the next PAN switch. For example, a DUAL_PAN_REMAIN readback value of 3, with a DUAL_PAN_DWELL PRESCALER setting of 2 (10ms), indicates that between 20ms (2*10ms) and 30ms (3*10ms), remain until the next automatic PAN switch. |
| 15-8<br><br>DUAL_PAN_DWELL | Dual PAN Channel Frequency Dwell Time<br><br>Channel Frequency Dwell Time. In Auto Dual PAN mode, hardware will toggle the PAN, after dwelling on the current PAN for the interval described below (assuming Preamble/SFD not detected). A write to DUAL_PAN_DWELL, always re-initilizes the DWELL TIMER to the programmed value. If a write to DUAL_PAN_DWELL occurs during an autosequence, the DWELL TIMER will begin counting down immediately. If a write to DUAL_PAN_DWELL occurs when there is no autosequence underway, the DWELL TIMER will not begin counting until the next autosequence begins; it will begin counting at the start of the sequence warmup.<br><br>DUAL_PAN_DWELL[1:0] select the timebase for the dwell time. There are 4 options:<br><br>DUAL_PAN_DWELL[1:0]=0b00 --> TIMEBASE=0.5ms<br>DUAL_PAN_DWELL[1:0]=0b01 --> TIMEBASE=2.5ms<br>DUAL_PAN_DWELL[1:0]=0b10 --> TIMEBASE=10ms<br>DUAL_PAN_DWELL[1:0]=0b11 --> TIMEBASE=50ms<br><br>DUAL_PAN_DWELL[7:2] select how many multiples of the time base to use, from 1 (DUAL_PAN_DWELL[7:2]=0) to 64 (DUAL_PAN_DWELL[7:2]=63). Given those definitions, the table below specifies the minimum and maximum dwell times for each TIMEBASE option:<br><br>DUAL_PAN_DWELL[1:0]=0b00 --> MIN_DWELL=0.5ms MAX_DWELL=32ms<br>DUAL_PAN_DWELL[1:0]=0b01 --> MIN_DWELL=2.5ms MAX_DWELL=160ms<br>DUAL_PAN_DWELL[1:0]=0b10 --> MIN_DWELL=10ms MAX_DWELL=640ms<br>DUAL_PAN_DWELL[1:0]=0b11 --> MIN_DWELL=50ms MAX_DWELL=3.2seconds<br><br>A write to DUAL_PAN_DWELL also causes the value of ACTIVE_NETWORK to get latched into the hardware. This latched value will be the starting point for the automatic dual-pan mode (i.e., start on PAN0 or on PAN1). The starting value takes effect immediately (if sequence is underway and DUAL_PAN_AUTO=1), or is otherwise delayed until sequence starts and DUAL_PAN_AUTO=1. |
| 7-6<br><br>— | Reserved. |
| 5<br><br>ZB_DP_CHAN_OVRD_SEL | Dual PAN Channel Override Selector<br><br>This bit works with **ZB_DP_CHAN_OVRD_EN** to allow one of the two Dual PAN channels to use Direct Frequency programming. See description for **ZB_DP_CHAN_OVRD_EN**. |
| 4<br><br>ZB_DP_CHAN_OVRD_EN | Dual PAN Channel Override Enable<br><br>In Dual PAN mode, in case there is a need to generate a frequency which may be offset from the 16 prescribed 5MHz-spaced channels, to, for example, avoid interference on one of the Dual PAN channels, a method has been provided to accomplish that, by designating one of the two PAN channels to use the transceiver's set of direct frequency-programming registers, instead of CHANNEL_NUMx. Programming |

*Table continues on the next page...*

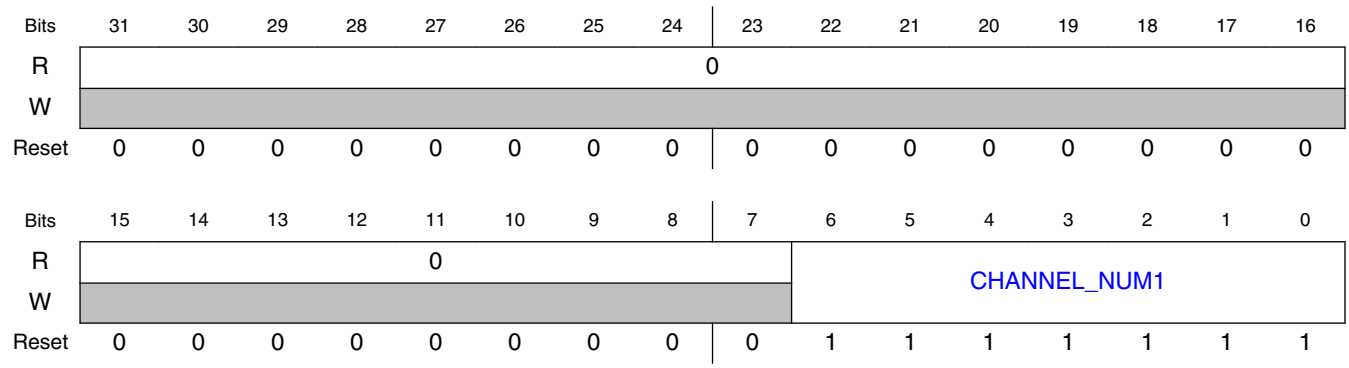
| Field | Function |
|---|---|
| | the direct frequency-programming registers -- integer, numerator, and denominator, allows an RF frequency to be selected with much more precision than the 5MHz granularity of the Zigbee mapped-channel registers, CHANNEL_NUM0 and CHANNEL_NUM1.<br><br>Two bits have been provided in 802.15.4 space to realize this feature: ZB_DP_CHAN_OVRD_SEL and ZB_DP_CHAN_OVRD_EN. When ZB_DP_CHAN_OVRD_EN=1, this enables one of the Dual PAN channels to use the direct frequency programming. The ZB_DP_CHAN_OVRD_SEL bit determines which channel uses the direct programming, according to the following:<br><br>If ZB_DP_CHAN_OVRD_EN=0, then PAN0 frequency is determined by register CHANNEL_NUM0[6:0] and PAN1 frequency is determined by register CHANNEL_NUM1[6:0].<br><br>If ZB_DP_CHAN_OVRD_EN=1 and ZB_DP_CHAN_OVRD_SEL=0, then PAN0 frequency is determined by DIRECT FREQUENCY PROGRAMMING and PAN1 frequency is determined by register CHANNEL_NUM1[6:0].<br><br>If ZB_DP_CHAN_OVRD_EN=1 and ZB_DP_CHAN_OVRD_SEL=1, then PAN0 frequency is determined by register CHANNEL_NUM0[6:0] and PAN1 frequency is determined by DIRECT FREQUENCY PROGRAMMING.<br><br>**NOTE:** Direct Frequency Programming is accomplished by setting the PLL's Integer, Numerator, and Denominator registers to the appropriate values for the desired RF frequency. |
| 3<br><br>CURRENT_NETWORK | Indicates which PAN is currently selected by hardware<br><br>This read-only bit indicates which PAN is currently selected by hardware in automatic Dual PAN mode<br>0b - PAN0 is selected<br>1b - PAN1 is selected |
| 2<br><br>PANCORDNTR1 | Device is a PAN Coordinator on PAN1<br><br>Device is a PAN Coordinator on PAN1. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| 1<br><br>DUAL_PAN_AUTO | Activates automatic Dual PAN operating mode<br><br>Activates automatic Dual PAN operating mode. In this mode, PAN-switching is controlled by hardware at a pre-programmed rate, determined by DUAL_PAN_DWELL.<br>0: Manual Dual PAN mode (or Single PAN mode).<br>1: Auto Dual PAN Mode<br><br>Whenever DUAL_PAN_AUTO=0, CURRENT_NETWORK=ACTIVE_NETWORK at all times. In other words, software directly controls which PAN is selected. Whenever DUAL_PAN_AUTO=1, CURRENT_NETWORK is controlled by hardware. |
| 0<br><br>ACTIVE_NETWORK | Active Network Selector<br><br>Selects the PAN on which to transceive, by activating a PAN parameter set (PAN0 or PAN1). In Manual Dual PAN mode (or Single PAN mode), this bit selects the active PAN parameter set (channel and addressing parameters) which governs all autosequences. In Auto Dual PAN mode, this bit selects the PAN on which to begin transceiving, latched at the point at which DUAL_PAN_DWELL register is written.<br>0b - Select PAN0<br>1b - Select PAN1 |

## 64.5.2.5.1.26 CHANNEL NUMBER 1 (CHANNEL_NUM1)

### *64.5.2.5.1.26.1 Offset*

| Register | Offset |
|---|---|
| CHANNEL_NUM1 | 64h |

#### 64.5.2.5.1.26.2 Diagram




#### 64.5.2.5.1.26.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>CHANNEL_NUM1 | Channel Number for PAN1<br>This is the mapped channel number used to transmit and receive Zigbee packets. This register applies to PAN1 only. CHANNEL_NUM1 should be in the range:<br>11 <= CHANNEL_NUM1 <= 26<br><br>**Note**: This register should not be programmed, and left in its default state, if Dual PAN mode is not in use. |

## 64.5.2.5.1.27 SAM CONTROL (SAM_CTRL)

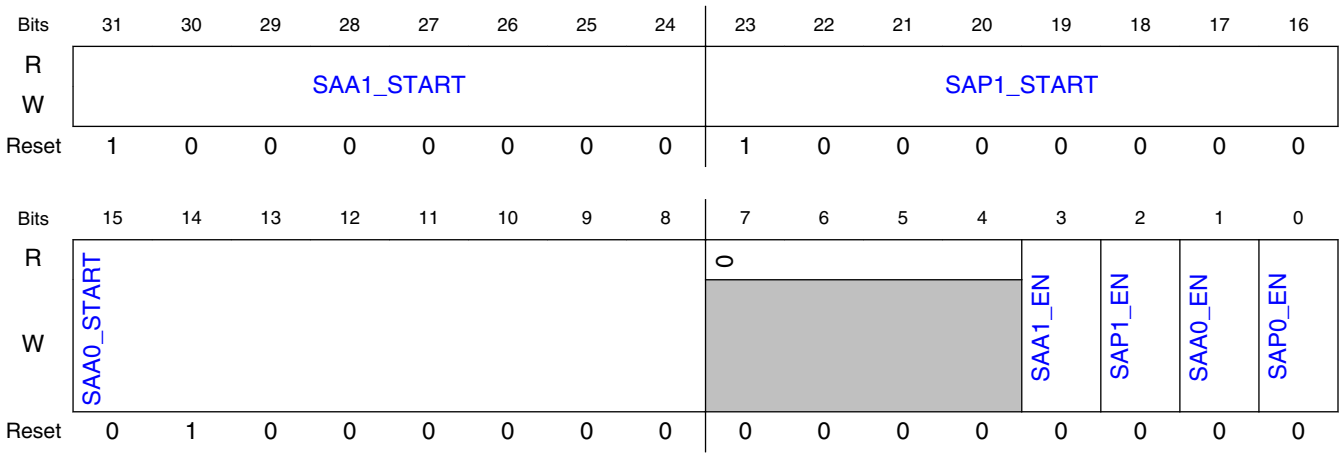
#### 64.5.2.5.1.27.1 Offset

| Register | Offset |
|---|---|
| SAM_CTRL | 68h |

#### 64.5.2.5.1.27.2 Function

Source Address Management Control Register

### 64.5.2.5.1.27.3 Diagram




### 64.5.2.5.1.27.4 Fields

| Field | Function |
|---|---|
| 31-24<br>SAA1_START | First Index of SAA1 partition |
| 23-16<br>SAP1_START | First Index of SAP1 partition |
| 15-8<br>SAA0_START | First Index of SAA0 partition |
| 7-4<br>— | Reserved. |
| 3<br>SAA1_EN | Enables SAA1 Partition of the SAM Table<br>0b - Disables SAA1 Partition<br>1b - Enables SAA1 Partition |
| 2<br>SAP1_EN | Enables SAP1 Partition of the SAM Table<br>0b - Disables SAP1 Partition<br>1b - Enables SAP1 Partition |
| 1<br>SAA0_EN | Enables SAA0 Partition of the SAM Table<br>0b - Disables SAA0 Partition<br>1b - Enables SAA0 Partition |
| 0<br>SAP0_EN | Enables SAP0 Partition of the SAM Table<br>0b - Disables SAP0 Partition<br>1b - Enables SAP0 Partition |

## 64.5.2.5.1.28 SOURCE ADDRESS MANAGEMENT TABLE (SAM_TABLE)

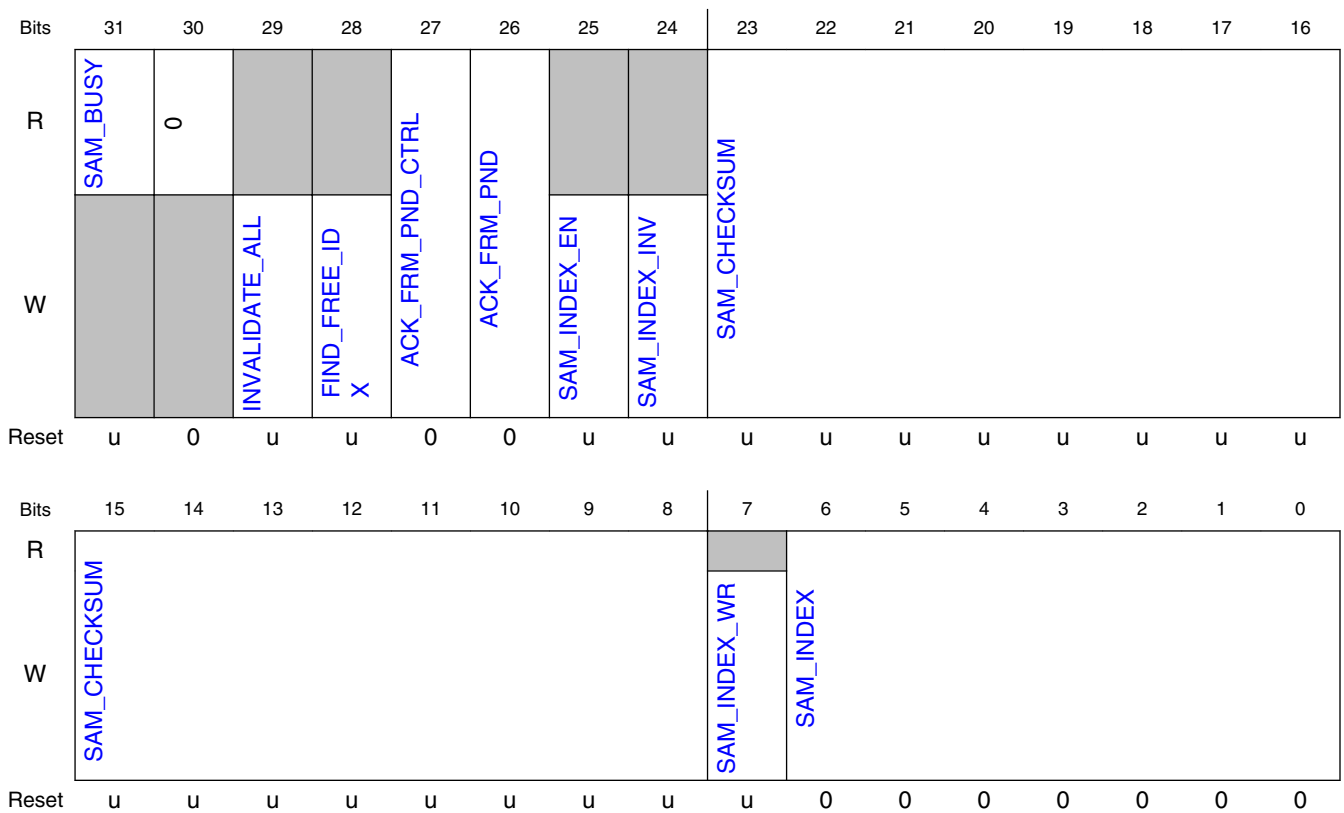
### 64.5.2.5.1.28.1 Offset

| Register | Offset |
|---|---|
| SAM_TABLE | 6Ch |

### 64.5.2.5.1.28.2 Function

Source Address Management Table

### 64.5.2.5.1.28.3 Diagram




### 64.5.2.5.1.28.4 Fields

| Field | Function |
|---|---|
| 31<br>SAM_BUSY | SAM Table Update Status Bit<br>Hardware is in the process of updating the Source Address table, either in response to a poll indication from the packet processor, or due to software setting FIND_FREE_IDX=1. In the latter case, software should poll SAM_BUSY until low before accessing the "First Free Index" registers. Read-only bit. |
| 30<br>— | Reserved. |

*Table continues on the next page...*

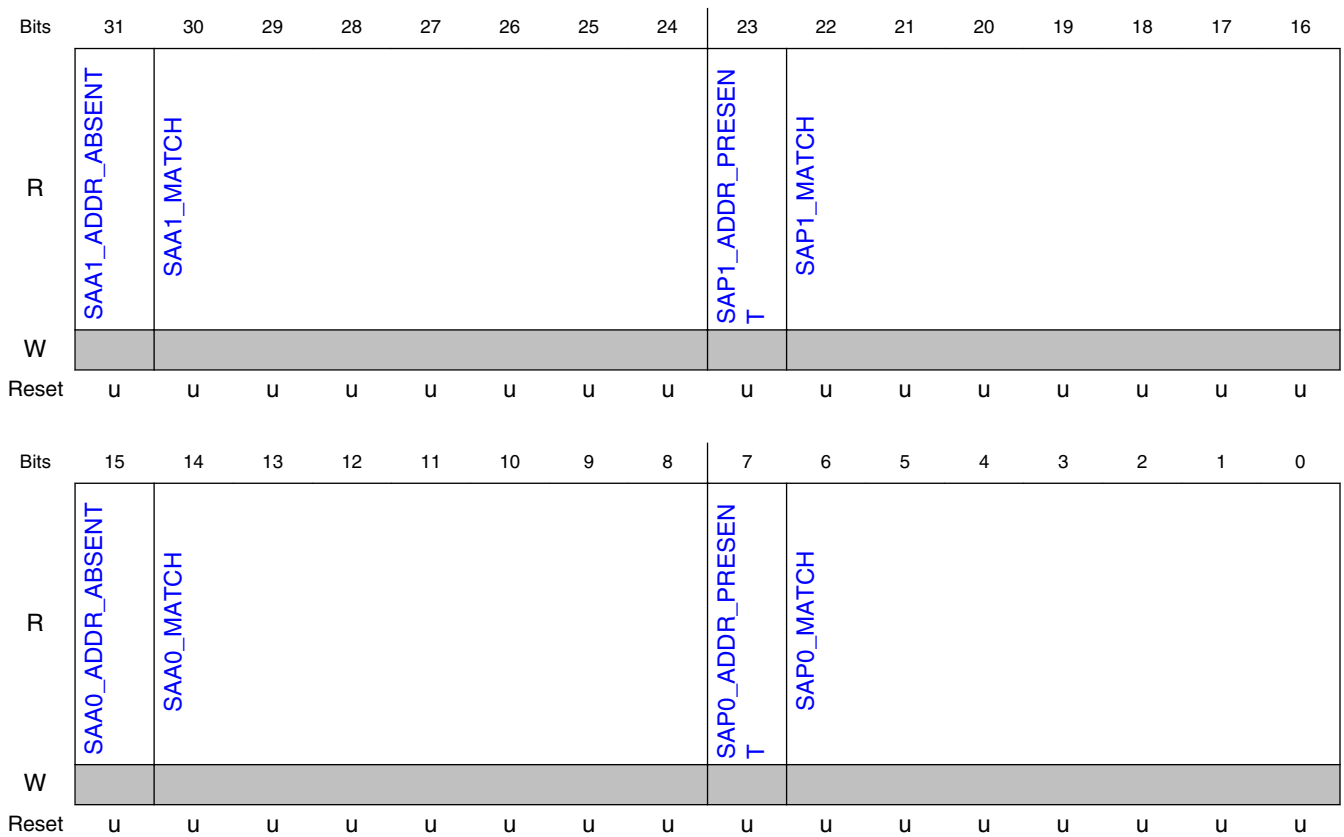
| Field | Function |
|---|---|
| 29<br>INVALIDATE_ALL | Invalidate Entire SAM Table<br>Writing a 1 to this bit clears all 128 Valid bits. Invalidates the entire table. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| 28<br>FIND_FREE_IDX | Find First Free Index<br>After modifying Valid bits (enabling or invalidating), write this bit to 1 to force hardware to update the "First Free Index" registers to account for the changed Valid bits. This hardware update process takes 4us. Software can poll SAM_BUSY to determine when the table update is complete. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| 27<br>ACK_FRM_PND_CTRL | Manual Control for AutoTxAck FramePending field<br>0b - the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet is determined by hardware<br>1b - the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet tracks ACK_FRM_PEND |
| 26<br>ACK_FRM_PND | State of AutoTxAck FramePending field when SAM Accelleration is Disabled<br>Software can take manual control of the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet, by setting ACK_FRM_PND_CTRL=1; in that case FramePending will track the state of this bit. The FramePending field also tracks this bit if Source Address Management is completely disabled, i.e.,<br>SAP0_EN=SAA0_EN=SAP1_EN=SAA1_EN=0<br>Otherwise, the FramePending field is determined by Source Address Management (SAM) hardware. |
| 25<br>SAM_INDEX_EN | Enable the SAM table index selected by SAM_INDEX |
| 24<br>SAM_INDEX_INV | Invalidate the SAM table index selected by SAM_INDEX |
| 23-8<br>SAM_CHECKSUM | Software-computed source address checksum, to be installed into a table index<br>Software-computed source address checksum, to be installed into a table index. The value on SAM_CHECKSUM[15:0] can be installed into the table with a single, atomic 32-bit write; in that case, the write data would contain the desired SAM_INDEX[6:0] and SAM_CHECKSUM[15:0], and SAM_INDEX_WR=1.<br>If SAM_INDEX_WR=0, then the SAM_INDEX[6:0] register is written, but the checksum is *not* written to the table.<br>The readback value of SAM_CHECKSUM[15:0] is the contents of the SAM Table at the location pointed to by SAM_INDEX[6:0]. To readback from a specific table index, software should first write the desired index to SAM_INDEX[6:0], and then read back the checksum from the table on SAM_CHECKSUM[15:0]. |
| 7<br>SAM_INDEX_WR | Enables SAM Table Contents to be updated<br>For 32-bit writes, SAM_INDEX_WR must be set to indicate that the table entry specified by SAM_INDEX[6:0] is to be written; if SAM_INDEX_WR=0, the table entry is not written, but the SAM_INDEX[6:0] register is updated. For 8-bit writes, this bit is ignored. |
| 6-0<br>SAM_INDEX | Contains the SAM table index to be enabled or invalidated<br>Contains the table index to be enabled or invalidated. Software must ensure that the index is within the range of the desired partition. |

# 64.5.2.5.1.29 SOURCE ADDRESS MANAGEMENT MATCH (SAM_MATCH)

## *64.5.2.5.1.29.1 Offset*

| Register | Offset |
|---|---|
| SAM_MATCH | 70h |

## *64.5.2.5.1.29.2 Diagram*




## *64.5.2.5.1.29.3 Fields*

| Field | Function |
|---|---|
| 31<br>SAA1_ADDR_ABSENT | A Checksum Match is Absent in the SAP1 Partition of the SAM Table |
| 30-24<br>SAA1_MATCH | Index in the SAA1 Partition of the SAM Table corresponding to the first checksum match |
| 23 | A Checksum Match is Present in the SAP1 Partition of the SAM Table |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SAP1_ADDR_PRESENT | |
| 22-16<br>SAP1_MATCH | Index in the SAP1 Partition of the SAM Table corresponding to the first checksum match |
| 15<br>SAA0_ADDR_ABSENT | A Checksum Match is Absent in the SAA0 Partition of the SAM Table |
| 14-8<br>SAA0_MATCH | Index in the SAA0 Partition of the SAM Table corresponding to the first checksum match |
| 7<br>SAP0_ADDR_PRESENT | A Checksum Match is Present in the SAP0 Partition of the SAM Table |
| 6-0<br>SAP0_MATCH | Index in the SAP0 Partition of the SAM Table corresponding to the first checksum match |

## 64.5.2.5.1.30 SAM FREE INDEX (SAM_FREE_IDX)

### *64.5.2.5.1.30.1 Offset*

| Register | Offset |
|---|---|
| SAM_FREE_IDX | 74h |

### *64.5.2.5.1.30.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SAA1_1ST_FREE_IDX | | | | | | | | SAP1_1ST_FREE_IDX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SAA0_1ST_FREE_IDX | | | | | | | | SAP0_1ST_FREE_IDX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### *64.5.2.5.1.30.3 Fields*

| Field | Function |
|---|---|
| 31-24 | First non-enabled (invalid) index in the SAA1 partition |

*Table continues on the next page...*

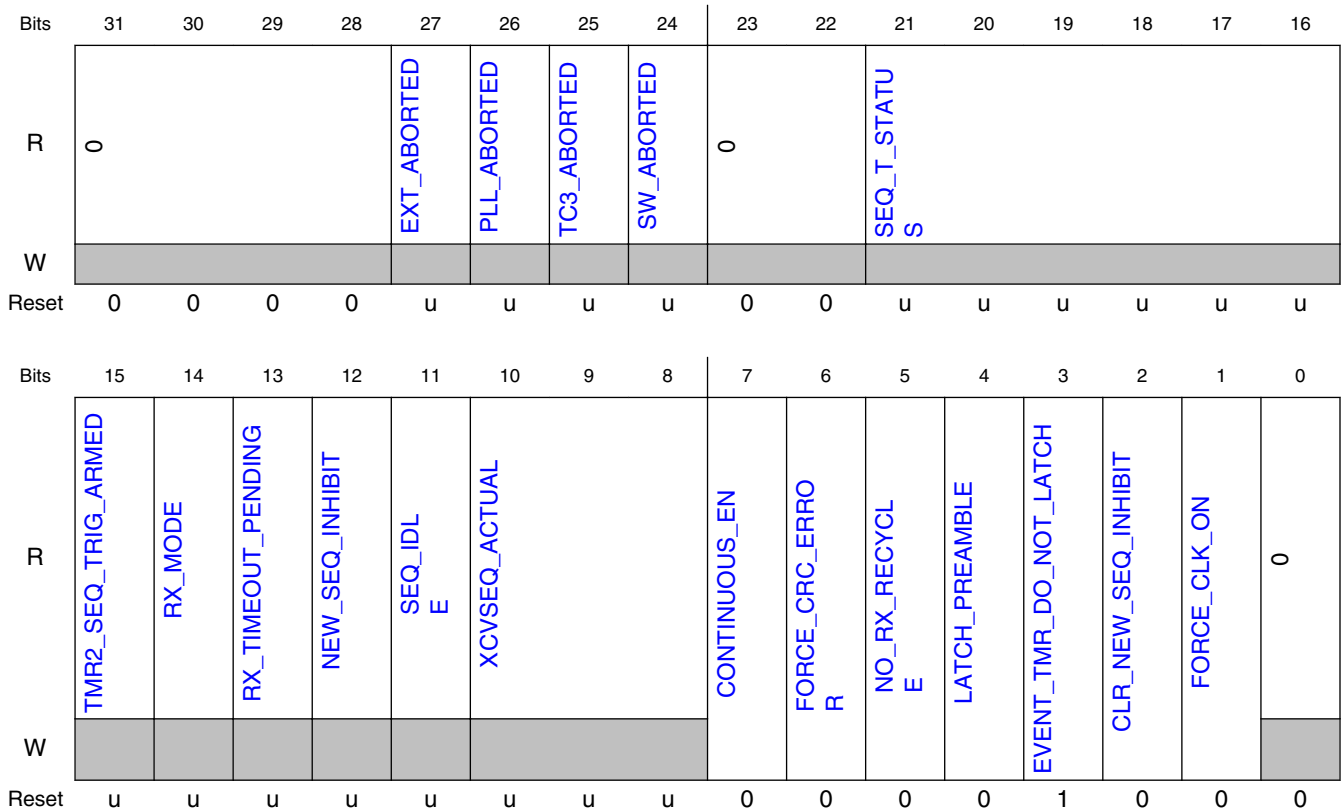
| Field | Function |
|---|---|
| SAA1_1ST_FREE_IDX | |
| 23-16<br>SAP1_1ST_FREE_IDX | First non-enabled (invalid) index in the SAP1 partition |
| 15-8<br>SAA0_1ST_FREE_IDX | First non-enabled (invalid) index in the SAA0 partition |
| 7-0<br>SAP0_1ST_FREE_IDX | First non-enabled (invalid) index in the SAP0 partition |

## 64.5.2.5.1.31 SEQUENCE CONTROL AND STATUS (SEQ_CTRL_STS)

### *64.5.2.5.1.31.1 Offset*

| Register | Offset |
|---|---|
| SEQ_CTRL_STS | 78h |

### 64.5.2.5.1.31.2 Diagram




### 64.5.2.5.1.31.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>EXT_ABORTED | Autosequence has terminated due to a Wake-On-Radio command<br>when asserted, indicates that the autosequence has terminated due to an external event, such as a Wake-On-Radio stop command. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 26<br>PLL_ABORTED | Autosequence has terminated due to an PLL unlock event<br>when asserted, indicates that the autosequence has terminated due to an PLL unlock event. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 25<br>TC3_ABORTED | autosequence has terminated due to an TMR3 timeout<br>when asserted, indicates that the autosequence has terminated due to an TC3 (TMR3) timeout during a receive operation. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 24<br>SW_ABORTED | Autosequence has terminated due to a Software abort. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | when asserted, indicates that the autosequence has terminated due to an Software abort. Software can abort any programmed autosequence by writing Sequence I to XCVSEQ. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 23-22<br>— | Reserved. |
| 21-16<br>SEQ_T_STATUS | Status of the just-completed or ongoing Sequence T or Sequence TR<br>Status of the just-completed (or ongoing) Sequence T or Sequence TR autosequence. This register is valid at all times during, and after, the Sequence T or Sequence TR. Not valid for other types of autosequences. This is a read-only register. The bits of this register map to status, according to the following table:<br>[0] 1st CCA complete (CCABFRTX=1)<br>[1] 2nd CCA complete (SLOTTED=1)<br>[2] Tx operation complete<br>[3] Rx Recycle occurred (Sequence TR only)<br>[4] Rx operation complete (Sequence TR only)<br>[5] TxAck operation complete(Sequence TR only) |
| 15<br>TMR2_SEQ_TRIG_ARMED | indicates that TMR2 has been programmed and is armed to trigger a new autosequence<br>when asserted, indicates that TMR2 has been programmed and is "armed" to trigger a new autosequence, when Zigbee Sequence Manager timer-triggering mode is selected (i.e., TMRTRIGEN=1). When timer-triggering mode is selected, TMR2 must be re-programmed (using either T2CMP or T2PRIMECMP), in advance of each new sequence. Once TMR2 is programmed, this bit will be asserted, and will remain asserted until the new sequence commences (at TMR2 match). Hardware will deassert this bit when the new sequence starts. When TMRTRIGEN=0, this bit should be ignored. Read-only bit. |
| 14<br>RX_MODE | RX Operation in Progress<br>when asserted, this Sequence Manager Output indicates that an RX operation is in progress. An RX operation can be part of a complex transmit autosequence such as a Sequence TR. CCA and ED operations are considered RX operations, during which rx_mode is asserted. Read-only bit. |
| 13<br>RX_TIMEOUT_PENDING | Indicates a TMR3 RX Timeout is Pending<br>when asserted, indicates that a TMR3 timeout (RX timeout) flag has been set by Hardware, but the Sequence Manager has not yet aborted because an RX operation is not currently underway. This would be the case, for example, during a Sequence TR, if a TMR3 timeout were to occur during the transmit operation of this sequence; the sequence would not be aborted by Hardware until the receive operation begins. This bit will always be 0 if TC3TMOUT=0. Read-only bit. |
| 12<br>NEW_SEQ_INHIBIT | New Sequence Inhibit<br>When asserted, indicates that a new programmed autosequence has commenced (TMR2 match has occurred if TMRTRIGEN=1). Once this bit is asserted, software is blocked from commanding any "new" autosequences (other than Sequence I to abort the current sequence), until the current sequence completes. Hardware will ignore a sequence-change command from software while this bit is asserted. Hardware will automatically deassert this bit once the sequence completes. Read-only bit. |
| 11<br>SEQ_IDLE | ZSM Sequence Idle Indicator |
| 10-8<br>XCVSEQ_ACTUAL | Indicates the programmed sequence that has been recognized by the ZSM Sequence Manager<br>Reflects the programmed sequence that has been recognized by the Zigbee Sequence Manager. Takes into account the fact that sequence-change commands from software are ignored while a sequence is underway (see NEW_SEQ_INHIBIT). Read-only bits. |
| 7 | Enable Continuous TX or RX Mode |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CONTINUOUS_EN | Continuous Mode Enable (Continuous TX or RX).<br>**Note:** Dual PAN mode should not be engaged in Continuous TX or RX modes.<br>0b - normal operation<br>1b - Continuous TX or RX mode is enabled (depending on XCVSEQ setting). |
| 6<br>FORCE_CRC_ERROR | Induce a CRC Error in Transmitted Packets<br>0b - normal operation<br>1b - Force the next transmitted packet to have a CRC error |
| 5<br>NO_RX_RECYCLE | Disable Automatic RX Sequence Recycling<br>when asserted, prevents the Zigbee Sequence Manager (ZSM) from automatically re-starting (recycling) the receiver when a packet is received which results in a FilterFail or CRC failure. Normally, on a RX recycle, the ZSM returns to the RX_WU (warmup) state, and then resumes from there with a new, foreshortened, Rx warmup, in search of a new preamble. When this bit is set, the Sequence Manager will instead return to idle state, and issue a SEQIRQ, after a FilterFail or CRC failure. |
| 4<br>LATCH_PREAMBLE | Stickiness Control for Preamble Detection<br>0b - Don't make PREAMBLE_DET and SFD_DET bits of PHY_STS (SEQ_STATE) Register "sticky", i.e, these status bits reflect the realtime, dynamic state of preamble_detect and sfd_detect<br>1b - Make PREAMBLE_DET and SFD_DET bits of PHY_STS (SEQ_STATE) Register "sticky", i.e.,occurrences of preamble and SFD detection are latched and held until the start of the next autosequence |
| 3<br>EVENT_TMR_DO_NOT_LATCH | Overrides the automatic hardware latching of the Event Timer<br>when asserted, overrides the automatic hardware latching of the Event Timer that prevents the timer from updating while software reads the 3 Event Timer bytes. This allows the Event Timer LS byte to continue to update without reading the upper 2 bytes. Overriding the automatic latching of the Event Timer should be used with caution, as it can allow the Event Timer lower bytes to get out-of-sync with the upper bytes. However, it can be useful when polling the Event Timer LS byte for a value that is just a few counts in the future. |
| 2<br>CLR_NEW_SEQ_INHIBIT | Overrides the automatic hardware locking of the programmed XCVSEQ while an autosequence is underway<br>when asserted, overrides the automatic hardware locking of the programmed XCVSEQ while an autosequence is underway. Asserting this feature will allow software to change the programmed autosequence "on-the-fly", without aborting and returning to idle between sequences. Overriding the hardware lockout of XCVSEQ should be used with caution, since the Sequence Manager is not designed (or verified) for manual state transitions between one type of autosequence and other (i.e., Sequence T -> Sequence R). |
| 1<br>FORCE_CLK_ON | Force On 802.15.4 phy_gck<br>Force On 802.15.4 phy_gck<br>0b - Allow TSM to control 802.15.4 phy_gck, for minimum power consumption (default)<br>1b - Force on 802.15.4 phy_gclk at all times, for debug purposes only |
| 0<br>— | Reserved. |

## 64.5.2.5.1.32 ACK DELAY (ACKDELAY)

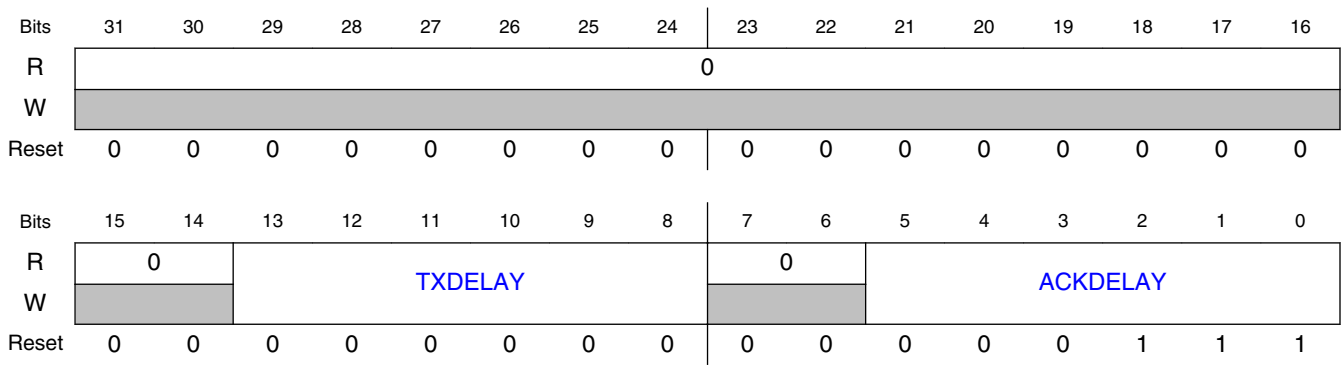
#### 64.5.2.5.1.32.1 Offset

| Register | Offset |
|---|---|
| ACKDELAY | 7Ch |

#### 64.5.2.5.1.32.2 Function

Provides a fine-tune adjustment of the time delay between Rx warmdown and the beginning of Tx warmup for an autoTxAck packet.

#### 64.5.2.5.1.32.3 Diagram




#### 64.5.2.5.1.32.4 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved. |
| 13-8<br>TXDELAY | TX Delay<br>Provides a fine-tune adjustment of the time delay between post-CCA Rx warm-down and the beginning of Tx warm-up for an Tx (non-Ack) packet. TXDELAY register will apply in both SLOTTED and UNSLOTTED modes, but only to T sequences (e.g., T, TR, and T(R) ), not TxAck operations. This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Max TXDELAY = 0x1F.<br>Min TXDELAY = 0x2D. |
| 7-6<br>— | Reserved. |
| 5-0<br>ACKDELAY | ACK Delay |

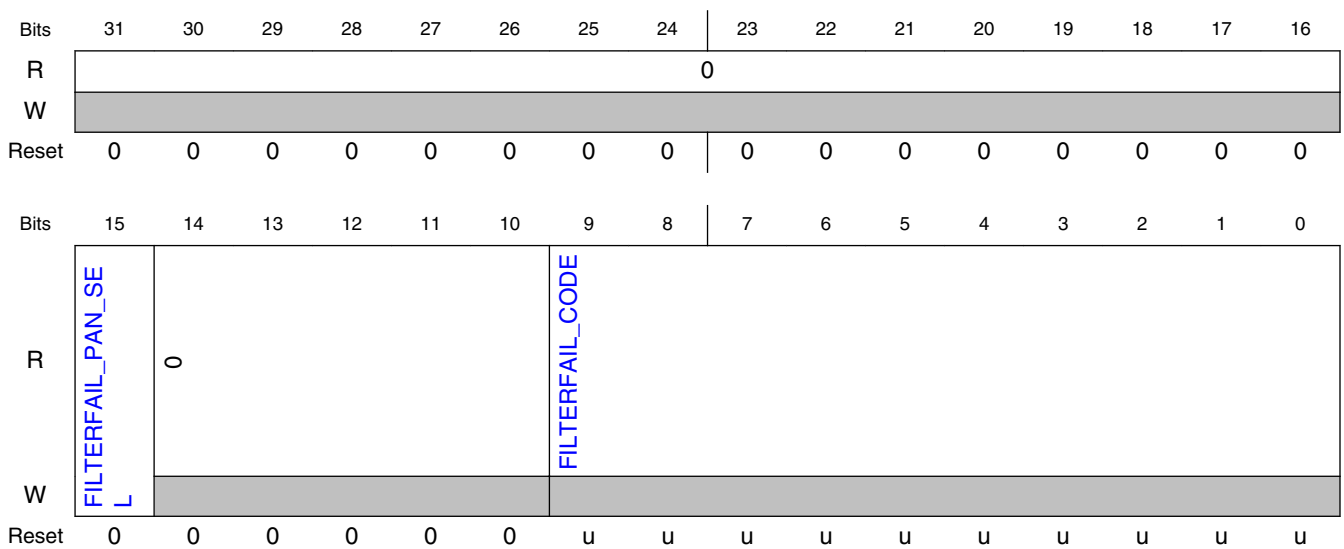
| Field | Function |
|---|---|
| | Provides a fine-tune adjustment of the time delay between Rx warmdown and the beginning of Tx warmup for an Tx Acknowledge packet. ACKDELAY register will apply to both SLOTTED and UNSLOTTED TxAck, but only to TxAck (not T sequences). This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Max ACKDELAY = 0x1F.<br>Min ACKDELAY = 0x2D. |

## 64.5.2.5.1.33 FILTER FAIL CODE (FILTERFAIL_CODE)

### *64.5.2.5.1.33.1 Offset*

| Register | Offset |
|---|---|
| FILTERFAIL_CODE | 80h |

### *64.5.2.5.1.33.2 Diagram*




### *64.5.2.5.1.33.3 Fields*

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |

*Table continues on the next page...*

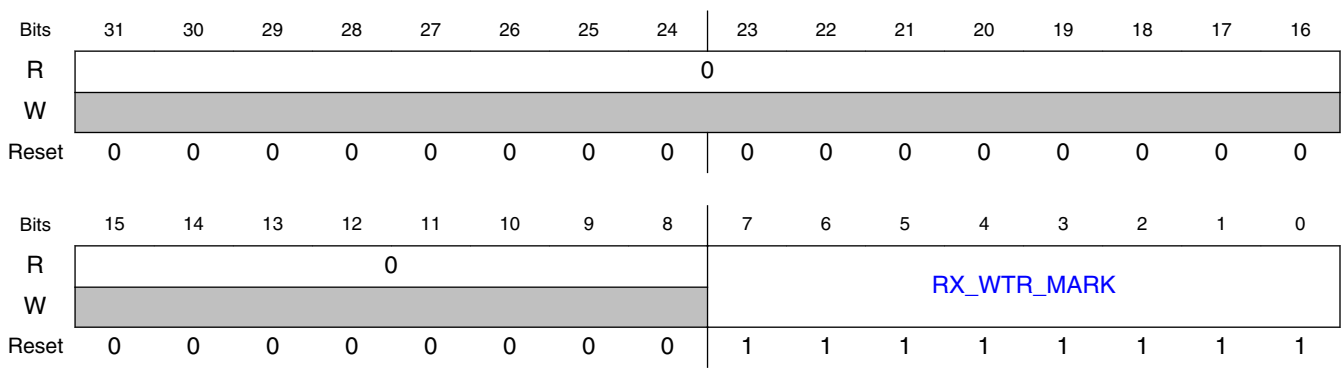
| Field | Function |
|---|---|
| 15<br>FILTERFAIL_PAN_SEL | PAN Selector for Filter Fail Code<br>0b - FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN0<br>1b - FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN1 |
| 14-10<br>— | Reserved. |
| 9-0<br>FILTERFAIL_CODE | Filter Fail Code<br>Code inidicating what condition, or conditions, caused the Packet Processor to reject the just-received packet. The bits of FILTERFAIL_CODE indicate the reason for packet rejection. For a description of the individual FILTERFAIL_CODE bits, see the 802.15.4 Link Layer Controller chapter. |

## 64.5.2.5.1.34 RECEIVE WATER MARK (RX_WTR_MARK)

### *64.5.2.5.1.34.1 Offset*

| Register | Offset |
|---|---|
| RX_WTR_MARK | 84h |

### *64.5.2.5.1.34.2 Diagram*




### *64.5.2.5.1.34.3 Fields*

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>RX_WTR_MARK | RECEIVE WATER MARK<br>Receive byte count (octets) needed to trigger a RXWTRMRKIRQ interrupt . A setting of 0 generates an interrupt at end of the Frame Length field (first byte after SFD). A setting of 1 generates an interrupt after the first byte of Frame Control Field, etc. |

## 64.5.2.5.1.35 SLOT PRELOAD (SLOT_PRELOAD)

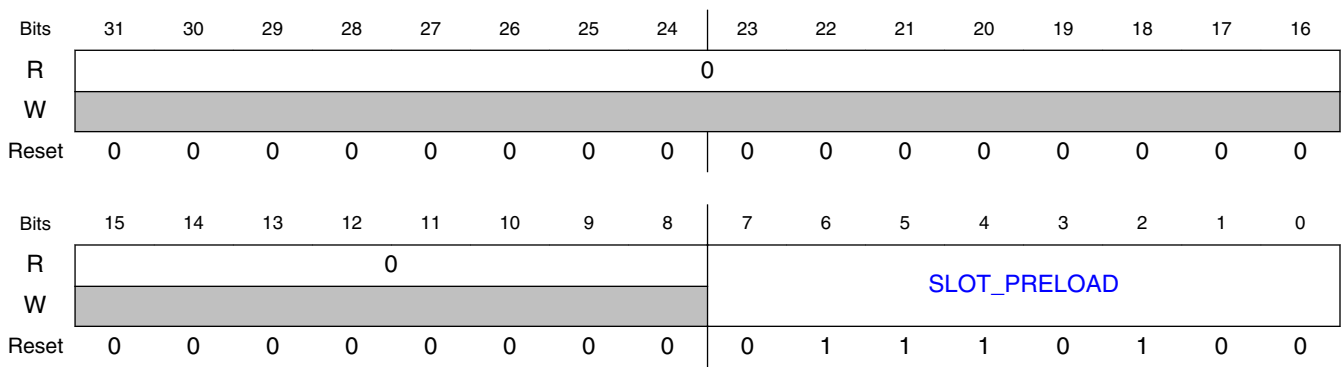
### *64.5.2.5.1.35.1 Offset*

| Register | Offset |
|---|---|
| SLOT_PRELOAD | 8Ch |

### *64.5.2.5.1.35.2 Function*

Slotted Mode Preload

### *64.5.2.5.1.35.3 Diagram*




### *64.5.2.5.1.35.4 Fields*

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SLOT_PRELOAD | Slotted Mode Preload<br>This register represents the number that gets loaded into the slot_timer at SFD detect, which ultimately determines when the next slot boundary will occur. Due to processing delays within the analog front-end and digital modem, the point at which SFD is detected by the modem, is delayed relative to over-the-air timing. This register setting compensates for that delay. This timing parameter is critical for the Sequence R autosequence in slotted mode, when an automatic TxAck is required. |

## 64.5.2.5.1.36 802.15.4 SEQUENCE STATE (SEQ_STATE)

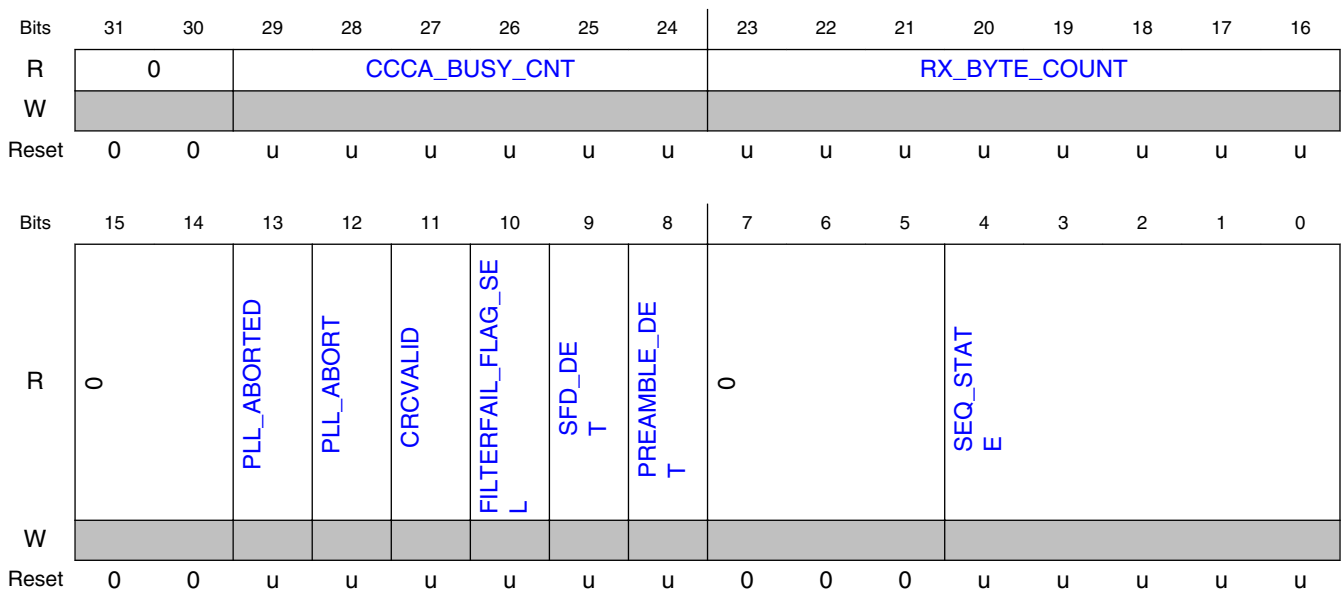
#### 64.5.2.5.1.36.1 Offset

| Register | Offset |
|---|---|
| SEQ_STATE | 90h |

#### 64.5.2.5.1.36.2 Function

802.15.4 Sequence State Register

#### 64.5.2.5.1.36.3 Diagram




#### 64.5.2.5.1.36.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-24<br>CCCA_BUSY_CNT | Number of CCA Measurements resulting in Busy Channel<br>For Sequence CCCA mode only, this register indicates the number of "busy" CCA attempts which occurred during the autosequence, before the channel was detected to be idle. This register can also be read in real-time (during the autosequence) to determine how many busy CCA attempts have occurred to that point. The register saturates at 63 (i.e, if there are more than 63 busy attempts, the register will continue to read 63). This register is automatically cleared to zero by hardware when the next autosequence commences. Read-only register. |
| 23-16<br>RX_BYTE_COUNT | Realtime Received Byte Count<br>During packet reception, this read-only register is a real-time indicator of the number of bytes that have been received. This register will read 0 until SFD and PHR have been received. It will read 1 after the first byte of Frame Control Field has been received, etc. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-14<br>— | Reserved. |
| 13<br>PLL_ABORTED | Autosequence has terminated due to an PLL unlock event<br>when asserted, indicates that the autosequence has terminated due to an PLL unlock event. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. This bit is a read-only mirror of the register bit of the same name in the ABORT_STS (SEQ_CTRL_STS) register. |
| 12<br>PLL_ABORT | Raw PLL Abort Signal<br>This bit reflects the instantaneous, consolidated status of the PLL unlock detection circuits; if asserted high, indicates that at least one of the three PLL unlock detect mechanisms is currently reporting an unlocked condition. |
| 11<br>CRCVALID | CRC Valid Indicator<br>Cyclic Redundancy Check Valid: This flag indicates the compare result between the FCS field, in the most-recently received frame, and the internally calculated CRC value. This flag is cleared at next receiver warm up.<br>0b - Rx FCS != calculated CRC (incorrect)<br>1b - Rx FCS = calculated CRC (correct) |
| 10<br>FILTERFAIL_FLAG_SEL | Consolidated Filter Fail Flag<br>0: The incoming, or just-received packet, passed packet filtering rules.<br>1: The incoming, or just-received packet, failed packet filtering rules<br>When FILTERFAIL_FLAG_SEL=1, a non-zero FILTERFAIL_CODE is present (see FILTERFAIL_CODE registers).<br>In Dual PAN mode, FILTERFAIL_FLAG_SEL applies to either or both networks, as follows:<br>**A:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=0, FILTERFAIL_FLAG_SEL applies to PAN0.<br>**B:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=1, FILTERFAIL_FLAG_SEL applies to PAN1.<br>**C:** If PAN0 and PAN1 occupy the same channel, FILTERFAIL_FLAG_SEL is the logical 'AND' of the individual PANs' FILTERFAIL_FLAG bits. |
| 9<br>SFD_DET | SFD Detected<br>0: an 802.15.4 preamble-and-SFD have not been detected.<br>1: An 802.15.4 preamble-and-SFD have been detected.<br>The function of this read-only bit depends on the setting of the LATCH_PREAMBLE bit of the SEQ_MGR_CTRL register. If LATCH_PREAMBLE=1, any preamble-and-SFD detection during a Sequence R (even false detections), will set this bit, and it will remain set (sticky) until the start of the next autosequence. If LATCH_PREAMBLE=0, this bit is not sticky, and reflects the instantaneous state of the SFD-detection circuit; for false SFD, the bit will clear when the false nature of the SFD is recognized (i.e., an RX recycle). When LATCH_PREAMBLE=0, SFD_DET should be considered valid only while an autosequence is underway. |
| 8<br>PREAMBLE_DET | Preamble Detected<br>0: an 802.15.4 preamble has not been detected.<br>1: An 802.15.4 preamble has been detected.<br>The function of this read-only bit depends on the setting of the LATCH_PREAMBLE bit of the SEQ_MGR_CTRL register. If LATCH_PREAMBLE=1, any preamble detection during a Sequence R (even false detections), will set this bit, and it will remain set (sticky) until the start of the next autosequence. If LATCH_PREAMBLE=0, this bit is not sticky, and reflects the instantaneous state of the |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | preamble-detection circuit; for false preambles, the bit will clear when the false nature of the preamble is recognized. When LATCH_PREAMBLE=0, PREAMBLE_DET should be considered valid only while an autosequence is underway. |
| 7-5<br>— | Reserved. |
| 4-0<br>SEQ_STATE | ZSM Sequence State<br>This read-only register reflects the instantaneous state of the 802.15.4 Sequence Manager |

## 64.5.2.5.1.37 TIMER PRESCALER (TMR_PRESCALE)

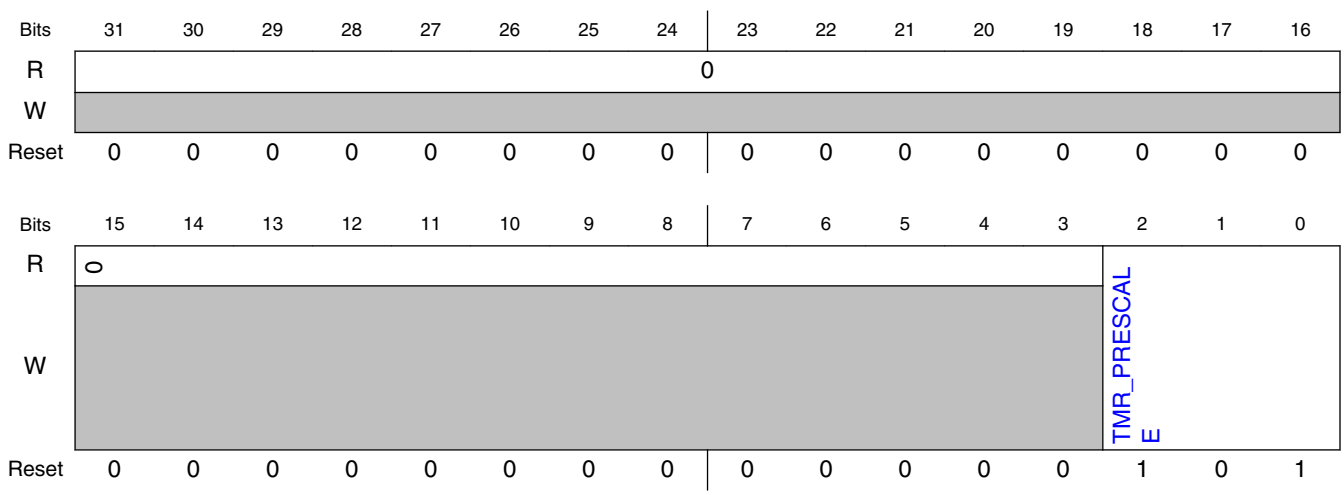
### *64.5.2.5.1.37.1 Offset*

| Register | Offset |
|---|---|
| TMR_PRESCALE | 94h |

### *64.5.2.5.1.37.2 Function*

Timer Prescaler Control Register

### *64.5.2.5.1.37.3 Diagram*




### *64.5.2.5.1.37.4 Fields*

| Field | Function |
|---|---|
| 31-3 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 2-0<br>TMR_PRESCALE | Timer Prescaler<br>Timer Prescaler. Establishes the Event Timer clock rate (and also maximum timer duration)<br><br>**Note:**To take advantage of the EVENT_TMR Fractional bits for 802.15.4 DSM mode, only the default setting ("5", or 62.5KHz) is allowed.<br>000b - Reserved<br>001b - Reserved<br>010b - 500kHz (33.55 S)<br>011b - 250kHz (67.11 S)<br>100b - 125kHz (134.22 S)<br>101b - 62.5kHz (268.44 S) -- default<br>110b - 31.25kHz (536.87 S)<br>111b - 15.625kHz (1073.74 S) |

## 64.5.2.5.1.38 LENIENCY LSB (LENIENCY_LSB)

### *64.5.2.5.1.38.1 Offset*

| Register | Offset |
|---|---|
| LENIENCY_LSB | 98h |

### *64.5.2.5.1.38.2 Function*

Packet Processor Leniency Bits (LSB)

### *64.5.2.5.1.38.3 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LENIENCY_LSB | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LENIENCY_LSB | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.5.2.5.1.38.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LENIENCY_LSB | Leniency LSB Register<br>The Packet Processor performs filtering on all received packets, in order to determine whether the packet is intended for the device. The packet filtering is based on rules. In case any of the packet filtering rules need to be overriden, a 40-bit "leniency register" has been provided. When the leniency register is programmed to its default value (0), all hardware packet filtering rules are in effect, and if an incoming packet violates any rule, a "Filter Fail" will occur (packet will be rejected). When a given leniency register bit is asserted, the packet filtering rule assigned to that bit will not be in effect, and if any incoming packet violates that rule (but no other rules), then a "Filter Fail" will not occur, the packet will not be rejected, the packet will be treated as "intended for the device", and software will be notified of the incoming packet. For a desciption of the function of the individual leniency bits, see the 802.15.4 Link Layer Controller chapter. |

### 64.5.2.5.1.39 LENIENCY MSB (LENIENCY_MSB)

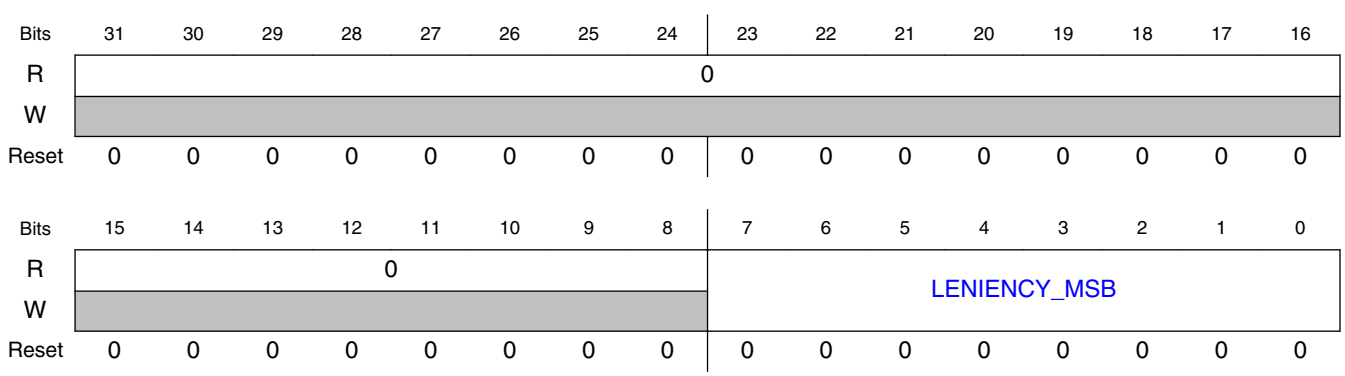
#### 64.5.2.5.1.39.1 Offset

| Register | Offset |
|---|---|
| LENIENCY_MSB | 9Ch |

#### 64.5.2.5.1.39.2 Function

Packet Processor Leniency Bits (MSB)

#### 64.5.2.5.1.39.3 Diagram




#### 64.5.2.5.1.39.4 Fields

| Field | Function |
|---|---|
| 31-8 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 7-0<br>LENIENCY_MSB | Leniency MSB Register<br>The Packet Processor performs filtering on all received packets, in order to determine whether the packet is intended for the device. The packet filtering is based on rules. In case any of the packet filtering rules need to be overriden, a 40-bit "leniency register" has been provided. When the leniency register is programmed to its default value (0), all hardware packet filtering rules are in effect, and if an incoming packet violates any rule, a "Filter Fail" will occur (packet will be rejected). When a given leniency register bit is asserted, the packet filtering rule assigned to that bit will not be in effect, and if any incoming packet violates that rule (but no other rules), then a "Filter Fail" will not occur, the packet will not be rejected, the packet will be treated as "intended for the device", and software will be notified of the incoming packet. For a desciption of the function of the individual leniency bits, see the 802.15.4 Link Layer Controller chapter. |

## 64.5.2.5.1.40 PART ID (PART_ID)

### *64.5.2.5.1.40.1 Offset*

| Register | Offset |
|---|---|
| PART_ID | A0h |

### *64.5.2.5.1.40.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PART_ID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### *64.5.2.5.1.40.3 Fields*

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>PART_ID | 802.15.4 Part ID |

## 64.5.2.5.1.41 Packet Buffer TX (PKT_BUFFER_TX0 - PKT_BUFFER_TX63)

### *64.5.2.5.1.41.1 Offset*

For a = 0 to 63:

| Register | Offset |
|---|---|
| PKT_BUFFER_TXa | 100h + (a × 2h) |

### *64.5.2.5.1.41.2 Function*

Packet Buffer TX

### *64.5.2.5.1.41.3 Diagram*

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKT_BUFFER_TX | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### *64.5.2.5.1.41.4 Fields*

| Field | Function |
|---|---|
| 15-0<br>PKT_BUFFER_TX | Packet Buffer Entry<br>Packet Buffer Word |

## 64.5.2.5.1.42 Packet Buffer RX (PKT_BUFFER_RX0 - PKT_BUFFER_RX63)

### *64.5.2.5.1.42.1 Offset*

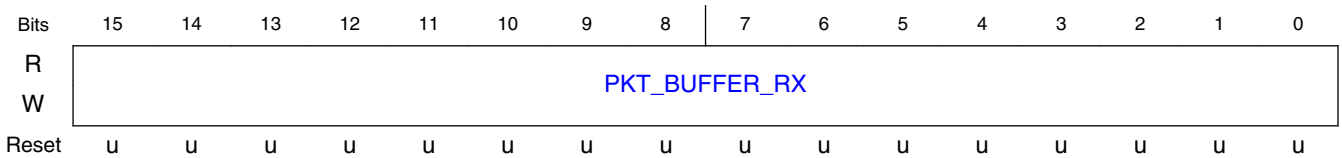
For a = 0 to 63:

| Register | Offset |
|---|---|
| PKT_BUFFER_RXa | 180h + (a × 2h) |

#### 64.5.2.5.1.42.2 Function

Packet Buffer RX

#### 64.5.2.5.1.42.3 Diagram




#### 64.5.2.5.1.42.4 Fields

| Field | Function |
|---|---|
| 15-0<br>PKT_BUFFER_RX | Packet Buffer Entry<br>Packet Buffer Word |

## 64.5.3 Generic_FSK Link Layer Controller

### 64.5.3.1 Introduction

The Generic FSK protocol enables radio operation using a custom GFSK/GMSK or MSK modulation format achieved by programming a set of PHY variables such as BT product, modulation index and modulation filter co-efficients (such that max frequency deviation <= 500kHz). Generic FSK mode also offers a highly configurable packet structure, variable bit rate transmission and reception, some limited packet (header) processing, and interface to a RAM-based Packet Buffer.

#### 64.5.3.1.1 Overview

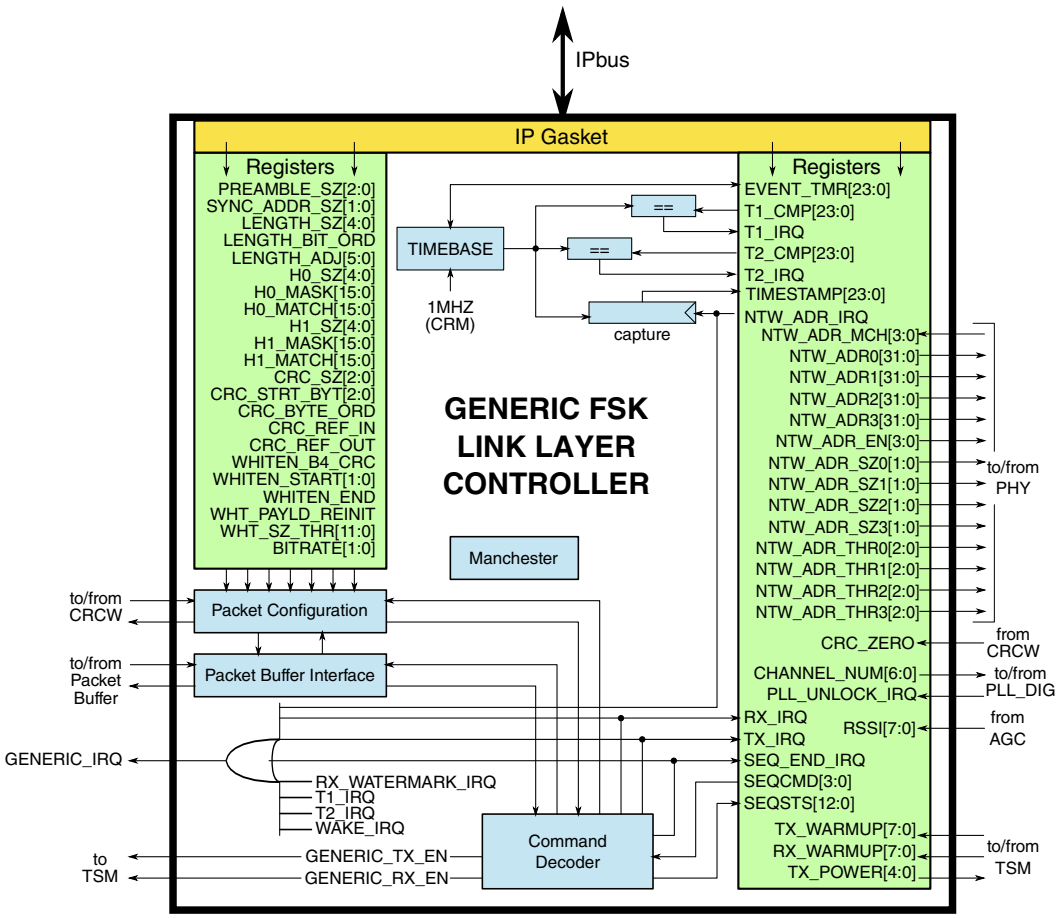
The Generic FSK Link Layer Controller provides the interface between Generic FSK software, and the 2.4GHz transceiver hardware. The interface consists of an IP bus-based set of registers, which provide command, control, and status to Generic FSK software. The Link Layer controller features programmability that allows for a highly configurable packet structure, defining the lengths, bit ordering, and contents of individual packet fields, defining the start and end points for whitening, CRC, and Manchester encoding/decoding, and providing for some primitive parsing of the packet header. The controller

provides multiple options for bitrate, for both TX and RX. The Link Layer Controller implements a Sequence Command Set consisting of 12 commands; a command decoder to interpret the commands in realtime as they are received from Generic FSK software, and to provide the necessary control signals to the radio's Transceiver Sequence Manager (TSM); a high-precision timebase based on an external crystal-based oscillator; an Interrupt Control Unit; finite state machines to control the transmission and reception of Generic FSK packets; controls for CRC generation (TX) and verification (RX); controls for data whitening; Manchester encoding; interface to RAM-based Packet Buffer for packet storage; and provisions for entering and exiting low-power Deep Sleep Mode.

#### 64.5.3.1.2 Features

- Highly configurable packet structure
- Optimized 12-entry Sequence Command Set
- High-precision timebase to maintain network timing
- Two timer-compare mechanisms for Interrupt Generation and Sequence Launching
- Hardware automation for packet transmit and receive, CRC, Whitening, and Manchester encoding
- Up to 4 Network Addresses to synchronize to, each can be 8-bit, 16-bit or 32-bit
- Packet Lengths up to 2047 Bytes
- Deep Sleep Mode support
- Highly configurable packet structure
- Optimized 12-entry Sequence Command Set
- High-precision timebase to maintain network timing
- Two timer-compare mechanisms for Interrupt Generation and Sequence Launching
- Hardware automation for packet transmit and receive, CRC, Whitening, and Manchester encoding
- Up to 4 Network Addresses to synchronize to, each can be 8-bit, 16-bit or 32-bit
- Packet Lengths up to 2047 Bytes
- Deep Sleep Mode support

#### 64.5.3.1.3 Block Diagram




## 64.5.3.2 Memory Map and Register Definition

### 64.5.3.2.1 GENERIC FSK register descriptions

#### 64.5.3.2.1.1 FSK Memory map

GENFSK base address: 4103_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | IRQ CONTROL (IRQ_CTRL) | 32 | RW | See description. |

*Table continues on the next page...*

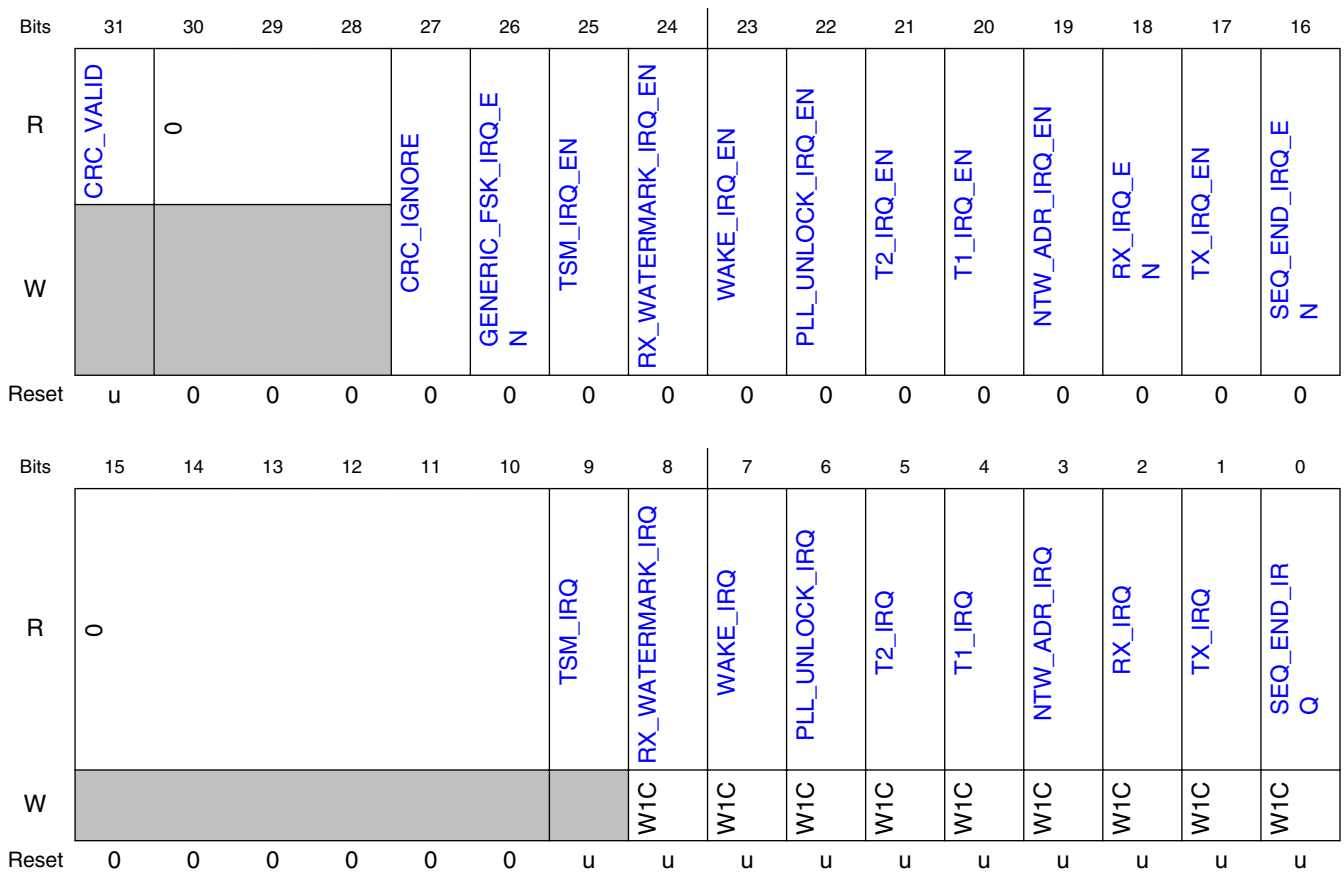
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | EVENT TIMER (EVENT_TMR) | 32 | RW | See description. |
| 8h | T1 COMPARE (T1_CMP) | 32 | RW | 00FF_FFFFh |
| Ch | T2 COMPARE (T2_CMP) | 32 | RW | 00FF_FFFFh |
| 10h | TIMESTAMP (TIMESTAMP) | 32 | RO | See description. |
| 14h | TRANSCEIVER CONTROL (XCVR_CTRL) | 32 | RW | See description. |
| 18h | TRANSCEIVER STATUS (XCVR_STS) | 32 | RO | See description. |
| 1Ch | TRANSCEIVER CONFIGURATION (XCVR_CFG) | 32 | RW | See description. |
| 20h | CHANNEL NUMBER (CHANNEL_NUM) | 32 | RW | 0000_0000h |
| 24h | TRANSMIT POWER (TX_POWER) | 32 | RW | 0000_0000h |
| 28h | NETWORK ADDRESS CONTROL (NTW_ADR_CTRL) | 32 | RW | See description. |
| 2Ch | NETWORK ADDRESS 0 (NTW_ADR_0) | 32 | RW | 5555_5555h |
| 30h | NETWORK ADDRESS 1 (NTW_ADR_1) | 32 | RW | 5555_5555h |
| 34h | NETWORK ADDRESS 2 (NTW_ADR_2) | 32 | RW | 5555_5555h |
| 38h | NETWORK ADDRESS 3 (NTW_ADR_3) | 32 | RW | 5555_5555h |
| 3Ch | RECEIVE WATERMARK (RX_WATERMARK) | 32 | RW | See description. |
| 40h | DSM CONTROL (DSM_CTRL) | 32 | RW | 0000_0000h |
| 44h | PART ID (PART_ID) | 32 | RO | 0000_0001h |
| 60h | PACKET CONFIGURATION (PACKET_CFG) | 32 | RW | See description. |
| 64h | H0 CONFIGURATION (H0_CFG) | 32 | RW | 0000_0000h |
| 68h | H1 CONFIGURATION (H1_CFG) | 32 | RW | 0000_0000h |
| 6Ch | CRC CONFIGURATION (CRC_CFG) | 32 | RW | 0000_0002h |
| 70h | CRC INITIALIZATION (CRC_INIT) | 32 | RW | 0000_0000h |
| 74h | CRC POLYNOMIAL (CRC_POLY) | 32 | RW | 1021_0000h |
| 78h | CRC XOR OUT (CRC_XOR_OUT) | 32 | RW | 0000_0000h |
| 7Ch | WHITENER CONFIGURATION (WHITEN_CFG) | 32 | RW | 01FF_0918h |
| 80h | WHITENER POLYNOMIAL (WHITEN_POLY) | 32 | RW | 0000_0021h |
| 84h | WHITENER SIZE THRESHOLD (WHITEN_SZ_THR) | 32 | RW | 0000_0800h |
| 88h | BIT RATE (BITRATE) | 32 | RW | 0000_0000h |
| 8Ch | PACKET BUFFER PARTITION POINT (PB_PARTITION) | 32 | RW | 0000_0220h |
| 700h - F7Eh | PACKET BUFFER (PACKET_BUFFER_0 - PACKET_BUFFER_1087) | 16 | RW | See description. |

# 64.5.3.2.1.2 IRQ CONTROL (IRQ_CTRL)

## *64.5.3.2.1.2.1 Offset*

| Register | Offset |
|---|---|
| IRQ_CTRL | 0h |

## *64.5.3.2.1.2.2 Diagram*




## *64.5.3.2.1.2.3 Fields*

| Field | Function |
|---|---|
| 31<br>CRC_VALID | CRC Valid<br>CRC Valid indicator for RX packets. This bit becomes valid at RX_IRQ, and remains valid until the start of the next RX TSM sequence.<br>0b - CRC of RX packet is not valid.<br>1b - CRC of RX packet is valid. |
| 30-28 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27<br>CRC_IGNORE | CRC Ignore<br>If set, assert RX_IRQ even for a received packet which fails CRC verification.<br>0b - RX_IRQ will not be asserted for a received packet which fails CRC verification.<br>1b - RX_IRQ will be asserted even for a received packet which fails CRC verification. |
| 26<br>GENERIC_FSK_IRQ_EN | GENERIC_FSK_IRQ Master Enable<br>Master enable for the GENERIC_FSK_IRQ interrupt line to the MCU.<br>0b - All GENERIC_FSK Interrupts are disabled.<br>1b - All GENERIC_FSK Interrupts can be enabled. |
| 25<br>TSM_IRQ_EN | TSM_IRQ Enable<br>0b - TSM Interrupt is not enabled.<br>1b - TSM Interrupt is enabled. |
| 24<br>RX_WATERMARK_IRQ_EN | RX_WATERMARK_IRQ Enable<br>0b - RX Watermark Interrupt is not enabled.<br>1b - RX Watermark Interrupt is enabled. |
| 23<br>WAKE_IRQ_EN | WAKE_IRQ Enable<br>0b - Wake Interrupt is not enabled.<br>1b - Wake Interrupt is enabled. |
| 22<br>PLL_UNLOCK_IRQ_EN | PLL_UNLOCK_IRQ Enable<br>0b - PLL Unlock Interrupt is not enabled.<br>1b - PLL Unlock Interrupt is enabled. |
| 21<br>T2_IRQ_EN | T2_IRQ Enable<br>0b - Timer1 (T2) Compare Interrupt is not enabled.<br>1b - Timer1 (T2) Compare Interrupt is enabled. |
| 20<br>T1_IRQ_EN | T1_IRQ Enable<br>0b - Timer1 (T1) Compare Interrupt is not enabled.<br>1b - Timer1 (T1) Compare Interrupt is enabled. |
| 19<br>NTW_ADR_IRQ_EN | NTW_ADR_IRQ Enable<br>0b - Network Address Match Interrupt is not enabled.<br>1b - Network Address Match Interrupt is enabled. |
| 18<br>RX_IRQ_EN | RX_IRQ Enable<br>0b - RX Interrupt is not enabled.<br>1b - RX Interrupt is enabled. |
| 17<br>TX_IRQ_EN | TX_IRQ Enable<br>0b - TX Interrupt is not enabled.<br>1b - TX Interrupt is enabled. |
| 16<br>SEQ_END_IRQ_EN | SEQ_END_IRQ Enable<br>0b - Sequence End Interrupt is not enabled.<br>1b - Sequence End Interrupt is enabled. |
| 15-10<br>— | Reserved. |
| 9<br>TSM_IRQ | TSM Interrupt<br>Indicates TSM0_IRQ or TSM1_IRQ is set in XCVR_STATUS. Clear the bits there. For debug purposes.<br>0b - TSM0_IRQ and TSM1_IRQ are both clear.<br>1b - Indicates TSM0_IRQ or TSM1_IRQ is set in XCVR_STATUS. |
| 8 | RX Watermark Interrupt<br>Asserts when RX Byte Counter == RX_WATERMARK[12:0] |

*Table continues on the next page...*

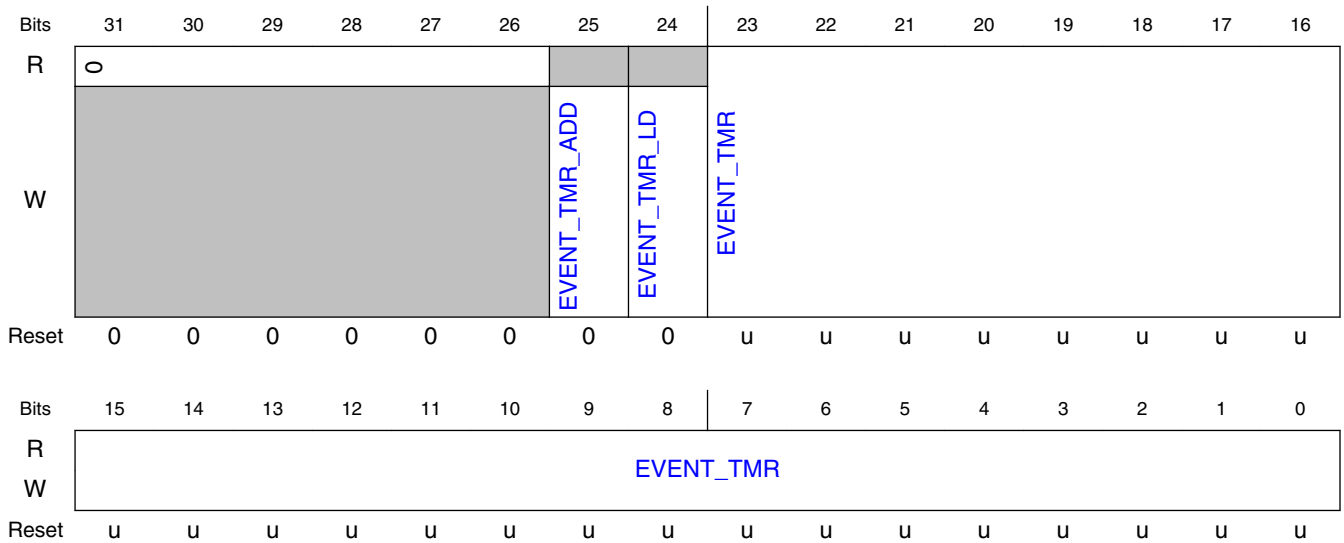
| Field | Function |
|---|---|
| RX_WATERMARK_IRQ | 0b - RX Watermark Interrupt is not asserted.<br>1b - RX Watermark Interrupt is asserted. |
| 7<br>WAKE_IRQ | Wake Interrrupt<br>**For Manual DSM:** This is WAKE_IRQ. A WAKE_IRQ will be triggered when the GENERIC_FSK Link Layer Controller has awoken from a Manual DSM (Deep Sleep Mode) cycle. WAKE_IRQ indicates that the RF Oscillator has been restarted, and the GENERIC_FSK EVENT_TMR has resumed counting.<br>**For Wake-On-Radio:** This is WOR_IRQ. See Section Wake-On-Radio<br>0b - Wake Interrupt is not asserted.<br>1b - Wake Interrupt is asserted. |
| 6<br>PLL_UNLOCK_IRQ | PLL Unlock Interrupt<br>An unlock event has occurred.<br>0b - PLL Unlock Interrupt is not asserted.<br>1b - PLL Unlock Interrupt is asserted. |
| 5<br>T2_IRQ | Timer2 (T2) Compare Interrupt<br>0b - Timer2 (T2) Compare Interrupt is not asserted.<br>1b - Timer2 (T2) Compare Interrupt is asserted. |
| 4<br>T1_IRQ | Timer1 (T1) Compare Interrupt<br>0b - Timer1 (T1) Compare Interrupt is not asserted.<br>1b - Timer1 (T1) Compare Interrupt is asserted. |
| 3<br>NTW_ADR_IRQ | Network Address Match Interrupt<br>A Network Address Match has occurred.<br>0b - Network Address Match Interrupt is not asserted.<br>1b - Network Address Match Interrupt is asserted. |
| 2<br>RX_IRQ | RX Interrupt<br>The RX sequence has completed with a successful packet reception.<br>0b - RX Interrupt is not asserted.<br>1b - RX Interrupt is asserted. |
| 1<br>TX_IRQ | TX Interrupt<br>The TX sequence has completed with a successful packet transmission.<br>0b - TX Interrupt is not asserted.<br>1b - TX Interrupt is asserted. |
| 0<br>SEQ_END_IRQ | Sequence End Interrupt<br>Will assert when any TX or RX sequence ends for any reason.<br>0b - Sequence End Interrupt is not asserted.<br>1b - Sequence End Interrupt is asserted. |

## 64.5.3.2.1.3 EVENT TIMER (EVENT_TMR)

### *64.5.3.2.1.3.1 Offset*

| Register | Offset |
|---|---|
| EVENT_TMR | 4h |

#### 64.5.3.2.1.3.2 Diagram




#### 64.5.3.2.1.3.3 Fields

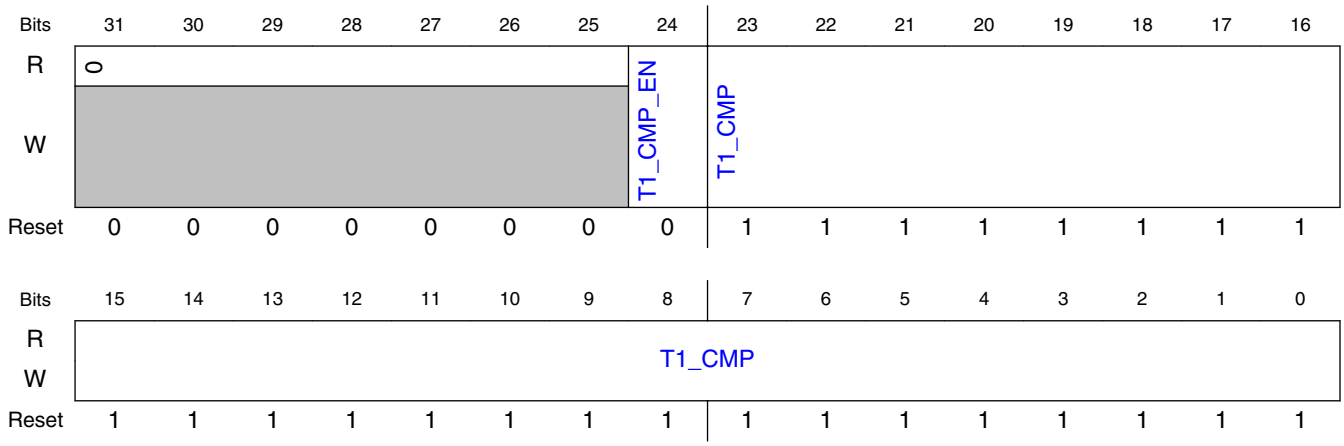
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>EVENT_TMR_ADD | Event Timer Add<br>A write access with this bit increments EVENT_TMR by the contents of EVENT_TMR[23:0]. This is a signed addition. |
| 24<br>EVENT_TMR_LD | Event Timer Load<br>A write access with this bit set loads EVENT_TMR with the contents of EVENT_TMR[23:0] |
| 23-0<br>EVENT_TMR | Event Timer<br>Event Timer can be read in these byte locations. To update the Event Timer, either: 1. Write the desired EVENT_TMR to these bytes and set EVENT_TMR_LD=1, or, 2. Write the desired EVENT_TMR increment amount to these bytes and set EVENT_TMR_ADD=1.<br><br>**Note:** for EVENT_TMR_ADD, EVENT_TMR[23:0] is a signed, two's-complement value. |

## 64.5.3.2.1.4 T1 COMPARE (T1_CMP)

#### 64.5.3.2.1.4.1 Offset

| Register | Offset |
|---|---|
| T1_CMP | 8h |

#### 64.5.3.2.1.4.2 Diagram




#### 64.5.3.2.1.4.3 Fields

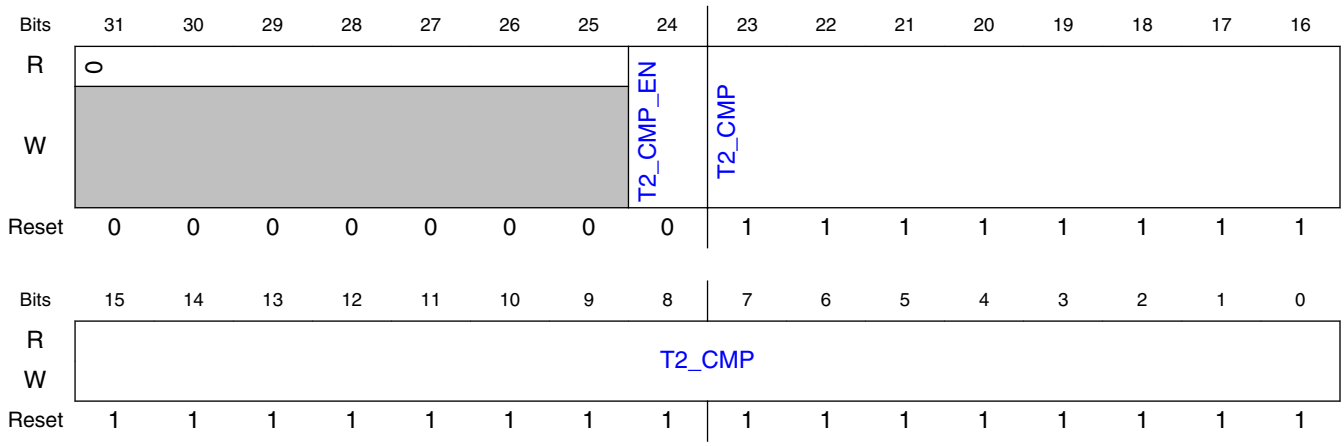
| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>T1_CMP_EN | Timer1 (T1) Compare Enable<br>Enable Timer Compare #1 (T1_CMP) to generate T1_IRQ and/or execute Sequence Commands. |
| 23-0<br>T1_CMP | Timer1 (T1) Compare Value<br>Timer1 (T1) Compare Value. Can be used to generate T1_IRQ and/or launch Sequence Commands |

### 64.5.3.2.1.5 T2 COMPARE (T2_CMP)

#### 64.5.3.2.1.5.1 Offset

| Register | Offset |
|---|---|
| T2_CMP | Ch |

#### 64.5.3.2.1.5.2 Diagram




#### 64.5.3.2.1.5.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>T2_CMP_EN | Timer2 (T2) Compare Enable<br>Enable Timer Compare #2 (T2_CMP) to generate T2_IRQ and/or execute Sequence Commands. |
| 23-0<br>T2_CMP | Timer2 (T2) Compare Value<br>Timer2 (T2) Compare Value. Can be used to generate T2_IRQ and/or launch Sequence Commands |

### 64.5.3.2.1.6 TIMESTAMP (TIMESTAMP)

#### 64.5.3.2.1.6.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP | 10h |

#### 64.5.3.2.1.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TIMESTAMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### 64.5.3.2.1.6.3 Fields

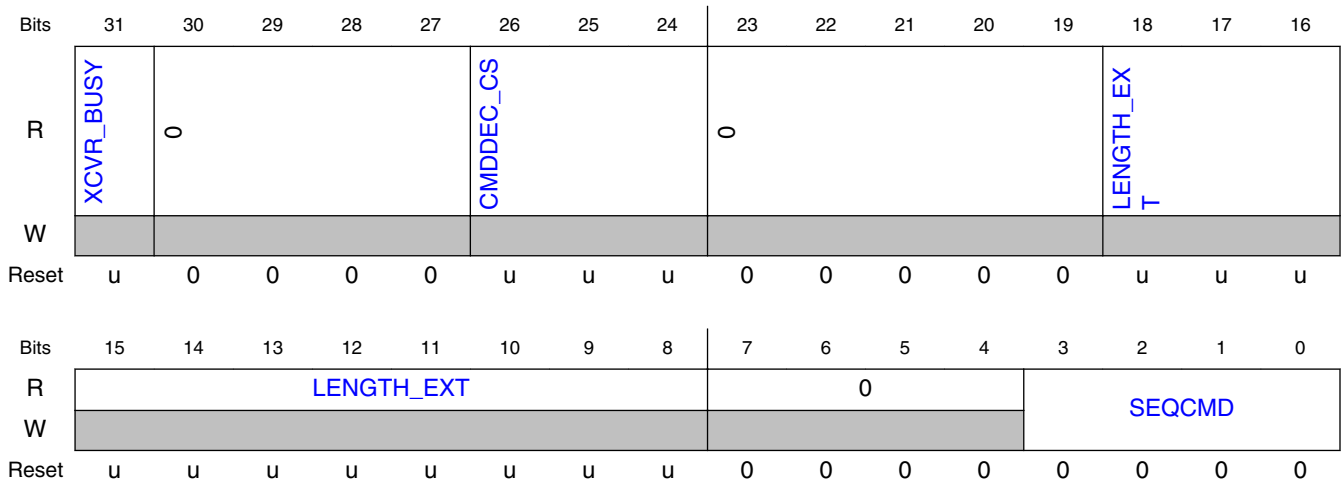
| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>TIMESTAMP | Received Packet Timestamp<br>Received Packet Timestamp. Captured at NTW_ADR_IRQ. |

### 64.5.3.2.1.7 TRANSCEIVER CONTROL (XCVR_CTRL)

#### 64.5.3.2.1.7.1 Offset

| Register | Offset |
|---|---|
| XCVR_CTRL | 14h |

### 64.5.3.2.1.7.2 Diagram




### 64.5.3.2.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>XCVR_BUSY | Transceiver Busy<br>For multi-protocol arbitration, XCVR_BUSY=1 indicates an RX or TX operation is underway, by either GENERIC_FSK, or some other protocol.<br>0b - IDLE<br>1b - BUSY |
| 30-27<br>— | Reserved. |
| 26-24<br>CMDDEC_CS | Command Decode<br>Current State of the Command Decoder FSM (debug only) |
| 23-19<br>— | Reserved. |
| 18-8<br>LENGTH_EXT | Extracted Length Field<br>The read-only register reflects the contents of the LENGTH field of the Header of the most-recently received packet. This is the raw, unmodified LENGTH field, as it appears in the received bit stream, unmodified by LENGTH_ADJ or any other parameter. |
| 7-4<br>— | Reserved. |
| 3-0<br>SEQCMD | Sequence Commands<br>0000b - No Action<br>0001b - TX Start Now<br>0010b - TX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP)<br>0011b - TX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>0100b - TX Cancel -- Cancels pending TX events but do not abort a TX-in-progress<br>0101b - RX Start Now<br>0110b - RX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP)<br>0111b - RX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>1000b - RX Stop @ T1 Timer Compare Match (EVENT_TMR = T1_CMP) |

| Field | Function |
|---|---|
| | 1001b - RX Stop @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>1010b - RX Cancel -- Cancels pending RX events but do not abort a RX-in-progress<br>1011b - Abort All - Cancels all pending events and abort any sequence-in-progress<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

## 64.5.3.2.1.8 TRANSCEIVER STATUS (XCVR_STS)

### *64.5.3.2.1.8.1 Offset*

| Register | Offset |
|---|---|
| XCVR_STS | 18h |

### *64.5.3.2.1.8.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LQI | | | | | | | | RSSI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VALID | LQI_VALID | 0 | RX_IN_WARMDN | RX_IN_PROGRESS | RX_IN_SEARCH | RX_IN_WARMUP | RX_STOP_T2_PEND | RX_STOP_T1_PEND | RX_START_T2_PEND | RX_START_T1_PEND | TX_IN_WARMDN | TX_IN_PROGRESS | TX_IN_WARMUP | TX_START_T2_PEND | TX_START_T1_PEND |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u |

### *64.5.3.2.1.8.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>LQI | Link Quality Indicator<br>This field is valid when LQI_VALID=1. LQI is a unsigned, unitless value. |
| 23-16 | Received Signal Stength Indicator, in dBm |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RSSI | |
| 15<br>CRC_VALID | CRC Valid Indicator<br>CRC Valid indicator for RX packets.<br>0b - CRC is not valid for RX packet.<br>1b - CRC is valid for RX packet. |
| 14<br>LQI_VALID | LQI Valid Indicator<br>LQI Valid indicator for RX packets. This bit becomes set when the LQI computation completes for the packet currently being received, and remains set for the remainder of the packet.<br>0b - LQI is not yet valid for RX packet.<br>1b - LQI is valid for RX packet. |
| 13<br>— | Reserved. |
| 12<br>RX_IN_WARMDN | RX Warmdown Status<br>RX Sequence in TSM Warmdown |
| 11<br>RX_IN_PROGRESS | RX in Progress Status<br>RX Packet Reception Currently Underway |
| 10<br>RX_IN_SEARCH | RX Search Status<br>RX Sequence in Network Address Search |
| 9<br>RX_IN_WARMUP | RX Warmup Status<br>RX Sequence in TSM Warmup |
| 8<br>RX_STOP_T2_PEND | RX T2 Start Pending Status<br>RX Sequence will stop @ next T2 Match |
| 7<br>RX_STOP_T1_PEND | RX T1 Stop Pending Status<br>RX Sequence will stop @ next T1 Match |
| 6<br>RX_START_T2_PEND | RX T2 Start Pending Status<br>RX Sequence will start @ next T2 Match |
| 5<br>RX_START_T1_PEND | RX T1 Start Pending Status<br>RX Sequence will start @ next T1 Match |
| 4<br>TX_IN_WARMDN | TX Warmdown Status<br>TX Sequence in TSM Warmdown |
| 3<br>TX_IN_PROGRESS | TX in Progress Status<br>TX Packet Transmission Currently Underway |
| 2<br>TX_IN_WARMUP | TX Warmup Status<br>TX Sequence in TSM Warmup |

*Table continues on the next page...*

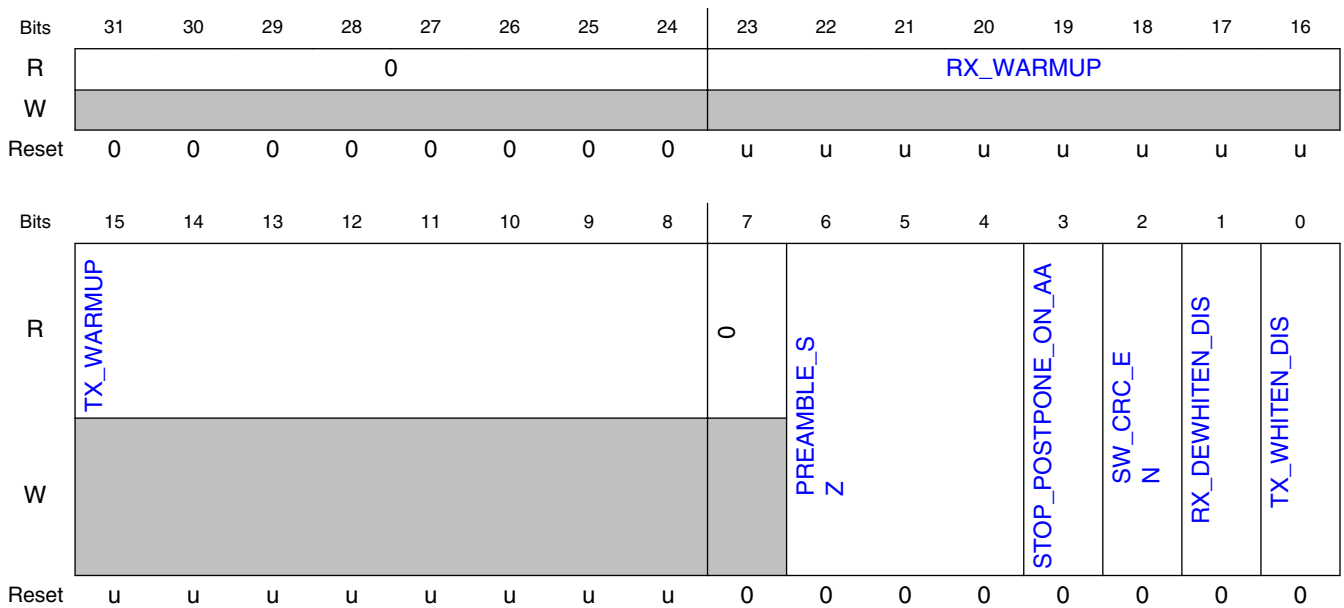
| Field | Function |
|---|---|
| 1<br>TX_START_T2_PEND | TX T2 Start Pending Status<br>TX Sequence will start @ next T2 Match |
| 0<br>TX_START_T1_PEND | TX T1 Start Pending Status<br>TX Sequence will start @ next T1 Match |

## 64.5.3.2.1.9 TRANSCEIVER CONFIGURATION (XCVR_CFG)

### *64.5.3.2.1.9.1 Offset*

| Register | Offset |
|---|---|
| XCVR_CFG | 1Ch |

### *64.5.3.2.1.9.2 Diagram*




### *64.5.3.2.1.9.3 Fields*

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>RX_WARMUP | Receive Warmup Time<br>RX warmup time, in microseconds, provided to Link Layer software, so that warmup time can be accounted for when scheduling over-the-air transactions. |
| 15-8<br>TX_WARMUP | Transmit Warmup Time<br>TX warmup time, in microseconds, provided to Link Layer software, so that warmup time can be accounted for when scheduling over-the-air transactions. |
| 7<br>— | Reserved. |
| 6-4<br>PREAMBLE_SZ | Preamble Size<br>Number of Octets = PREAMBLE_SZ + 1, where 0 <= PREAMBLE_SZ <= 7 |
| 3<br>STOP_POSTPONE_ON_AA | Postpone Stop Command Timeout On Access Address Match Enable<br>If this bit is set, if a RX STOP condition occurs, due to an Event Timer match to a previously issued RX_STOP_T1 or RX_STOP_T2 sequence command, while a packet is being received (*ntw_adr_matched* = 1), the timeout-related abort will be deferred untl the packet reception is completed. If the packet is good, HW will signal Data Indication; otherwise the Sequence Manager will return to IDLE and await further commanding.<br>0b - STOP Abort will occur on RX_STOP_T1 or RX_STOP_T1 Event Timer match, regardless of ntw_adr_matched<br>1b - STOP Abort will be deferred on RX_STOP_T1 or RX_STOP_T1 Event Timer match, if ntw_adr_matched is asserted; otherwise the RX_STOP Abort will occur immediately |
| 2<br>SW_CRC_EN | Software CRC Enable<br>Software override of the HW-computed CRC for TX. Software must write CRC to Packet Buffer (RAM) |
| 1<br>RX_DEWHITEN_DIS | RX De-Whitening Disable<br>Disable all de-whitening on RX packets |
| 0<br>TX_WHITEN_DIS | TX Whitening Disable<br>Disable all whitening on TX packets |

## 64.5.3.2.1.10 CHANNEL NUMBER (CHANNEL_NUM)

### *64.5.3.2.1.10.1 Offset*

| Register | Offset |
|---|---|
| CHANNEL_NUM | 20h |

#### *64.5.3.2.1.10.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHANNEL_NUM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.5.3.2.1.10.3 Fields*

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>CHANNEL_NUM | Channel Number<br>RF Channel Select: 0 <= CHANNEL_NUM <= 127; Formula: F = (2360 + CHANNEL_NUM) [in MHz] |

### 64.5.3.2.1.11 TRANSMIT POWER (TX_POWER)

#### *64.5.3.2.1.11.1 Offset*

| Register | Offset |
|---|---|
| TX_POWER | 24h |

#### *64.5.3.2.1.11.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | TX_POWER | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### 64.5.3.2.1.11.3 Fields

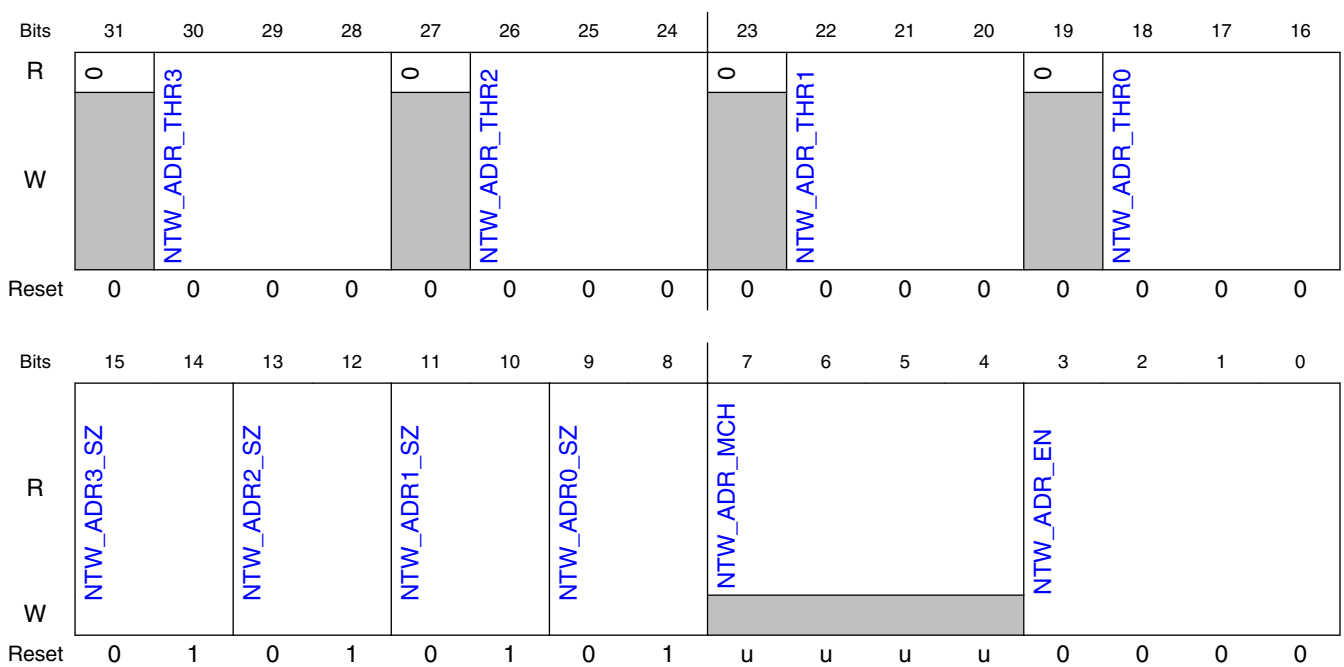
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>TX_POWER | Transmit Power<br>PA Power Level. |

## 64.5.3.2.1.12 NETWORK ADDRESS CONTROL (NTW_ADR_CTRL)

#### 64.5.3.2.1.12.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_CTRL | 28h |

#### 64.5.3.2.1.12.2 Diagram



### *64.5.3.2.1.12.3 Fields*

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-28<br>NTW_ADR_THR3 | Network Address 3 Threshold<br>Number of Tolerated bit errors for Network Address 3 |
| 27<br>— | Reserved. |
| 26-24<br>NTW_ADR_THR2 | Network Address 2 Threshold<br>Number of Tolerated bit errors for Network Address 2 |
| 23<br>— | Reserved. |
| 22-20<br>NTW_ADR_THR1 | Network Address 1 Threshold<br>Number of Tolerated bit errors for Network Address 1 |
| 19<br>— | Reserved. |
| 18-16<br>NTW_ADR_THR0 | Network Address 0 Threshold<br>Number of Tolerated bit errors for Network Address 0 |
| 15-14<br>NTW_ADR3_SZ | Network Address 3 Size<br>00b - Network Address 3 requires a 8-bit correlation<br>01b - Network Address 3 requires a 16-bit correlation<br>10b - Network Address 3 requires a 24-bit correlation<br>11b - Network Address 3 requires a 32-bit correlation |
| 13-12<br>NTW_ADR2_SZ | Network Address 2 Size<br>00b - Network Address 2 requires a 8-bit correlation<br>01b - Network Address 2 requires a 16-bit correlation<br>10b - Network Address 2 requires a 24-bit correlation<br>11b - Network Address 2 requires a 32-bit correlation |
| 11-10<br>NTW_ADR1_SZ | Network Address 1 Size<br>00b - Network Address 1 requires a 8-bit correlation<br>01b - Network Address 1 requires a 16-bit correlation<br>10b - Network Address 1 requires a 24-bit correlation<br>11b - Network Address 1 requires a 32-bit correlation |
| 9-8<br>NTW_ADR0_SZ | Network Address 0 Size<br>00b - Network Address 0 requires a 8-bit correlation<br>01b - Network Address 0 requires a 16-bit correlation<br>10b - Network Address 0 requires a 24-bit correlation<br>11b - Network Address 0 requires a 32-bit correlation |
| 7-4<br>NTW_ADR_MCH | Network Address Match<br>Indicates which of the 4 Network Addresses has matched in the PHY. Valid during an RX sequence at the point of match, and remains asserted until either:<br><br>1. The next RX sequence begins (if the current packet passed CRC and header filtering), or, |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 2. An RX recycle to Network Address search (if the current packet failed CRC or header filtering)<br>0001b - Network Address 0 has matched<br>0010b - Network Address 1 has matched<br>0100b - Network Address 2 has matched<br>1000b - Network Address 3 has matched |
| 3-0<br>NTW_ADR_EN | Network Address Enable<br>Enable Network Address N for PHY correlation, where 0 ≤ N ≤ 3. Any bit combination can be set.<br>0001b - Enable Network Address 0 for correlation<br>0010b - Enable Network Address 1 for correlation<br>0100b - Enable Network Address 2 for correlation<br>1000b - Enable Network Address 3 for correlation |

## 64.5.3.2.1.13 NETWORK ADDRESS 0 (NTW_ADR_0)

### *64.5.3.2.1.13.1 Offset*

| Register | Offset |
|---|---|
| NTW_ADR_0 | 2Ch |

### *64.5.3.2.1.13.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_0 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_0 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

### *64.5.3.2.1.13.3 Fields*

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_0 | Network Address 0<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[0]] = 1 |

## 64.5.3.2.1.14 NETWORK ADDRESS 1 (NTW_ADR_1)

### *64.5.3.2.1.14.1 Offset*

| Register | Offset |
|---|---|
| NTW_ADR_1 | 30h |

### *64.5.3.2.1.14.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_1 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_1 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

### *64.5.3.2.1.14.3 Fields*

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_1 | Network Address 1<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[1]] = 1 |

## 64.5.3.2.1.15 NETWORK ADDRESS 2 (NTW_ADR_2)

### *64.5.3.2.1.15.1 Offset*

| Register | Offset |
|---|---|
| NTW_ADR_2 | 34h |

#### 64.5.3.2.1.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_2 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_2 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

#### 64.5.3.2.1.15.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_2 | Network Address 2<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[2]] = 1 |

### 64.5.3.2.1.16 NETWORK ADDRESS 3 (NTW_ADR_3)

#### 64.5.3.2.1.16.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_3 | 38h |

#### 64.5.3.2.1.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_3 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_3 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

#### 64.5.3.2.1.16.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_3 | Network Address 2<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[3]] = 1 |

## 64.5.3.2.1.17 RECEIVE WATERMARK (RX_WATERMARK)

#### 64.5.3.2.1.17.1 Offset

| Register | Offset |
|---|---|
| RX_WATERMARK | 3Ch |

#### 64.5.3.2.1.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | BYTE_COUNTER | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | RX_WATERMARK | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### 64.5.3.2.1.17.3 Fields

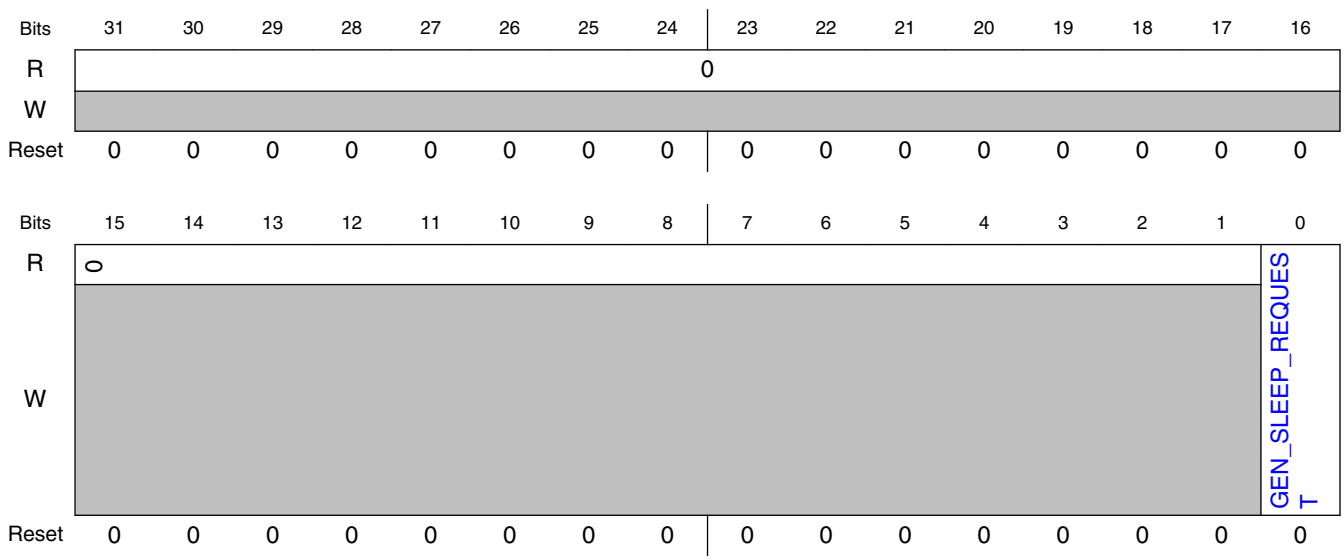
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>BYTE_COUNTER | Byte Counter<br>Reflects the current Byte Count, for TX and RX. This is a signed, twos-complement value. For values less than zero, indicates the preamble transmission is underway (TX only). A value of 0 indicates the first octet of Network Address is being transmitted or received. A value of 1 indicates the second octet of Network Address is being transmitted or received. Etc. |
| 15-13<br>— | Reserved. |
| 12-0<br>RX_WATERMARK | Receive Watermark<br>Sets the trigger for RX_WATERMARK_IRQ. Trigger the RX_WATERMARK_IRQ when: RX Byte Counter == RX_WATERMARK[12:0] |

## 64.5.3.2.1.18 DSM CONTROL (DSM_CTRL)

### *64.5.3.2.1.18.1 Offset*

| Register | Offset |
|---|---|
| DSM_CTRL | 40h |

### *64.5.3.2.1.18.2 Diagram*




### *64.5.3.2.1.18.3 Fields*

| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>GEN_SLEEP_REQUEST | GENERIC_FSK Deep Sleep Mode Request<br>If GENERIC_FSK is assigned to Manual DSM (XCVR_CTRL[MAN_DSM_SEL] = 2), setting GEN_SLEEP_REQUEST to 1 enables Deep Sleep Mode (DSM) to be entered when the RSIM DSM_TIMER matches the RSIM MAN_SLEEP register. |

## 64.5.3.2.1.19 PART ID (PART_ID)

#### *64.5.3.2.1.19.1 Offset*

| Register | Offset |
|---|---|
| PART_ID | 44h |

#### *64.5.3.2.1.19.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PART_ID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### *64.5.3.2.1.19.3 Fields*

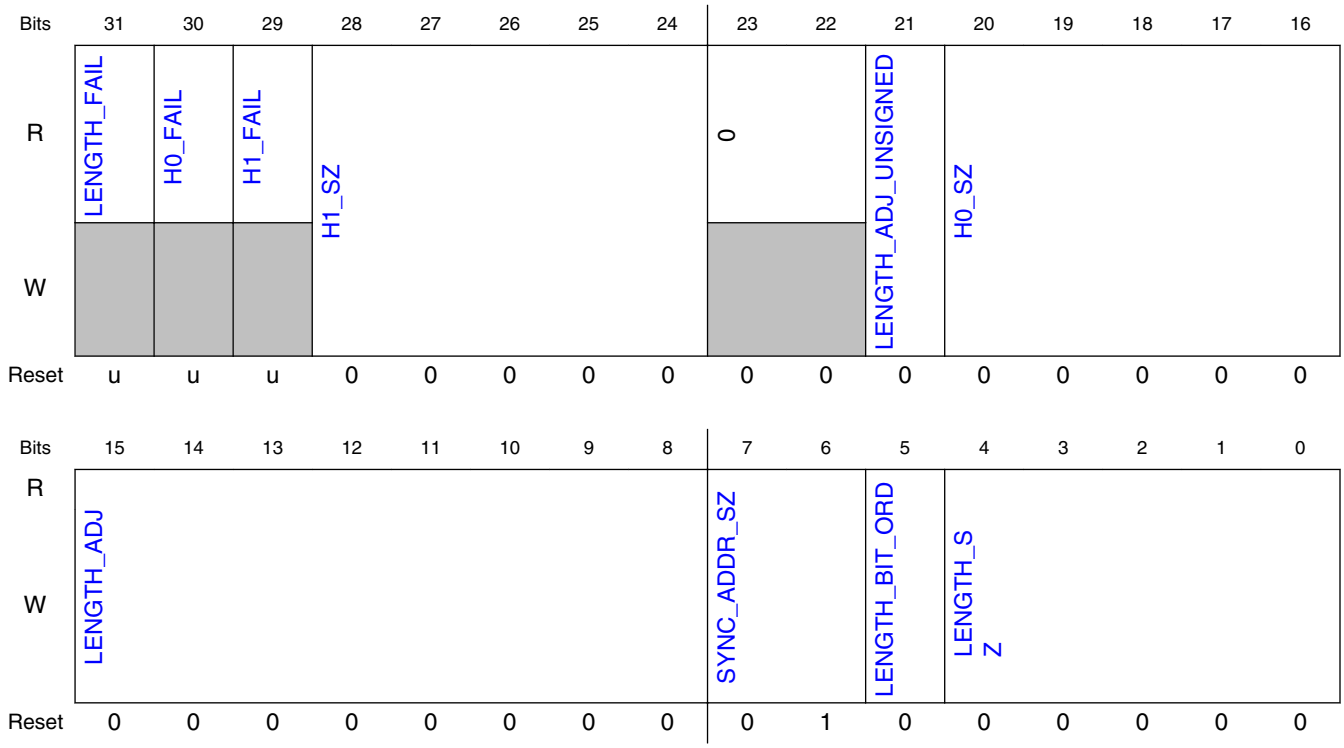
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>PART_ID | Part ID<br>Part ID to identify HW revision of the Generic FSK Link Layer |

### 64.5.3.2.1.20 PACKET CONFIGURATION (PACKET_CFG)

#### *64.5.3.2.1.20.1 Offset*

| Register | Offset |
|---|---|
| PACKET_CFG | 60h |

### 64.5.3.2.1.20.2 Diagram




### 64.5.3.2.1.20.3 Fields

| Field | Function |
|---|---|
| 31<br>LENGTH_FAIL | Maximum Length Violated Status Bit<br>For packets received with REC_BAD_PKT=1, LENGTH_FAIL indicates the extracted LENGTH header field exceeded LENGTH_MAX |
| 30<br>H0_FAIL | H0 Violated Status Bit<br>For packets received with REC_BAD_PKT=1, H0_FAIL indicates the received H0 header field violates the H0_MASK/H0_MATCH pattern |
| 29<br>H1_FAIL | H1 Violated Status Bit<br>For packets received with REC_BAD_PKT=1, H1_FAIL indicates the received H1 header field violates the H1_MASK/H1_MATCH pattern |
| 28-24<br>H1_SZ | H1 Size<br>Length of H1 in bits; 0 ≤ H1_SZ ≤ 16 |
| 23-22<br>— | Reserved. |
| 21<br>LENGTH_ADJ_UNSIGNED | Length Adjustment Unsigned Enabled<br>0b - Hardware interprets LENGTH_ADJ as a signed integer (default)<br>1b - Hardware interprets LENGTH_ADJ as a unsigned integer |
| 20-16<br>H0_SZ | H0 Size<br>Size of H0 in bits; 0 ≤ H0_SZ ≤ 16 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-8<br>LENGTH_ADJ | Length Adjustment<br>Signed Adjustment to the LENGTH field for TX and RX. A value of 0 (default) means LENGTH is interpreted as PAYLOAD + CRC |
| 7-6<br>SYNC_ADDR_SZ | Sync Address Size<br>Number of Octets = SYNC_ADDR_SZ + 1, 0 ≤ SYNC_ADDR_SZ ≤ 3. |
| 5<br>LENGTH_BIT_ORD | LENGTH Bit Order<br>Bit order for the LENGTH field of the header<br>0b - LS Bit First<br>1b - MS Bit First |
| 4-0<br>LENGTH_SZ | LENGTH Size<br>Size of LENGTH field of the header, in bits; 0 ≤ LENGTH_SZ ≤ 16 |

## 64.5.3.2.1.21 H0 CONFIGURATION (H0_CFG)

### *64.5.3.2.1.21.1 Offset*

| Register | Offset |
|---|---|
| H0_CFG | 64h |

### *64.5.3.2.1.21.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H0_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H0_MATCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.5.3.2.1.21.3 Fields*

| Field | Function |
|---|---|
| 31-16<br>H0_MASK | H0 Mask Register<br>For each bit that is set to 1, the received H0 field must match the corresponding bit of H0_MATCH[15:0], else the received packet is rejected. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-0<br>H0_MATCH | H0 Match Register<br>For each bit of H0_MASK[15:0] that is set to 1, the received H0 field must match this register, else the received packet is rejected. |

## 64.5.3.2.1.22 H1 CONFIGURATION (H1_CFG)

### *64.5.3.2.1.22.1 Offset*

| Register | Offset |
|---|---|
| H1_CFG | 68h |

### *64.5.3.2.1.22.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H1_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H1_MATCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.5.3.2.1.22.3 Fields*

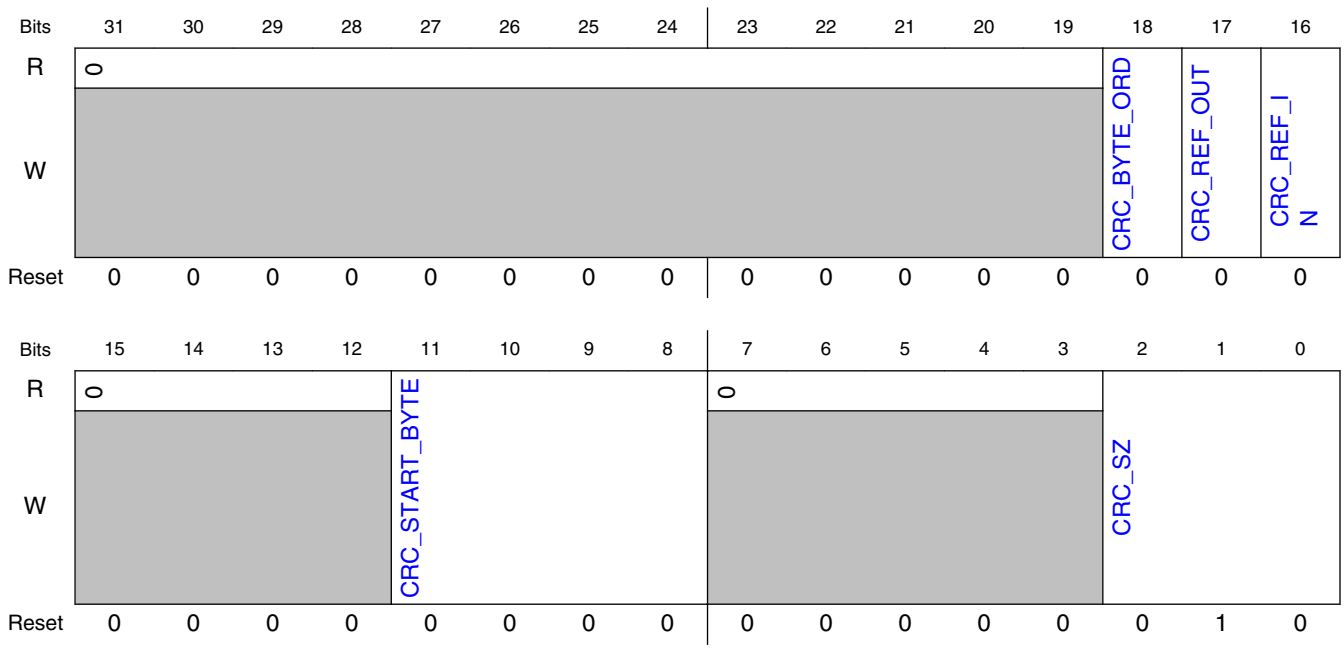
| Field | Function |
|---|---|
| 31-16<br>H1_MASK | H1 Mask Register<br>For each bit that is set to 1, the received H1 field must match the corresponding bit of H1_MATCH[15:0], else the received packet is rejected. |
| 15-0<br>H1_MATCH | H1 Match Register<br>For each bit of H1_MASK[15:0] that is set to 1, the received H1 field must match this register, else the received packet is rejected. |

## 64.5.3.2.1.23 CRC CONFIGURATION (CRC_CFG)

### 64.5.3.2.1.23.1 Offset

| Register | Offset |
|---|---|
| CRC_CFG | 6Ch |

### 64.5.3.2.1.23.2 Diagram




### 64.5.3.2.1.23.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved. |
| 18<br>CRC_BYTE_ORD | CRC Byte Order<br>0b - LS Byte First<br>1b - MS Byte First |
| 17<br>CRC_REF_OUT | CRC Reflect Out<br>0b - do not manipulate CRC result<br>1b - CRC result is to be reflected bitwise (operated on entire word) |
| 16<br>CRC_REF_IN | CRC Reflect In<br>0b - do not manipulate input data stream<br>1b - reflect each byte in the input stream bitwise |
| 15-12<br>— | Reserved. |
| 11-8 | Configure CRC Start Point |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CRC_START_BYTE | Start CRC with this byte position. Byte #0 is the first byte of Sync Address |
| 7-3<br>— | Reserved. |
| 2-0<br>CRC_SZ | CRC Size (in octets)<br>Number of CRC Octets = CRC_SZ, 0 ≤ CRC_SZ ≤ 4. |

## 64.5.3.2.1.24 CRC INITIALIZATION (CRC_INIT)

### *64.5.3.2.1.24.1 Offset*

| Register | Offset |
|---|---|
| CRC_INIT | 70h |

### *64.5.3.2.1.24.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_SEED | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_SEED | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *64.5.3.2.1.24.3 Fields*

| Field | Function |
|---|---|
| 31-0<br>CRC_SEED | CRC Seed Value<br>Initial Value for CRC LFSR |

## 64.5.3.2.1.25 CRC POLYNOMIAL (CRC_POLY)

#### *64.5.3.2.1.25.1 Offset*

| Register | Offset |
|---|---|
| CRC_POLY | 74h |

#### *64.5.3.2.1.25.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POLY | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POLY | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.5.3.2.1.25.3 Fields*

| Field | Function |
|---|---|
| 31-0<br>CRC_POLY | CRC Polynomial. |

## 64.5.3.2.1.26 CRC XOR OUT (CRC_XOR_OUT)

#### *64.5.3.2.1.26.1 Offset*

| Register | Offset |
|---|---|
| CRC_XOR_OUT | 78h |

#### *64.5.3.2.1.26.2 Diagram*

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_XOR_OUT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_XOR_OUT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *64.5.3.2.1.26.3 Fields*

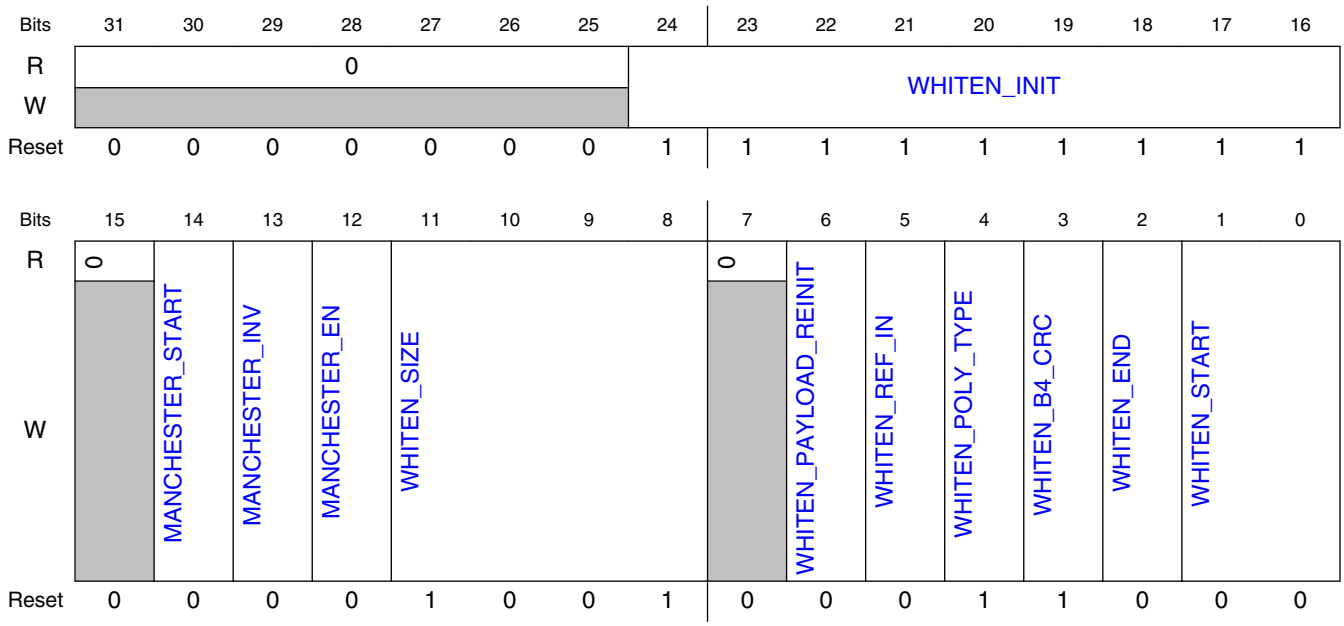
| Field | Function |
|---|---|
| 31-0<br>CRC_XOR_OUT | CRC XOR OUT Register<br>XOR mask for CRC result (for no mask, should be 0) |

## 64.5.3.2.1.27 WHITENER CONFIGURATION (WHITEN_CFG)

#### *64.5.3.2.1.27.1 Offset*

| Register | Offset |
|---|---|
| WHITEN_CFG | 7Ch |

### 64.5.3.2.1.27.2 Diagram




### 64.5.3.2.1.27.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-16<br>WHITEN_INIT | Initialization Value for Whitening/De-whitening |
| 15<br>— | Reserved. |
| 14<br>MANCHESTER_START | Configure Manchester Encoding Start Point<br>0b - Start Manchester coding at start-of-payload<br>1b - Start Manchester coding at start-of-header |
| 13<br>MANCHESTER_INV | Configure for Inverted Manchester Encoding<br>0b - Manchester coding as per 802.3<br>1b - Manchester coding as per 802.3 but with the encoding signal inverted |
| 12<br>MANCHESTER_EN | Configure for Manchester Encoding/Decoding<br>0b - Disable Manchester encoding (TX) and decoding (RX)<br>1b - Enable Manchester encoding (TX) and decoding (RX) |
| 11-8<br>WHITEN_SIZE | Length of Whitener LFSR |
| 7<br>— | Reserved. |
| 6 | Configure for Whitener re-initialization<br>0b - Don't re-initialize Whitener LFSR at start-of-payload |

*Table continues on the next page...*

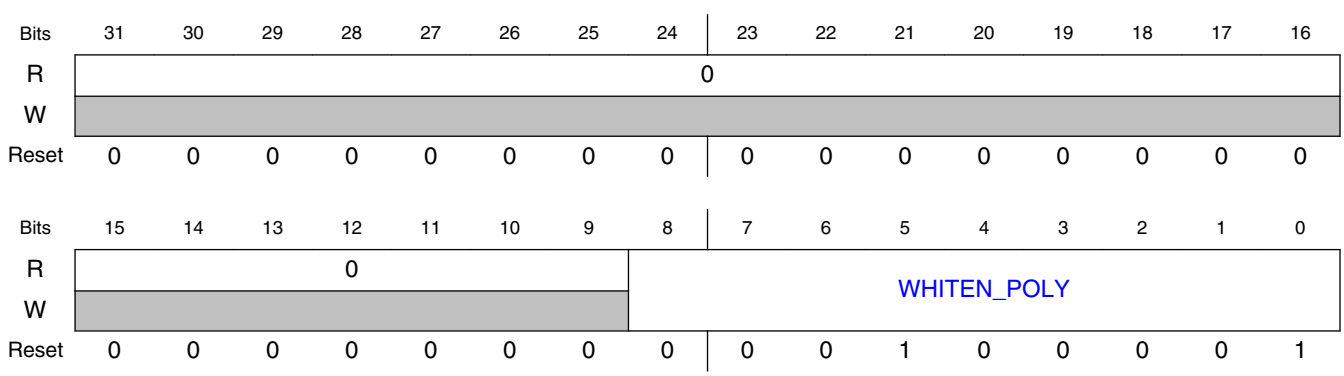
| Field | Function |
|---|---|
| WHITEN_PAYLOAD_REINIT | 1b - Re-initialize Whitener LFSR at start-of-payload |
| 5<br>WHITEN_REF_IN | Whiten Reflect Input<br>The input data stream is reflected, bit-wise, per byte, if this register bit is asserted. Bit 7 becomes bit 0, bit 6 becomes bit 1, etc. This register bit will only cause the reflection of the payload data bits as they are used in the whiten calculation and will not cause any change in the output bit order. |
| 4<br>WHITEN_POLY_TYPE | Whiten Polynomial Type<br>A Fibonacci type LFSR is used with the whiten polynomial if this register bit is asserted. Otherwise, a Galois type LFSR is used. |
| 3<br>WHITEN_B4_CRC | Congifure for Whitening-before-CRC<br>Sets the order of Bit Stream Processing for TX and RX.<br>0b - CRC before whiten/de-whiten<br>1b - Whiten/de-whiten before CRC |
| 2<br>WHITEN_END | Configure end-of-whitening<br>0b - end whiten at end-of-payload<br>1b - end whiten at end-of-crc |
| 1-0<br>WHITEN_START | Configure Whitener Start Point<br>00b - no whitening<br>01b - start whitening at start-of-H0<br>10b - start whitening at start-of-H1 but only if LENGTH > WHITEN_SZ_THR<br>11b - start whitening at start-of-payload but only if LENGTH > WHITEN_SZ_THR |

## 64.5.3.2.1.28 WHITENER POLYNOMIAL (WHITEN_POLY)

### *64.5.3.2.1.28.1 Offset*

| Register | Offset |
|---|---|
| WHITEN_POLY | 80h |

### *64.5.3.2.1.28.2 Diagram*



#### 64.5.3.2.1.28.3 Fields

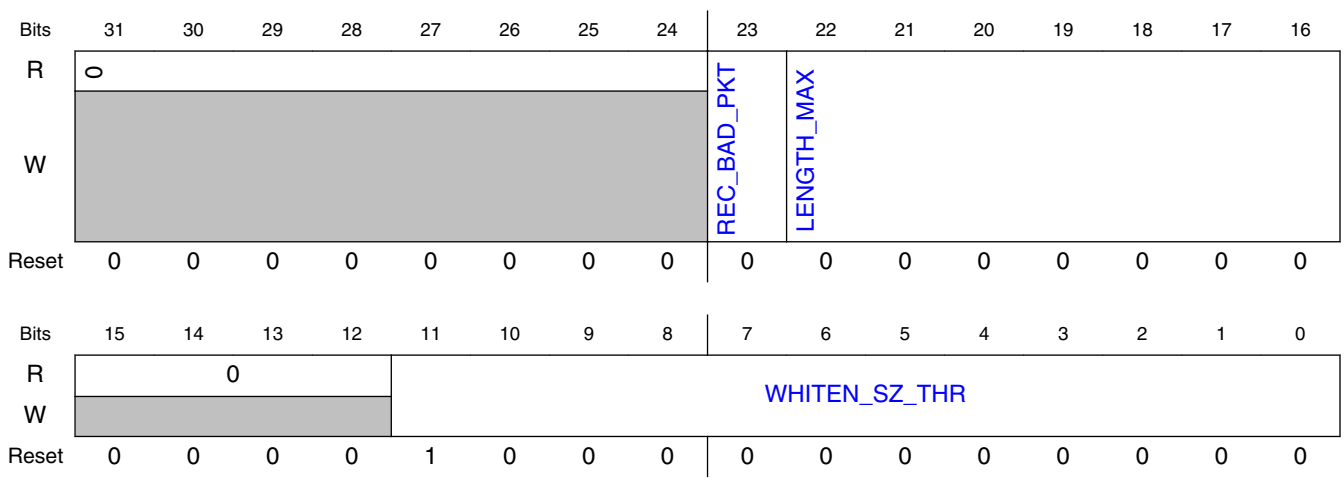
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>WHITEN_POLY | Whitener Polynomial<br>The 9-bit polynomial used in the whiten calculation. The polynomial value must be right-justified if smaller than 9-bits. |

## 64.5.3.2.1.29 WHITENER SIZE THRESHOLD (WHITEN_SZ_THR)

#### 64.5.3.2.1.29.1 Offset

| Register | Offset |
|---|---|
| WHITEN_SZ_THR | 84h |

#### 64.5.3.2.1.29.2 Diagram




#### 64.5.3.2.1.29.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23<br>REC_BAD_PKT | Receive Bad Packets<br>Enable Packets which fail H0-filtering, H1-filtering, or Maximum Length-filtering, to be fully received without an RX recycle (intended for debug purposes) |

*Table continues on the next page...*

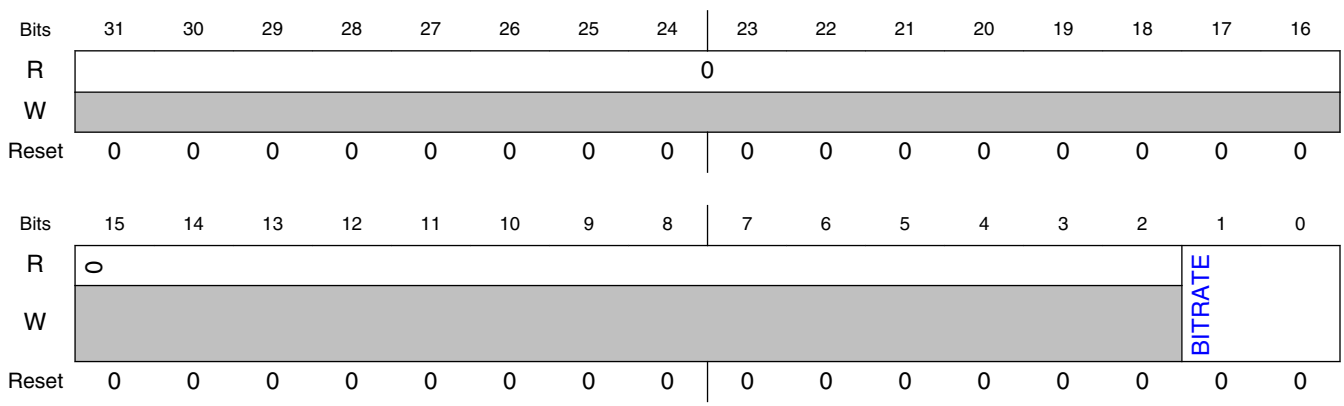
| Field | Function |
|---|---|
| | 0b - packets which fail H0, H1, or LENGTH_MAX result in an automatic recycle after the header is received and parsed<br>1b - packets which fail H0, H1, or LENGTH_MAX are received in their entirety |
| 22-16<br>LENGTH_MAX | Maximum Length for Received Packets<br>Sets the Maximum Length Packet that can be received, in multiples of 16 bytes. LENGTH_MAX is compared directly against the extracted LENGTH field of the header (not the adjusted length). LENGTH_MAX=0 (default) is a special case that implies no limit. |
| 15-12<br>— | Reserved. |
| 11-0<br>WHITEN_SZ_THR | Whitener Size Threshold<br>Minimum Packet Length (extracted LENGTH field) required to enable whiten. Requires WHITEN_START=2 or 3 |

## 64.5.3.2.1.30 BIT RATE (BITRATE)

### *64.5.3.2.1.30.1 Offset*

| Register | Offset |
|---|---|
| BITRATE | 88h |

### *64.5.3.2.1.30.2 Diagram*




### *64.5.3.2.1.30.3 Fields*

| Field | Function |
|---|---|
| 31-2<br>— | Reserved. |

*Table continues on the next page...*

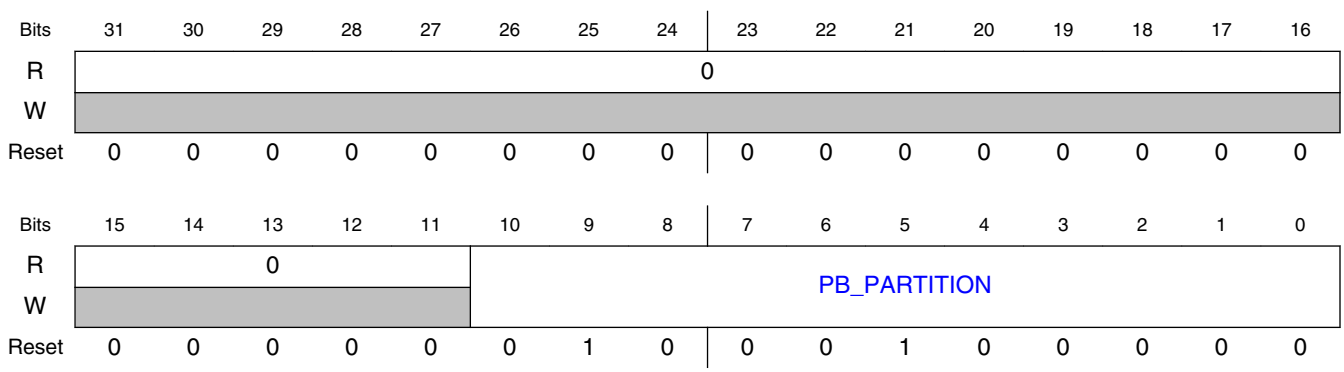
| Field | Function |
|---|---|
| 1-0<br>BITRATE | Bit Rate<br>Selects the Bit Rate for the Generic FSK Link Layer. The blocks which interface to the Link Layer (e.g., PHY, CRC/Whitener), also adapt to the selected bit rate.<br>**NOTE:** For FSK FAD (Fast Antenna Diversity), only the 1Mbit/sec bit rate is supported.<br>00b - 1Mbit/sec<br>01b - 500Kbit/sec<br>10b - 250Kbit/sec (not supported if WHITEN_CFG[MANCHESTER_EN]=1)<br>11b - 2Mbit/sec (not supported if WHITEN_CFG[MANCHESTER_EN]=1) |

## 64.5.3.2.1.31 PACKET BUFFER PARTITION POINT (PB_PARTITION)

### *64.5.3.2.1.31.1 Offset*

| Register | Offset |
|---|---|
| PB_PARTITION | 8Ch |

### *64.5.3.2.1.31.2 Diagram*




### *64.5.3.2.1.31.3 Fields*

| Field | Function |
|---|---|
| 31-11<br>— | Reserved. |
| 10-0<br>PB_PARTITION | Packet Buffer Partition Point<br>The Packet Buffer Partition Point defines the starting point for the RX segment of the Packet Buffer. The partitioning of the consolidated RAM between TX and RX is controlled by this configurable TX/RX Partition Point. PB_PARTITION can be programmed with any value between 0 (base of RAM0) and 1088 (after the last address of RAM1). The consolidated RAM is partitioned such that the TX buffer resides before the Partition Point (RAM entries 0 to (PB_PARTITION-1)), and the RX buffer resides after it (RAM |

| Field | Function |
|---|---|
| | entries PB_PARTITION to 1087). Programming the Partition Point register to 0 dedicates the entire consolidated RAM to RX; programming the Partition Point register to 1088 dedicates the entire consolidated RAM to TX; programming the Partition Point register to any value between 1 and 1087 yields a split TX/RX buffer. The TX segment (if any) is always first in the Packet Buffer, starting at Packet RAM address 0 of RAM0. Since the Packet Buffer RAM word-width is 2 bytes, the units for PB_PARTITION is "words". (Multiply by 2 to convert to byte addressing) |

## 64.5.3.2.1.32 PACKET BUFFER (PACKET_BUFFER_0 - PACKET_BUFFER_1087)

### *64.5.3.2.1.32.1 Offset*

For a = 0 to 1087:

| Register | Offset |
|---|---|
| PACKET_BUFFER_a | 700h + (a × 2h) |

### *64.5.3.2.1.32.2 Diagram*

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PACKET_BUFFER | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### *64.5.3.2.1.32.3 Fields*

| Field | Function |
|---|---|
| 15-0<br>PACKET_BUFFER | PACKET BUFFER RAM<br>Storage for packet data. |

## 64.5.3.2.2 Register Summary

## 64.5.3.2.3 Register Descriptions

| Field | R/W | Description |
|---|---|---|
| SEQCMD[3:0] | w | Sequence Command Register. See Section Sequence Commands and Status for details |
| SEQSTS[12:0] | r | Sequence Status Register. See Section Sequence Commands and Status for details |
| CMDDEC_CS[2:0] | r | Current State of the Command Decoder FSM (debug only) |
| TX_WHITEN_DIS | rw | Disable all whitening on TX packets |
| RX_DEWHITEN_DIS | rw | Disable all de-whitening on RX packets |
| SW_CRC_EN | rw | SW override of the HW-computed CRC for TX. SW must include CRC in the Packet Buffer TX buffer |
| CRC_VALID | r | CRC Valid indicator for RX packets.<br>1: good CRC<br>0: bad CRC |
| TX_WARMUP[7:0] | r | TX warmup time, in microseconds, provided to Link Layer SW |
| RX_WARMUP[7:0] | r | RX warmup time, in microseconds, provided to Link Layer SW |
| CHANNEL_NUM[6:0] | rw | RF Channel Select: 0 <= CHANNEL_NUM <= 127;<br>Formula: F = (2360 + CHANNEL_NUM) [in MHz] |
| TX_POWER[5:0] | rw | PA Power Level. Refer Data Sheet for mapping to dBm. |
| RSSI[7:0] | r | Received Signal Stength Indicator, signed value |
| LQI[7:0] | r | Link Quality Indicator, signed value |
| PREAMBLE_SZ[2:0] | rw | Number of Octets = PREAMBLE_SZ + 1,<br>where 0 <= PREAMBLE_SZ <= 7 |
| BYTE_COUNTER[12:0] | r | Signed value, reflects the current Byte Count, for TX and RX.<br>< 0: Preamble byte(s), TX only<br>0: First Byte of Network Address;<br>1: Second Byte of Network Address<br>… |
| XCVR_BUSY | r | For multi-protocol arbitration, XCVR_BUSY=1 indicates an RF channel access is underway (TSM is busy) |
| EVENT_TMR[23:0] | r +load + add | Event Timer can be read in these byte locations.<br>To update the Event Timer, either:<br>1. Write the desired EVENT_TMR to these bytes and set EVENT_TMR_LD=1, or,<br>2. Write the desired EVENT_TMR *increment* amount to these bytes and set EVENT_TMR_ADD=1<br>Note for EVENT_TMR_ADD, EVENT_TMR[23:0] is a signed value. |
| EVENT_TMR_LD | w | A write access with this bit sets loads EVENT_TMR with the contents of EVENT_TMR[23:0] |
| EVENT_TMR_ADD | w | A write access with this bit sets increments EVENT_TMR by the contents of EVENT_TMR[23:0]. This is a signed addition. |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| TIMESTAMP[23:0] | r | Received Packet Timestamp, Captured from EVENT_TMR at NTW_ADR_IRQ. |
| T1_CMP[23:0] | rw | Timer1 (T1) Compare Value. Can be used to generate T1_IRQ and/or launch Sequence Commands |
| T2_CMP[23:0] | rw | Timer2 (T2) Compare Value. Can be used to generate T2_IRQ and/or launch Sequence Commands |
| T1_CMP_EN | rw | Enable Timer Compare #1 (T1_CMP) to generate T1_IRQ and/or launch Sequence Commands |
| T2_CMP_EN | rw | Enable Timer Compare #2 (T2_CMP) to generate T2_IRQ and/or launch Sequence Commands |
| NTW_ADR_MCH[3:0] | r | Indicates which of the 4 Network Addresses has matched in the PHY. Valid during a RX sequence at the point of match, and remains valid until either:<br>1. The next RX sequence begins (if the current packet passed CRC and header filtering), or,<br>2. An RX recycle to Network Address search (if the current packet failed CRC or header filtering) |
| NTW_ADR0[31:0] | rw | Network Address 0, to PHY for correlation |
| NTW_ADR1[31:0] | rw | Network Address 1, to PHY for correlation |
| NTW_ADR2[31:0] | rw | Network Address 2, to PHY for correlation |
| NTW_ADR3[31:0] | rw | Network Address 3, to PHY for correlation |
| NTW_ADR_EN[3:0] | rw | Enable Network Address N for PHY correlation, where 0 <= N <= 3. Any bit combination can be set. |
| NTW_ADR0_SZ[1:0] | rw | 3: Network Address 0 requires a 32-bit correlation<br>2: Network Address 0 requires a 24-bit correlation<br>1: Network Address 0 requires a 16-bit correlation<br>0: Network Address 0 requires a 8-bit correlation |
| NTW_ADR1_SZ[1:0] | rw | 3: Network Address 1 requires a 32-bit correlation<br>2: Network Address 1 requires a 24-bit correlation<br>1: Network Address 1 requires a 16-bit correlation<br>0: Network Address 1 requires a 8-bit correlation |
| NTW_ADR2_SZ[1:0] | rw | 3: Network Address 2 requires a 32-bit correlation<br>2: Network Address 2 requires a 24-bit correlation<br>1: Network Address 2 requires a 16-bit correlation<br>0: Network Address 2 requires a 8-bit correlation |
| NTW_ADR3_SZ[1:0] | rw | 3: Network Address 3 requires a 32-bit correlation<br>2: Network Address 3 requires a 24-bit correlation<br>1: Network Address 3 requires a 16-bit correlation<br>0: Network Address 3 requires a 8-bit correlation |
| NTW_ADR_THR0[2:0] | rw | Number of Tolerated bit errors for Network Address 0 |
| NTW_ADR_THR1[2:0] | rw | Number of Tolerated bit errors for Network Address 1 |
| NTW_ADR_THR2[2:0] | rw | Number of Tolerated bit errors for Network Address 2 |
| NTW_ADR_THR3[2:0] | rw | Number of Tolerated bit errors for Network Address 3 |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| RX_WATERMARK_IRQ | r / w1tc | Asserts when Byte Counter == RX_WATERMARK[12:0] |
| NTW_ADR_IRQ | r / w1tc | Network Address Match Interrupt. A NA Match has occurred |
| TX_IRQ | r / w1tc | TX Interrupt. The TX sequence has completed with a successful packet transmission. |
| RX_IRQ | r / w1tc | RX Interrupt. The RX sequence has completed with a successful packet reception. |
| PLL_UNLOCK_IRQ | r / w1tc | PLL Unlock Interrupt. An unlock event has occurred. |
| SEQ_END_IRQ | r / w1tc | Sequence End Interrupt. Will assert when any TX or RX sequence ends for any reason. |
| T1_IRQ | r / w1tc | Timer Compare #1 (T1) Interrupt |
| T2_IRQ | r / w1tc | Timer Compare #2 (T2) Interrupt |
| WAKE_IRQ | r / w1tc | **For Manual DSM:** This is WAKE_IRQ. A WAKE_IRQ will be triggered when the GENERIC_FSK Link Layer Controller has awoken from a Manual DSM (Deep Sleep Mode) cycle. WAKE_IRQ indicates that the RF Oscillator has been restarted, and the GENERIC_FSK EVENT_TMR has resumed counting.<br><br>**For Wake-On-Radio:** This is WOR_IRQ. See Section Wake-On-Radio |
| TSM_IRQ | r | TSM Interrupt. Indicates TSM0_IRQ or TSM1_IRQ is set in XCVR_STATUS. For debug purposes. |
| RX_WATERMARK_IRQ_EN | rw | Enable for RX_WATERMARK_IRQ |
| NTW_ADR_IRQ_EN | rw | Enable for NTW_ADR_IRQ |
| TX_IRQ_EN | rw | Enable for TX_IRQ |
| RX_IRQ_EN | rw | Enable for RX_IRQ |
| PLL_UNLOCK_IRQ_EN | rw | Enable for PLL_UNLOCK_IRQ |
| SEQ_END_IRQ_EN | rw | Enable for SEQ_END_IRQ |
| T1_IRQ_EN | rw | Enable for T1_IRQ |
| T2_IRQ_EN | rw | Enable for T2_IRQ |
| WAKE_IRQ_EN | rw | Enable for WAKE_IRQ |
| TSM_IRQ_EN | rw | Enable for TSM_IRQ |
| GENERIC_FSK_IRQ_EN | rw | Master enable for the GENERIC_FSK_IRQ interrupt line to the MCU. |
| CRC_IGNORE | rw | If set, assert RX_IRQ even for a received packet which fails CRC verification, and do not recycle. |
| RX_WATERMARK[12:0] | rw | Sets the trigger for RX_WATERMARK_IRQ.<br>Trigger the RX_WATERMARK_IRQ when:<br>RX Byte Counter == RX_WATERMARK[12:0] |
| GEN_SLEEP_REQUEST | w | Enable a match on MAN_SLEEP[23:0] to DSM_TIMER[23:0], to enter Deep Sleep Mode |
| DSM_TIMER[23:0] | r | Current State of the 32 KHz Sleep Timer<br>**NOTE:** This register resides in RSIM Address Space |
| PART_ID[7:0] | r | Part ID to identify HW revision |
| SYNC_ADDR_SZ[1:0] | rw | Number of Octets = SYNC_ADDR_SZ + 1, 0 <= SYNC_ADDR_SZ <= 3. Applies to TX only. |
| H0_SZ[4:0] | rw | Size of H0 in bits; 0 <= H0_SZ <= 16 |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| H0_MASK[15:0] | rw | For each bit that is set to 1, the received H0 field must match the corresponding bit of H0_MATCH[15:0], else the received packet is rejected. |
| H0_MATCH[15:0] | rw | For each bit of H0_MASK[15:0] that is set to 1, the received H0 field must match this register, else the received packet is rejected. |
| LENGTH_SZ[4:0] | rw | Size of Packet Length in bits; 0 <= LENGTH_SZ <= 16 |
| LENGTH_BIT_ORD | rw | Bit order for the LENGTH field<br>0: LS Bit First;<br>1: MS Bit First |
| LENGTH_ADJ[7:0] | rw | Signed Adjustment to the LENGTH field for TX and RX. A value of 0 (default) means LENGTH is interpreted as PAYLOAD + CRC |
| LENGTH_ADJ_UNSIGNED | rw | Length Adjustment Unsigned Enabled<br>0: Hardware interprets LENGTH_ADJ as a signed integer (default)<br>1: Hardware interprets LENGTH_ADJ as a unsigned integer |
| LENGTH_MAX[6:0] | rw | Sets the Maximum Length Packet that can be received, in multiples of 16 bytes. LENGTH_MAX is compared directly against the extracted LENGTH field of the header (not the adjusted length). LENGTH_MAX=0 (default) is a special case that implies no limit. |
| REC_BAD_PKT | rw | 1: packets which fail H0, H1, or LENGTH_MAX are received in their entirety<br>0: packets which fail H0, H1, or LENGTH_MAX result in a recycle after the header is fully received |
| LENGTH_FAIL | r | For packets received with REC_BAD_PKT=1, LENGTH_FAIL indicates the extracted LENGTH header field exceeded LENGTH_MAX |
| H0_FAIL | r | For packets received with REC_BAD_PKT=1, H0_FAIL indicates the received H0 header field violates the H0_MASK/H0_MATCH pattern |
| H1_FAIL | r | For packets received with REC_BAD_PKT=1, H1_FAIL indicates the received H1 header field violates the H1_MASK/H1_MATCH pattern |
| H1_SZ[4:0] | rw | Length of H1 in bits; 0 <= H1_SZ <= 16 |
| H1_MASK[15:0] | rw | For each bit that is set to 1, the received H1 field must match the corresponding bit of H1_MATCH[15:0], else the received packet is rejected. |
| H1_MATCH[15:0] | rw | For each bit of H1_MASK[15:0] that is set to 1, the received H1 field must match this register, else the received packet is rejected. |
| CRC_SZ[2:0] | rw | Number of CRC Octets = CRC_SZ, 0 <= CRC_SZ <= 4. |
| CRC_POLY[31:0] | rw | CRC Polynomial.<br>1: XOR exists in the bit's feedback path;<br>0: no XOR in the bit's feedback path |
| CRC_SEED[31:0] | rw | CRC Seed Value |
| CRC_START_BYTE[3:0] | rw | Start CRC with this byte position. Byte #0 is the first byte of Sync Address |
| CRC_BYTE_ORD | rw | 0: LS Byte First<br>1: MS Byte First |
| CRC_REF_IN | rw | 0: do not manipulate input data stream; |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| | | 1: reflect each byte in the input stream bitwise |
| CRC_REF_OUT | rw | 0: do not manipulate CRC result;<br>1: CRC result is to be reflected bitwise (operated on entire word) |
| CRC_XOR_OUT[31:0] | rw | XOR mask for CRC result (for no mask, should be 0) |
| WHITEN_B4_CRC | rw | Order of Bit Stream Processing for TX and RX:<br>1: Whiten/de-whiten before CRC<br>0: CRC before whiten/de-whiten |
| WHITEN_SEED[8:0] | rw | Initialization Value for Whitening/De-whitening |
| WHITEN_ POLY[8:0] | rw | Polynomial Value for Whitening/De-whitening |
| WHITEN_POLY_TYPE | rw | Whiten polynomial type. A Fibonacci type LFSR is used with the whiten polynomial if this bit is asserted. Otherwise, a Galois type LFSR is used. |
| WHITEN_SIZE[3:0] | rw | Whitener Length |
| WHITEN_REF_IN | rw | The input data stream is reflected, bit-wise, per byte, if this bit is asserted. Bit 7 becomes bit 0, bit 6 becomes bit 1, etc. This bit will only cause the reflection of the payload data bits as they are used in the whiten calculation and will not cause any change in the output bit order |
| WHITEN_START[1:0] | rw | 0: no whiten;<br>1: start whiten @ start-of-H0;<br>2: start whiten @ start-of-H1 but only if LENGTH > WHITEN_SZ_THR<br>3: start whiten @ start-of-payload but only if LENGTH > WHITEN_SZ_THR |
| WHITEN_END | rw | 0: end whiten at end-of-payload<br>1: end whiten at end-of-crc |
| WHITEN_PAYLOAD_REINIT | rw | 1: re-assert whiten_init at start-of-payload<br>0 = don't re-assert whiten_init |
| WHITEN_SZ_THR[11:0] | rw | Minimum Packet Length (extracted LENGTH field) required to enable whiten. Requires WHITEN_START=2 or 3 |
| BITRATE[1:0] | rw | 0: 1 Mbs<br>1: 500 Kbs<br>2: 250 Kbs<br>3: 2 Mbs |
| MANCHESTER_EN | rw | 1: Enable Manchester encoding (TX) and decoding (RX)<br>0: Disable Manchester Encoding and Decoding |
| MANCHESTER_INV | rw | 0: Manchester coding as per 802.3<br>1: Manchester coding as per 802.3 but with the encoding signal inverted |
| MANCHESTER_START | rw | 0: Start Manchester coding @ start-of-payload<br>1: Start Manchester coding @ start-of-header |

## 64.5.3.3 Functional Description

### 64.5.3.3.1 Packet Configuration

The Generic FSK Link Layer Controller provides the capability to configure the over-the-air packet structure by way of a set of programmable registers. The Generic FSK packet has a consistent structure, which consists of various elements which are transmitted and received in fixed order. Most of the packet elements have associated configurability, which allows software to pre-configure, for example, the size of the element (in number of bits, or octets), or bit ordering of the element. Other configuration options apply to the entire packet, such as bitrate. In addition, the generic structure defines a variable-length header, for which some (optional) primitive parsing is provided, allowing hardware to differentiate packets based on header-bit settings, into compliant and non-compliant packets. Data Indication is provided on compliant packets only. Non-complaint packets are discarded. For transmitting and receiving, multiple bitrate options are provided. By default, all packet elements are transmitted and received LSB-first.

The Generic FSK Link Layer Controller supports a packet structure based on the following template:

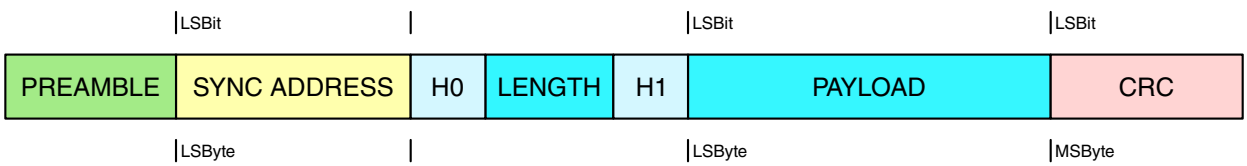



**Figure 64-167. GENERIC_FSK Packet Structure**

The Generic_FSK Link Layer controller transmits and receives packets which conform to the format shown above. The elements of the packet are transmitted in received in the order shown, although some of the packet elements may be optionally skipped, depending on the configuration options chosen.

The following diagram depicts the register configurability associated with the Preamble, Sync Address, H0, Length, and H1 elements of the packet structure.

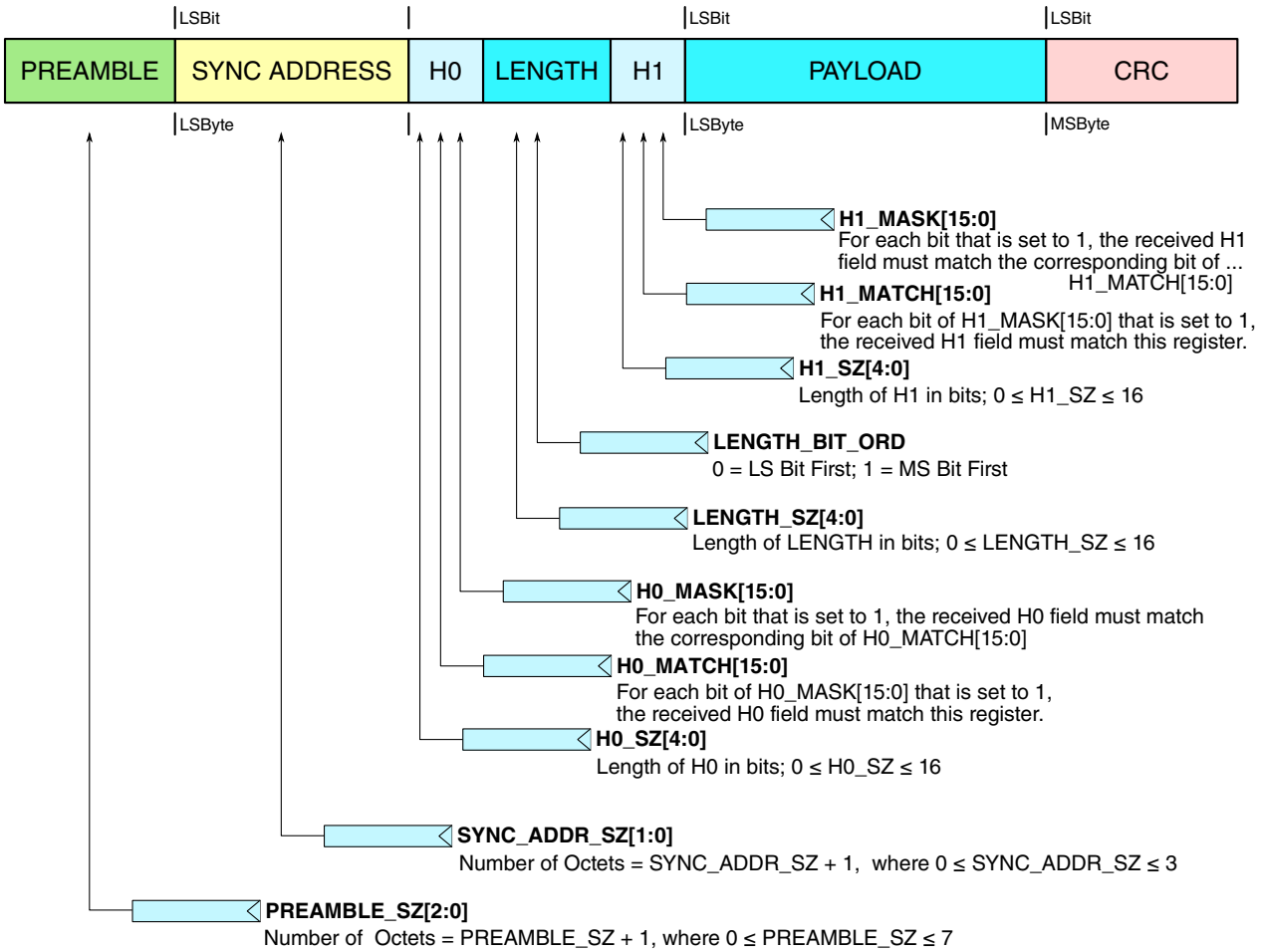



**Figure 64-168. Packet Structure Definitions (Part 1)**

The following diagram depicts the register configurability associated with the Payload and CRC elements of the packet structure.

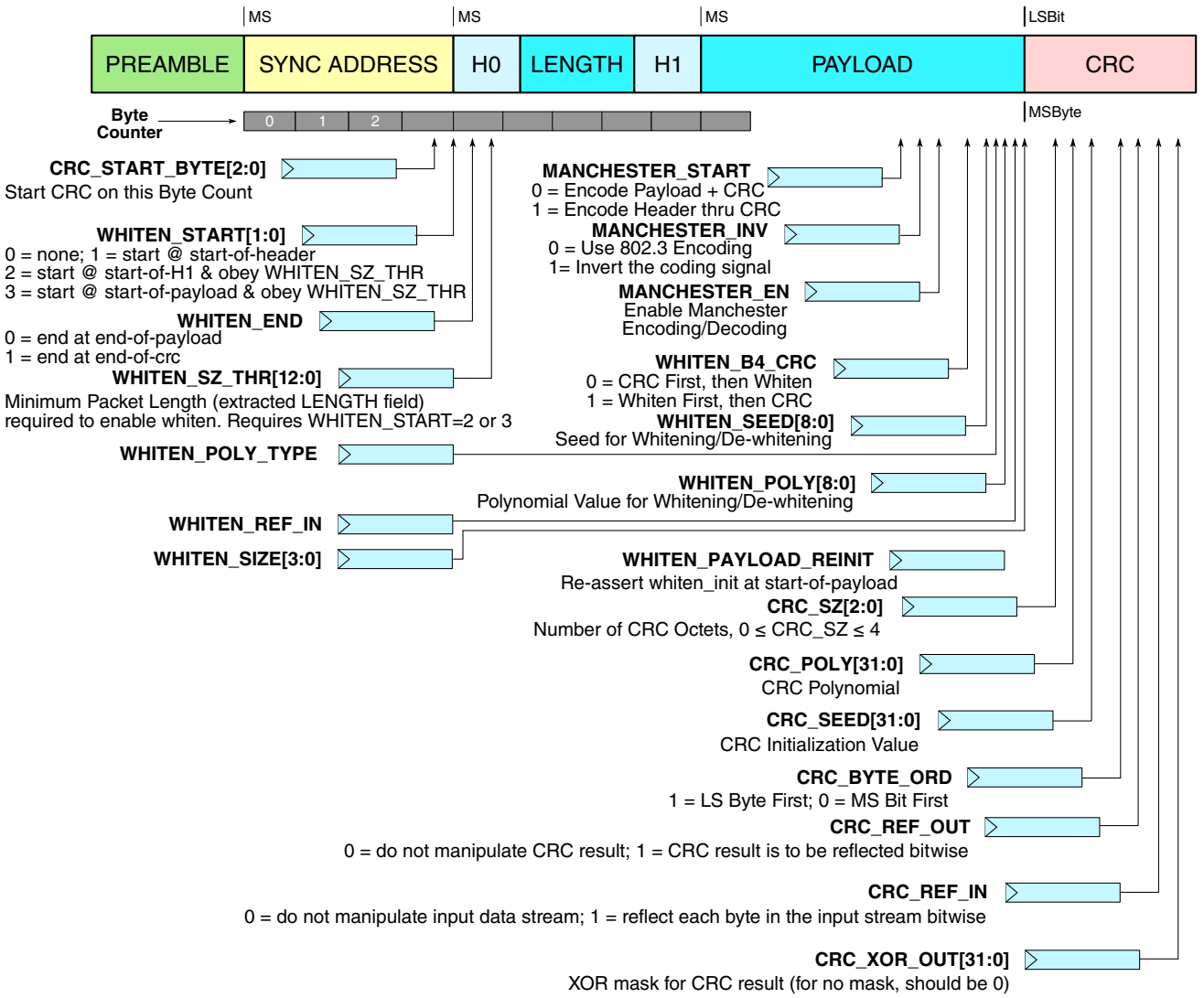



**Figure 64-169. Packet Structure Definitions (Part 2)**

The following subsections describe each of the packet elements, and the configurability associated with each.

## PREAMBLE

The preamble pattern is 0x55 or 0xAA. Like all packet elements, preamble is transmitted LSB first. The Sync Address, which immediately follows the preamble, determines which of the 2 preamble options is selected by hardware: the controller hardware selects the preamble pattern based on the first transmitted bit of Network Address, such that the last bit of preamble is the opposite polarity from the first bit of Network Address, forcing a bit transition at this boundary. The number of octets of preamble is programmable, via the PREAMBLE_SZ[2:0] register, with a range from 1 to 8 octets. A setting of PREAMBLE_SZ=0 yields a single preamble octet, whereas a setting of PREAMBLE_SZ=7 yields 8 octets. For transmission, the Generic FSK

BYTE_COUNTER will be negative during preamble transmission. If PREAMBLE_SZ=0 (1 preamble octet), BYTE_COUNTER will be -1 while the single preamble octet is being transmitted. If PREAMBLE_SZ=7 (8 octets), BYTE_COUNTER will be -8 during the first preamble octet, -7 during the next, and so on and so forth, until the eighth and final octet, during which the BYTE_COUNTER will be -1. Unlike most packet elements, the preamble is generated and shifted out by hardware (only preamble and CRC are hardware-generated); there is no software setup required, and preamble does not appear in the Packet Buffer TX buffer. During reception, the Link Layer controller does not receive preamble, since preamble detection is performed by the PHY; preamble does not appear in the Packet Buffer RX buffer.

## SYNC (NETWORK) ADDRESS

Sync Address, known synonymously as Network Address or Access Address depending on the protocol in use, is the second packet element transmitted by the Link Layer controller, and the first element received. In a network of associated devices, all devices generally share a common Network Address. The first bit of Network Address determines the preamble selection for transmission, as noted above. For TX, Network Address is the first packet element programmed into the Packet Buffer TX buffer, in LSB-first format. For RX, Network Address is also the first element stored into the Packet Buffer RX buffer, in LSB-first format. The register NTW_ADRx_SZ[1:0] determines the length, in octets, of the Network Address field in the packet structure. A setting of NTW_ADRx_SZ=0 yields a single-octet Network Address, whereas a setting of NTW_ADRx_SZ=3 yields 4 octets. For reception, up to four unique Network Addresses are supported. Any combination of the 4 can be simultaneously searched for. Network Address correlation is performed in the PHY. Multiple Network Address capability allows GENERIC_FSK devices to be associated to more than one network simultaneously, if the network topology supports it; this feature also enables multi-protocol operation. Each of the 4 Network Address options has its own enable bit, NTW_ADR_EN[x]; when the enable bit for a Network Address is set to 1, the PHY is instructed to search for that Network Address pattern in the demodulated data, and when the enable bit is set to 0, the PHY will disregard that pattern. Each of the 4 Network Address options has its own NTW_ADRx_SZ register, to program its octet size during reception. Additionally, each of the 4 Network Address options has a register, NTW_ADR_THRx[2:0], which determine the number of bit errors that can be tolerated by the PHY during Network Address correlation to the respective Network Address option. Finally, when the PHY detects a match to one of the 4 Network Address options, with a number of bit errors less than or equal to the threshold programmed into NTW_ADR_THRx, a read only status bit, NTW_ADR_MCH[x] becomes set, indicating which of the 4 patterns matched. After a match has been detected, the PHY shall hold NTW_ADR_MCH[x] valid (sticky), until the next receiver warmup, or the next RX recycle. (Specifically, the TSM output rx_init clears the NTW_ADR_MCH[x] bits; TSM

will assert this signal during either an RX warmup or a recycle). A table listing the registers which apply to Network Address Management, for each of the 4 Network Address options, is shown below.

| NTW_ADR PATTERN REGISTER | NTW_ADR ENABLE BIT | NTW_ADR SIZE REGISTER | NTW_ADR THRESHOLD REGISTER | NTW_ADR PATTERN-MATCH STATUS BIT |
|---|---|---|---|---|
| NTW_ADR_0[31:0] | NTW_ADR_EN[0] | NTW_ADR0_SZ[1:0] | NTW_ADR_THR0[2:0] | NTW_ADR_MCH[0] |
| NTW_ADR_1[31:0] | NTW_ADR_EN[1] | NTW_ADR0_SZ[1:0] | NTW_ADR_THR1[2:0] | NTW_ADR_MCH[1] |
| NTW_ADR_2[31:0] | NTW_ADR_EN[2] | NTW_ADR2_SZ[1:0] | NTW_ADR_THR2[2:0] | NTW_ADR_MCH[2] |
| NTW_ADR_3[31:0] | NTW_ADR_EN[3] | NTW_ADR3_SZ[1:0] | NTW_ADR_THR3[2:0] | NTW_ADR_MCH[3] |

For Network Addresses of size less than 4 octets, the PHY expects the desired pattern to occupy the least-significant byte positions of the NTW_ADR_x register. Unused byte positions may be filled with any value.

**NOTE**

When programming the NTW_ADR_x, NTW_ADR_EN[x], NTW_ADRx_SZ, and NTW_ADR_THRx for any RX operation, the following software restrictions apply:

1. All NTW_ADR_x patterns which are enabled by their respective NTW_ADR_EN[x]=1 must be unique (disabled NTW_ADR_x registers need not follow this restriction).
2. Given restriction #1 above, not only must the patterns be unique, but the bit patterns of NTW_ADR_x and NTW_ADR_y must differ by an amount greater than the sum of the bit-error thresholds for each pattern, i.e., NTW_ADR_THRx + NTW_ADR_THRy.
3. Given all combinations of NTW_ADR_x and NTW_ADR_y, where (x != y) and both Network Addresses are enabled by NTW_ADR_EN[x]=NTW_ADR_EN[y]=1, if the sizes of the Network Address patterns differ because NTW_ADRx_SZ != NTW_ADRy_SZ, then the “matching portion” of NTW_ADR_x and NTW_ADR_y, that is, the octets of the pattern which the PHY is instructed to correlate to for each pattern, programmed into the lower byte positions of NTW_ADR_x or NTW_ADR_y for patterns of size less than 4 octets, must be unique.
4. Given restriction #3 above, not only must the “matching portion” of the patterns be unique, but bit patterns of the matching portions of NTW_ADR_x and NTW_ADR_y must differ by greater than the sum of the bit-error thresholds for each pattern, i.e., NTW_ADR_THRx + NTW_ADR_THRy.

These restrictions are designed to preclude multiple, simultaneous pattern matches during a pattern search. Results are indeterminate if the restrictions are violated.

When the PHY detects a pattern match on any enabled NTW_ADR_x during reception, and asserts NTW_ADR_MCH[x], the GENERIC_FSK Link Layer Controller shall assert NTW_ADR_IRQ. The NTW_ADR_IRQ is the Link Layer's first indication that a packet is being received. After any NTW_ADR_IRQ assertion, a packet will be received and stored into the Packet Buffer RX buffer.

## HEADER

The packet header is comprised of H0, LENGTH, and H1, in that order. Each of the 3 fields of the header has programmable length, from 0 to 16 bits. A size of zero for any of the header components infers that that component is skipped in the packet structure. Although the size of the individual H0, LENGTH, and H1 components need not be aligned to a byte boundary, the overall header must be byte-aligned. That is, the sum of the sizes (in bits) of H0, LENGTH, and H1, must be a integer multiple of 8 bits. This is a software restriction, a violation of which may result in indeterminate behavior.

For GENERIC_FSK protocol, the purpose of the header is twofold:

1. To frame the LENGTH field, so that LENGTH can be extracted and interpreted by the GENERIC_FSK packet processor, so as to determine how and when perform the required bit-stream processing on the data. In most wireless protocols with regular structures, the LENGTH field is embedded within the packet header, but its positioning, number of LENGTH bits, bit ordering, and interpretation can differ. The GENERIC_FSK configurability thus provides support for a wide range of header formats, yielding a high likelihood that most current and future packet formats and LENGTH fields will be parsable by the GENERIC_FSK Link Layer controller.
2. To provide rudimentary packet filtering on the non-LENGTH header bits, namely H0 and H1. Screening of packets based on H0 and/or H1 bit comparisons can reduce power consumption, by avoiding notification to the host processor on received packets that do not comply with pre-configured H0 and/or H1 bitmasks, enabling longer MCU sleep durations.

The header components, and their configurability options, are further described below.

### H0

H0 is an optional header packet element. Its length in bits is determined by register H0_SZ[4:0]. Legal values for H0_SZ are:

0 <= H0_SZ <= 16

If H0 is present (non-zero size), it can be filtered-on by the GENERIC_FSK packet processor; each bit of H0 can be forced to match a programmable bitmask, or ignored. Every bit of H0 that is forced to match the bitmask, must match; otherwise, the packet is rejected. The registers H0_MATCH[15:0] and H0_MASK[15:0] control H0 filtering. For every bit of H0_MASK that is set to one, the received bit in the H0 portion of the header

must match the corresponding bit of H0_MATCH. Bits of H0_MASK which are cleared, do not require a match of received data bits to H0_MATCH bits. If any bit of received data which is qualified with H0_MASK[x]=1 fails to match its corresponding H0_MATCH[x] bit, the H0_FAIL register status bit will be set, an RX recycle will occur, and the host processor will not be notified. (The recycle operation will self-clear the H0_FAIL bit). For debug purposes, the control bit REC_BAD_PKT has been provided. If REC_BAD_PKT=1 and a H0-mismatch occurs, the RX recycle is prevented, the full packet is received, the MCU is notified, and the H0_FAIL flag stays asserted. In this case, the H0-fail flag will stay asserted until the next RX warmup (specifically, rx_init clears the bit).

The equation for H0_FAIL is thus:

$H0_FAIL = [H0_{received} \text{\^{}} H0_MATCH[15:0]) \ \& \ H0_MASK > 0$

## LENGTH

LENGTH is an optional header packet element. Its length in bits is determined by the register LENGTH_SZ[4:0]. Legal values for LENGTH_SZ are:

0 <= LENGTH_SZ <= 16

If LENGTH is present (non-zero size), its value determines the number of octets remaining in the packet after the header is complete, that is, PAYLOAD octets plus CRC octets, as depicted below.

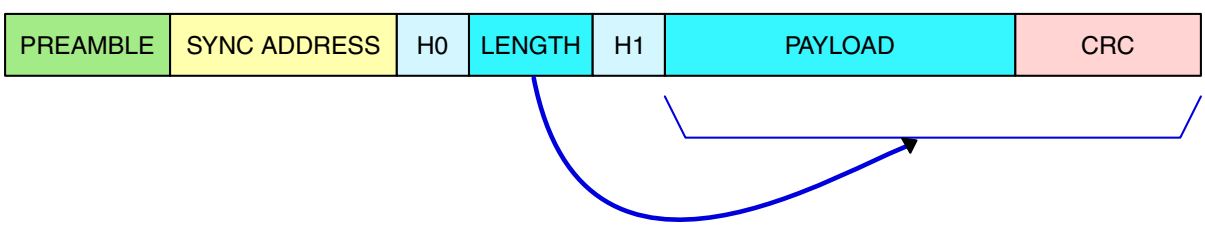



**Figure 64-170. LENGTH Field Interpretation with No LENGTH_ADJ**

For example, if the payload consists of 16 octets, and CRC is 4 octets, the LENGTH field of the header should be set to 20. The packet processor will extract the packet length from the LENGTH field of the header, and use this information to determine the timing of the CRC, Whitening, and Manchester control signals, as well as end-of-packet interrupt triggering.

In case a protocol exists for which the LENGTH field is to be interpreted differently, a configurability option has been provided to compensate for this. Hypothetically, this could happen, for example, in a proprietary packet structure, in which LENGTH is to be

defined to represent PAYLOAD only, excluding CRC. The register LENGTH_ADJ, allows a fixed offset to be applied to the extracted LENGTH field, such that the sum of LENGTH + LENGTH_ADJ represents the number of PAYLOAD + CRC octets, as depicted below:

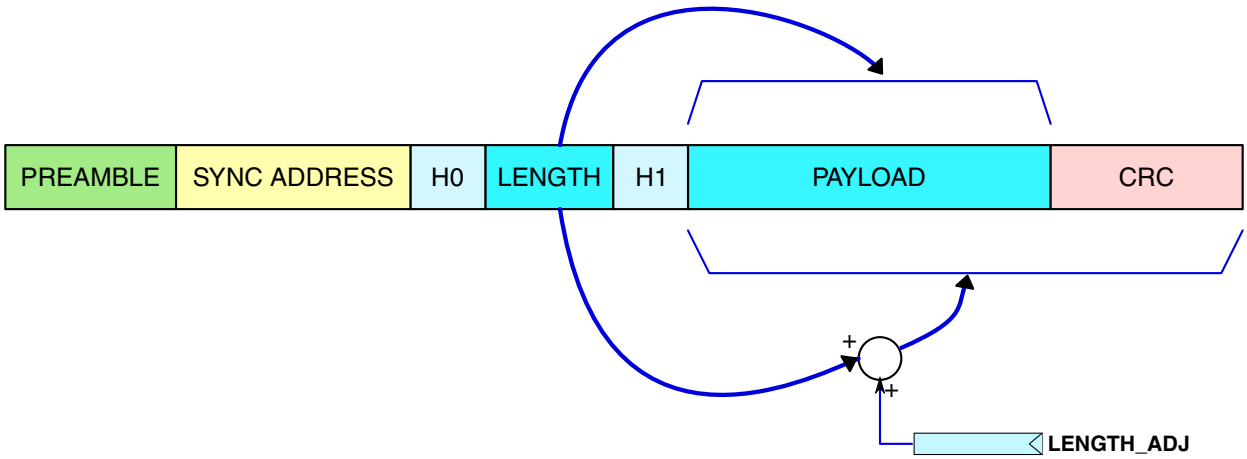



**Figure 64-171. LENGTH Field Interpretation with LENGTH_ADJ**

Applying the previous example, if, using a proprietary format, LENGTH were to be defined as payload-length only, LENGTH would be set to 16. In order for the packet processor to correctly time the CRC and Whitening control signals, etc., the LENGTH_ADJ register should be programmed to 4 in this case, so that the packet processor would operate on a 20-octet “PAYLOAD + CRC” PDU. The LENGTH_ADJ register represents a signed value, so that negative offsets can be applied. The legal range for LENGTH_ADJ is:

-31 <= LENGTH_ADJ <= 31

The LENGTH field could be ordered MSB first or LSB first. The packet processor must be able to accommodate either case, in order to extract the LENGTH field properly from the header, so a configurability option has been provided. The register bit LENGTH_BIT_ORD specifies the bit order. If LENGTH_BIT_ORD=0, ordering is LSB-first, as shown below.

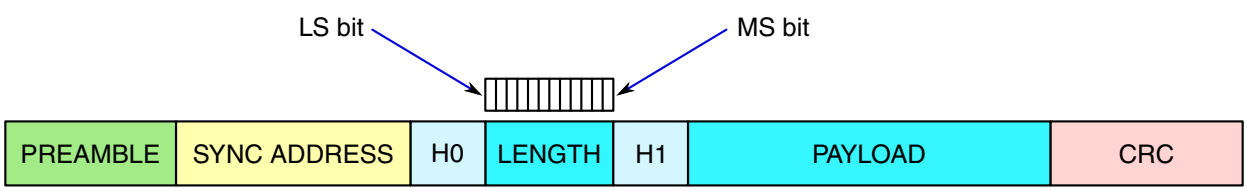



**Figure 64-172. LENGTH Bit Ordering (LSB First)**

If LENGTH_BIT_ORD=1, ordering is MSB-first, as shown below.

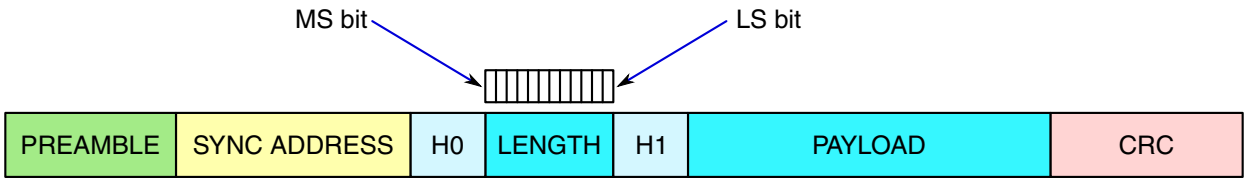



**Figure 64-173. LENGTH Bit Ordering (MSB First)**

If LENGTH_SZ=0, then LENGTH is not present in the header. The packet processor interprets this as LENGTH=0. If, in this case, LENGTH_ADJ=0 also, the packet processor interprets this to mean that the packet ends with the last bit of H1, and there are no PAYLOAD or CRC octets (CRC_SZ should be programmed to 0 in this case). There is no CRC verification in this scenario. If, in this case, LENGTH_SZ=0, meaning an empty payload field, but a CRC check is desired, CRC_SZ should be programmed to the desired number of CRC octets, and LENGTH_ADJ should be programmed to the same value: LENGTH_ADJ = CRC_SZ.

For GENERIC_FSK, valid payload lengths are from 0 to 2047 octets, i.e.,

0 <= LENGTH <= 2047

If LENGTH_ADJ is non-zero, then the maximum limit of 2047 applies to the sum of LENGTH and LENGTH_ADJ, i.e.,

0 <= (LENGTH + LENGTH_ADJ) <= 2047

A restriction on LENGTH_ADJ follows:

If the packet ends immediately after the LENGTH field of the header, i.e., H1_SZ=0 and there is neither payload nor CRC, then LENGTH_ADJ must be set to 0.

Although the LENGTH field has a maximum size of 16 bits, since the maximum receivable packet size is 2047 bytes, any received LENGTH bits above bit [10] are ignored by the packet processor, which effectively clamps LENGTH at 2047 bytes. There may be cases where it is desirable to limit the maximum received packet length to values less than 2047, so that received packets whose extracted LENGTH field indicates to be larger than a pre-set limit, are rejected by the packet processor. A configurability option has been provided for this, the register LENGTH_MAX[6:0]. The LENGTH_MAX setting configures a pre-set limit on the extracted LENGTH field, in multiples of 16 octets. A special case exists for LENGTH_MAX=0 (the register default), which implies no hardware-limiting; this setting should be used to allow reception of full-length 2047-octet packets. If LENGTH_MAX=1, packets with LENGTH >= 16 octets are rejected. If LENGTH_MAX=2, packets with LENGTH >= 32 octets are rejected. If LENGTH_MAX=127 (the maximum setting) packets with LENGTH >= 2032 octets are rejected. To determine compliance with LENGTH_MAX, the packet processor compares

the LENGTH_MAX register with the extracted LENGTH field directly, not the adjusted length (LENGTH + LENGTH_ADJ). If a packet is received during which the extracted LENGTH field exceeds the LENGTH_MAX setting, the LENGTH_FAIL register status bit will be set, an RX recycle will occur, and the host processor will not be notified. (The recycle will self-clear the LENGTH_FAIL bit). For debug purposes, the control bit REC_BAD_PKT has been provided. If REC_BAD_PKT=1 and a LENGTH_MAX violation occurs, the RX recycle is prevented, the full packet is received, the MCU is notified, and the LENGTH_FAIL flag stays asserted. In this case, the LENGTH_FAIL flag will stay asserted until the next RX warmup (specifically, rx_init clears the bit).

### H1

H1 is an optional header packet element. Its length in bits is determined by register H1_SZ[4:0]. Legal values for H1_SZ are $0 \leq H1_SZ \leq 16$. If H1 is present (non-zero size), it can be filtered-on by the GENERIC_FSK packet processor; each bit of H1 can be forced to match a programmable bitmask, or ignored. Every bit of H1 that is forced to match the bitmask, must match; otherwise, the packet is rejected. The registers H1_MATCH[15:0] and H1_MASK[15:0] control H1 filtering. For every bit of H1_MASK that is set to one, the received bit in the H1 portion of the header must match the corresponding bit of H1_MATCH. Bits of H1_MASK which are cleared, do not require a match of received data bits to H1_MATCH bits. If any bit of received data which is qualified with H1_MASK[x]=1 fails to match its corresponding H1_MATCH[x] bit, the H1_FAIL register status bit will be set, an RX recycle will occur, and the host processor will not be notified. (The recycle will self-clear the H1_FAIL bit). For debug purposes, the control bit REC_BAD_PKT has been provided. If REC_BAD_PKT=1 and a H1-mismatch occurs, the RX recycle is prevented, the full packet is received, the MCU is notified, and the H1_FAIL flag stays asserted. In this case, the H1-fail flag will stay asserted until the next RX warmup (specifically, rx_init clears the bit).

The equation for H1_FAIL is thus:

$H1_FAIL = [H1_{received} \text{^} H1_MATCH[15:0]) \ \& \ H1_MASK > 0$

### PAYLOAD

In the GENERIC_FSK packet structure, the payload directly followed the header. For transmission, payload bytes are serialized, and transmitted, LSB first, in order, as they are received from the Packet Buffer interface. During reception, payload bytes are received LSB first, undergo serial-to-parallel conversion to octets, 4 octets are packed into a 32-bit word, and the word is transferred to memory via Packet Buffer. For both TX and RX, since the packet processor interprets the extracted LENGTH to represent the number of Payload + CRC octets, the actual number of payload octets is computed from the following equation:

$Number_of_Payload_Octets = (LENGTH_{extracted} + LENGTH_ADJ - CRC_SZ)$

For TX, payload bytes are subjected to whitening or Manchester encoding. They are also shifted through CRC to compute a CRC checksum value. Whitening, encoding, and CRC computing are all optional. Whitening and Manchester encoding are mutually exclusive. For RX, payload bytes are subjected to de-whitening, Manchester decoding, and CRC verification. De-whitening, Manchester decoding, and CRC verification are all optional. De-whitening and Manchester decoding are mutually exclusive. For both TX and RX, (de-)whitening can precede CRC, or follow it. Manchester encoding always operates on post-CRC bitstream for transmission, and Manchester decoding always precedes CRC verification for reception. For more details on CRC, whitening, and Manchester encoding, see Section Bitstream Processing.

### CRC

In the GENERIC_FSK packet structure, the CRC field directly followed the payload. The size of the CRC field, in octets, is set by the CRC_SZ register. The valid range for CRC_SZ is:

0 <= CRC_SZ <= 4

For both TX and RX, the portion of the packet which are subjected to CRC is programmable, via the register CRC_START_BYTE[3:0]. The register CRC_START_BYTE programs the absolute byte count of the first byte for which CRC shifting is desired. The CRC_START_BYTE value is referenced to the GENERIC_FSK Byte Counter. The Byte Counter starts at 0 for the first octet of Sync Address, and increments octet-for-octet henceforth, until the last octet of CRC is transmitted or received. An example of CRC_START_BYTE programming is shown below.

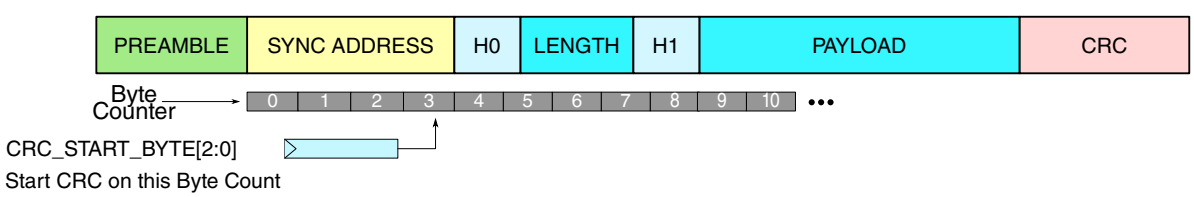



**Figure 64-174. CRC Start Byte**

In this example

SYNC_ADDR_SZ = 4

Header Size = 4 i.e. (H0_SZ + LENGTH_SZ +H1_SZ)/8 = 4

To initiate CRC shifting at the start of Sync Address, program CRC_START_BYTE = 0

To initiate CRC shifting at the start of Header, program CRC_START_BYTE = 4

To initiate CRC shifting at the start of Payload, program CRC_START_BYTE = 9

The CRC_START_BYTE should not be programmed so large as to exceed the length of the PDU; that is, larger than extracted LENGTH – CRC_SZ. If CRC_SZ=0, program CRC_START_BYTE=0.

For more details on how the GENERIC_FSK packet processor controls the CRC generation and verification processes, see Section Bitstream Processing.

## EXAMPLE

A real-world example follows, describing how the GENERIC_FSK Link Layer packet configuration can be applied to transmit and receive packets which conform to IEEE 802.15.4g . For this example, please refer to IEEE Std 802.15.4g™-2012, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), Amendment 3: Physical Layer (PHY) Specifications for Low-Data-Rate, Wireless, Smart Metering Utility Networks.

Section 18.1.1.3 of 802.15.4g describes the packet header (PHR), which is replicated below as it appears in that standard:

**Table 64-48. Format of the PHR(without mode switching) for MR-FSK**

| Bit string index | 0 | 1-2 | 3 | 4 | 5-15 |
|---|---|---|---|---|---|
| Bit mapping | MS | $R_1$-$R_0$ | FCS | DW | $L_{10}$-$L_0$ |
| Field Name | Mode Switch | Reserved | FCS Type | Data Whitening | Frame Length |

The GENERIC_FSK Link Layer controller supports such a header format, and can be programmed to transmit and receive packets which conform to it. To correctly size and order the GENERIC_FSK header elements so as to conform with this header format:

1. The LENGTH field ends the header, so there is no H1 (H1_SZ=H1_MASK=0)
2. The 5 bits which precede LENGTH in the header, are grouped into H0 (H0_SZ=5)
3. The size of the LENGTH field is 11 bits (LENGTH_SZ=11)
4. The LENGTH is specified as MSB-first (LENGTH_BIT_ORD=1)

For this example, we would like to leverage the H0-filtering capability built into the GENERIC_FSK Link Layer to screen the 802.15.4g header, to require that the bit fields in the received packet comply with the following conditions:

1. MS (Mode Switch, bit [0] of header) = 0
2. DW (Data Whitening, bit [4]) = 1
3. FCS (FCS Type, bit [3]) = 1 (Four-octet FCS)
4. Bits [1] and [2] are marked as Reserved, to be ignored on reception

Given these parameters, the following diagram shows how we would program the GENERIC_FSK Link Layer controller to accept this packet configuration, and then screen H0 accordingly:

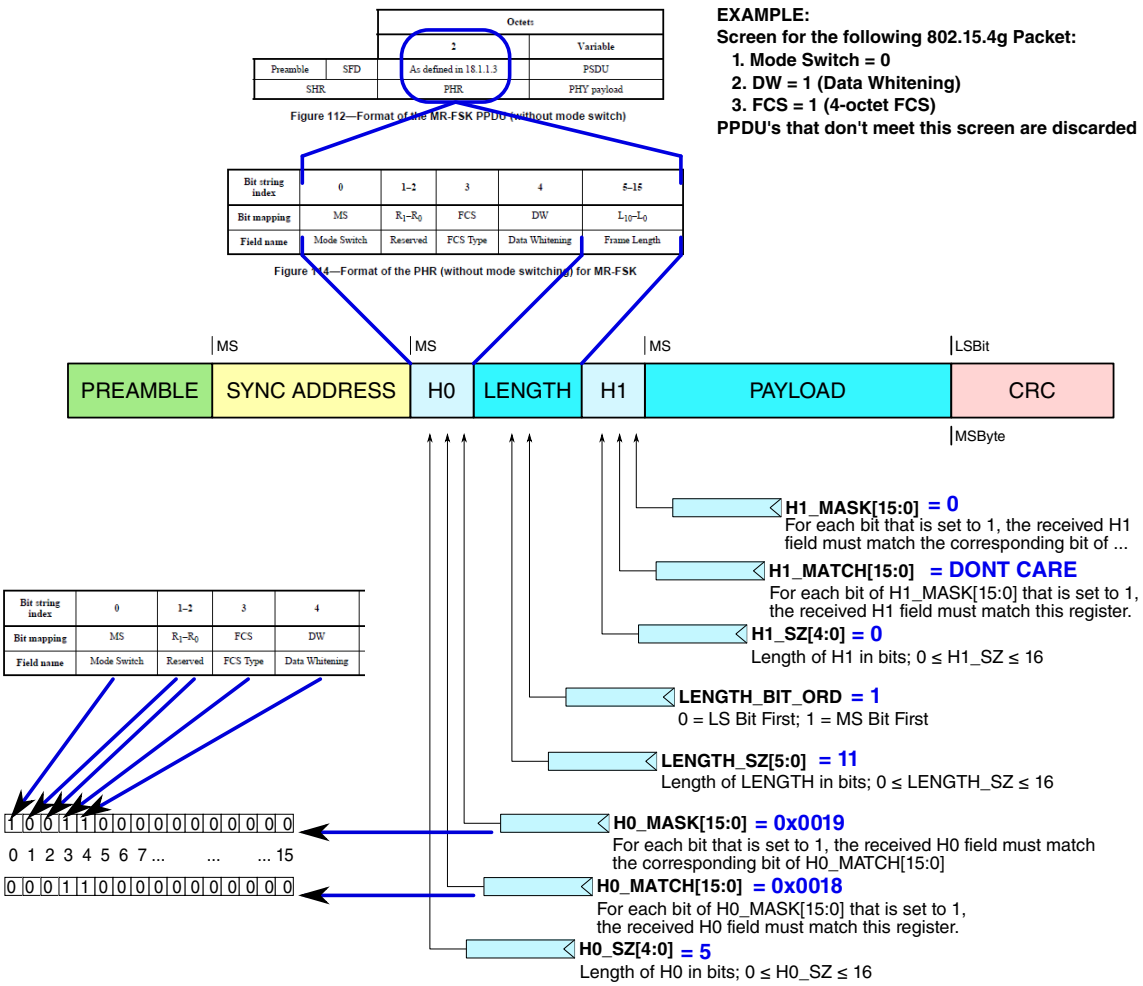



**Figure 64-175. GENERIC FSK PACKET STRUCTURE AS APPLIED TO 802.15.4g PHR**

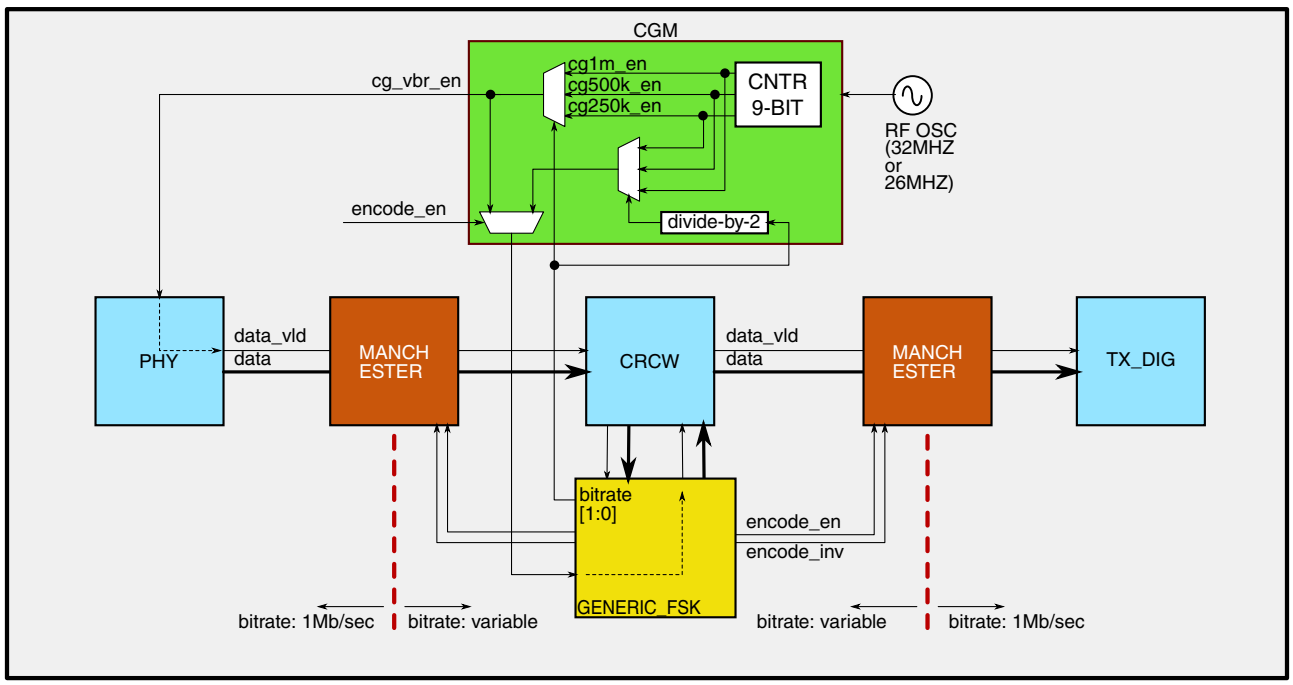
Received packets which meet the H0 filtering screen are fully received, and Data Indication is provided to the host processor. Packets which fail the screen result in an RX recycle, and no notification to the host.

## 64.5.3.3.2 Bitstream Processing

The GENERIC_FSK Link Layer controller provides a number of options to allow for control and modification of the serial bitstream as the packet is being transmitted and received. From a high-level point-of-view, these 4 options are provided:

1. CRC generation (TX) and CRC verification (RX)
2. Data Whitening (TX) and De-whitening (RX)
3. Manchester Encoding (TX) and Decoding (RX)
4. Variable Bit Rate

The following diagram provides another perspective of bitstream processing.




**BITSTREAM PROCESSING WITH MANCHESTER ENCODING**

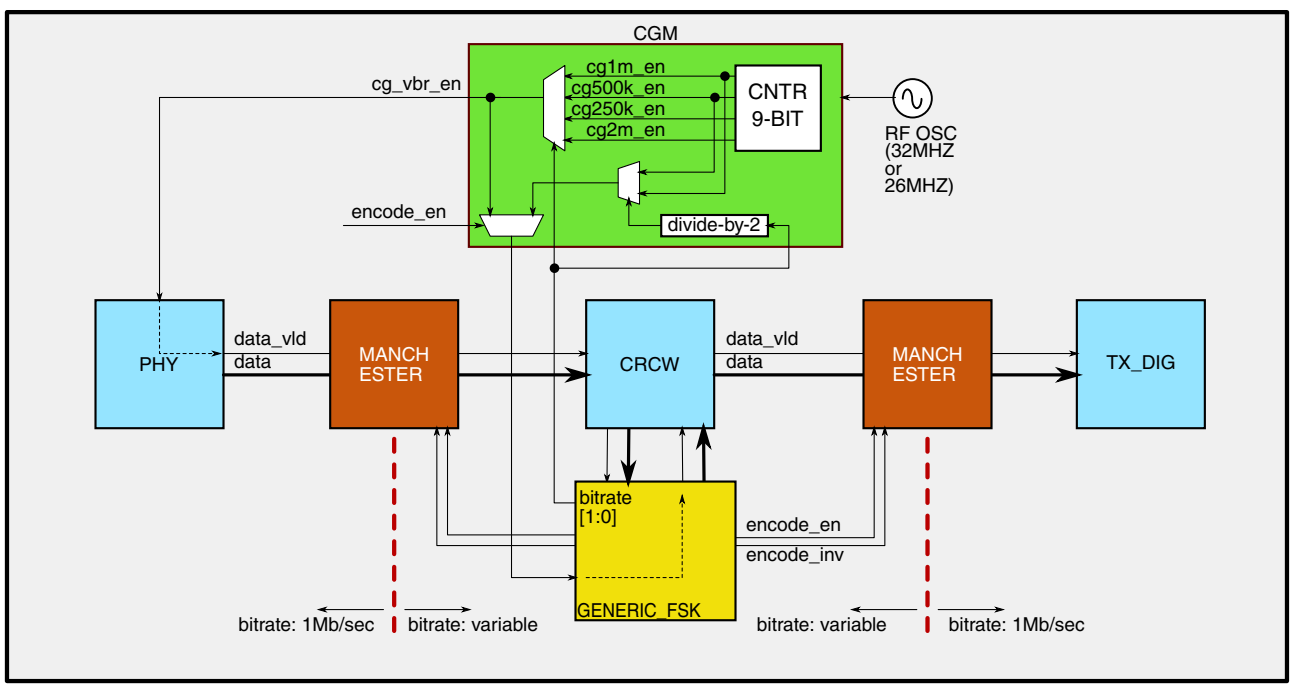



**Figure 64-176. BITSTREAM PROCESSING WITH MANCHESTER ENCODING**

For GENERIC_FSK, CRC and Whitening are somewhat consolidated, and are performed in a separate module external to the Link Layer Controller. The CRC/Whitener block performs these functions. Manchester encoding is also performed externally, and is described later in this section. Bitrate configurability is managed jointly by the

GENERIC_FSK Link Layer controller and the 2.4GHz Radio's Clock Generation Module (CGM). The following sections provide more detail on how CRC, whitening, Manchester encoding, and bitrate, are controlled by the GENERIC_FSK Link Layer, and describes the configuration options associated with each.

## CRC & WHITENING

The GENERIC_FSK Link Layer controller dynamically manages the CRC generation process (for TX), and CRC verification process (for RX), by way of 3 timing signals that it asserts to the CRC/Whitener module.

For GENERIC_FSK, the CRC engine in the CRC/Whitener block shall only shift incoming bits (data_in) through its LFSR when all the following conditions are met:

1. crcw_init=0
2. crc_en=1
3. tx_data_in_vld=1 (signal is data_in_vld at the CRC/Whitener)

The CRC/Whitener shall only shift bits out of its CRC LFSR, to the TX digital block, when all the following conditions are met:

1. crcw_init=0
2. crc_comp=1
3. tx_data_in_vld=1 (signal is data_in_vld at the CRC/Whitener)

The packet processor uses the extracted LENGTH field of the packet, as well as the registers which control the size of the individual packet elements, in order to determine when to assert the CRC-related control signals, **crc_en** and **crc_comp**. This is true for both transmission and reception. The calculations used by the packet processor to determine when to assert the CRC-related control signals are summarized in the following diagram.

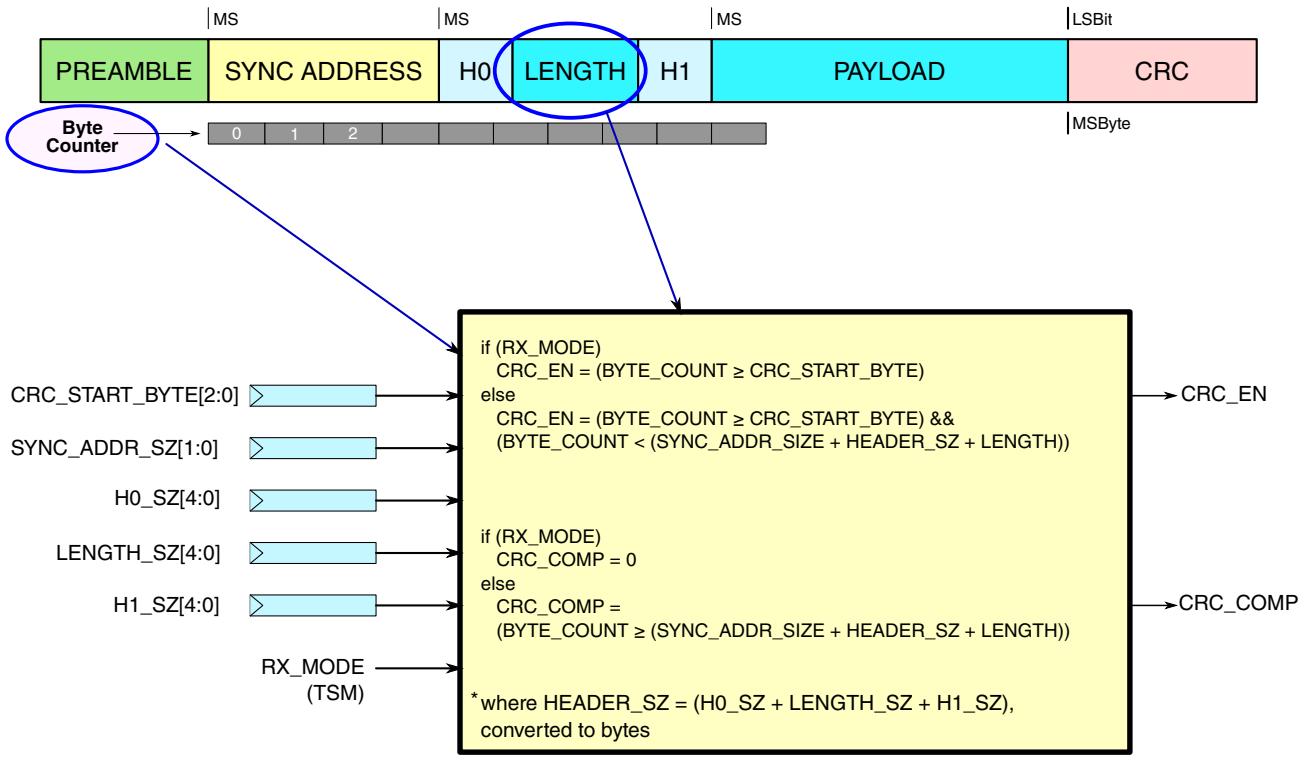



**Figure 64-177. GENERIC FSK: CONTROL OF CRC**

Likewise, the GENERIC_FSK Link Layer controller optionally enables whitening and de-whitening of the packet bitstream, and controls the portions of the packet which are subjected to whitening. For both TX and RX, the portion of the packet which are subjected to whitening/de-whitening is programmable; the following table describes the 3 registers which determine which elements of the packet bitstream are whitened/de-whitened.

| REGISTER NAME | FUNCTIONAL DESCRIPTION |
|---|---|
| WHITEN_START[1:0] | **0**: No whitening/de-whitening<br>**1**: Begin whitening/de-whitening at the start of the header (H0) field, regardless of packet length.<br>**2**: Begin whitening/de-whitening at the start of the header H1 field, but only if the extracted length field meets or exceeds WHITEN_SZ_THR, i.e.,<br>($LENGTH_{extracted}$ >= WHITEN_SZ);<br>otherwise, there is no whitening/de-whitening of the bitstream<br>**3**: Begin whitening/de-whitening at the start of the PAYLOAD, but only if the extracted length field meets or exceeds WHITEN_SZ_THR, i.e.,<br>($LENGTH_{extracted}$ >= WHITEN_SZ);<br>otherwise, there is no whitening/de-whitening of the bitstream |

*Table continues on the next page...*

| REGISTER NAME | FUNCTIONAL DESCRIPTION |
|---|---|
| WHITEN_END | **0**: Whitening/de-whitening ends with the last bit of the last PAYLOAD byte<br>**1**: Whitening/de-whitening ends with the last bit of the last CRC byte |
| WHITEN_SZ_THR[12:0] | If WHITEN_START=2 or WHITEN_START=3, apply whitening/de-whitening at the point determined by WHITEN_START, but only if the extracted LENGTH field of the packet meets or exceeds this WHITEN_SZ_THR threshold; if LENGTHextracted < WHITEN_SZ_THR, there is no whitening/de-whitening of the bitstream. If WHITEN_START < 2, this register has no effect. |

The GENERIC_FSK Link Layer controller dynamically manages the data whitening process (for TX), and de-whitening (for RX), via 2 timing signals that it asserts to the CRC/Whitener module.

For GENERIC_FSK, the Whitening/de-whitening engine in the CRC/Whitener block shall only shift incoming bits (data_in) through its LFSR when all the following conditions are met:

1. crcw_init=0
2. whiten_en=1
3. tx_data_in_vld=1 (signal is data_in_vld at the CRC/Whitener)

The packet processor uses the extracted LENGTH field of the packet, as well as the registers which control the size of the individual packet elements, in order to determine when to assert the whitening-related control signal, whiten_en. This is true for both transmission and reception. The calculations used by the packet processor to determine when to assert whiten_en control signals are summarized in the following diagram.

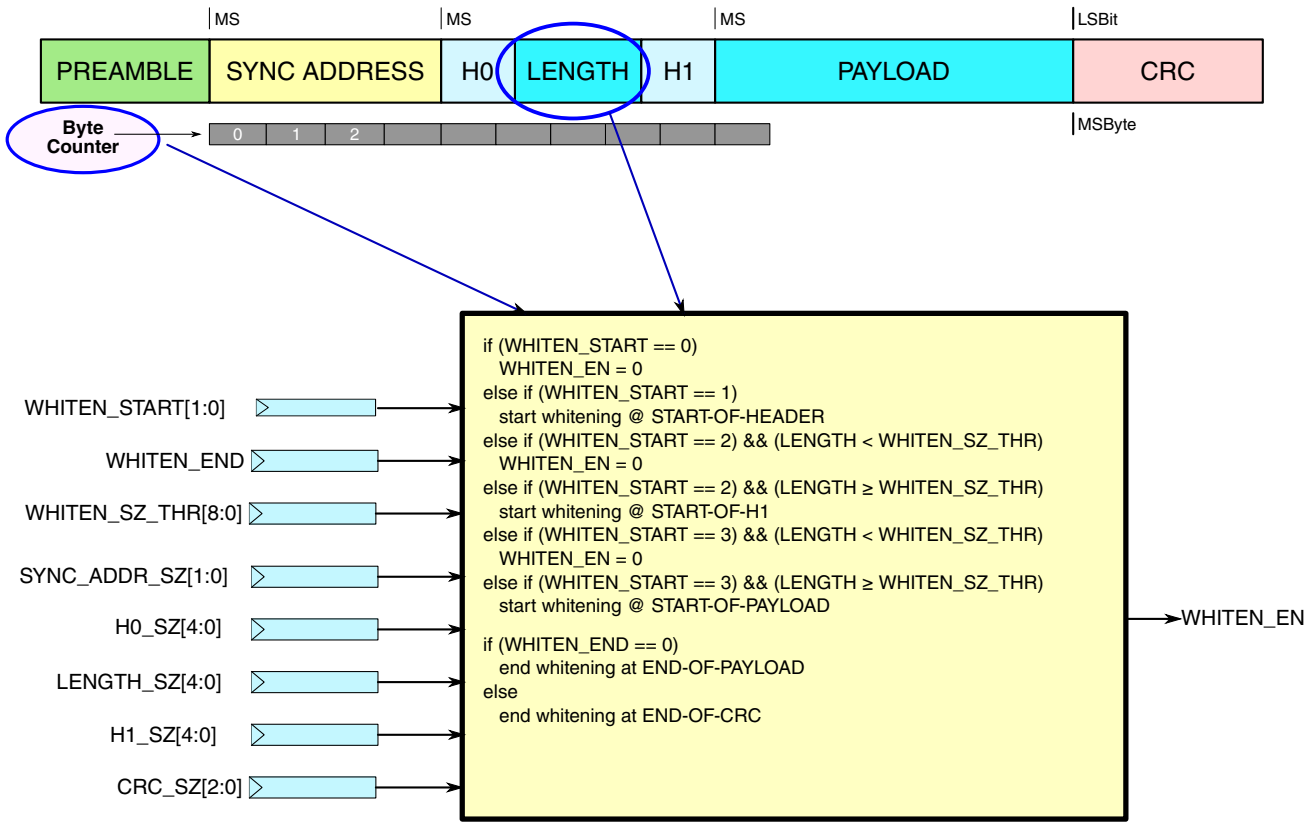



**Figure 64-178. GENERIC FSK: CONTROL OF WHITENING**

The order in which CRC and whitening are performed, is configurable, via the static register bit WHITEN_B4_CRC. This bit is programmed in the GENERIC_FSK Link Layer, and sent to the CRC/Whitening engine to control the ordering. The following table describes how the bitstream is processed for both settings of WHITEN_B4_CRC, for both TX and RX.

| | TX | RX |
|---|---|---|
| WHITEN_B4_CRC=1 | In the CRC/Whitener, the CRC engine shifts in data bits which have been whitened first through the whitener LSFR. The entire post-whitened bitstream is then transmitted to the TX Digital. | Since the entire bitstream, including CRC, has been pre-whitened, the incoming bitstream from the PHY is shifted directly through the CRC LFSR to verify the checksum. The entire bitstream is simultaneously de-whitened by the whitener LFSR and shifted out to the Link Layer. |
| WHITEN_B4_CRC=0 | In the CRC/Whitener, the CRC engine shifts in data bits which have not been whitened. Whitening is performed on the CRC LFSR output. The entire post-whitened bitstream is then transmitted to the TX Digital. | The entire bitstream is de-whitened first by the whitener LFSR, and then shifted through the CRC LFSR to verify the checksum. The entire de-whitened bitstream is simultaneously shifted out to the Link Layer. |

A diagram depicting an example of bitstream processing flow during packet transmission, with WHITEN_B4_CRC=1 (the register default), is shown below.

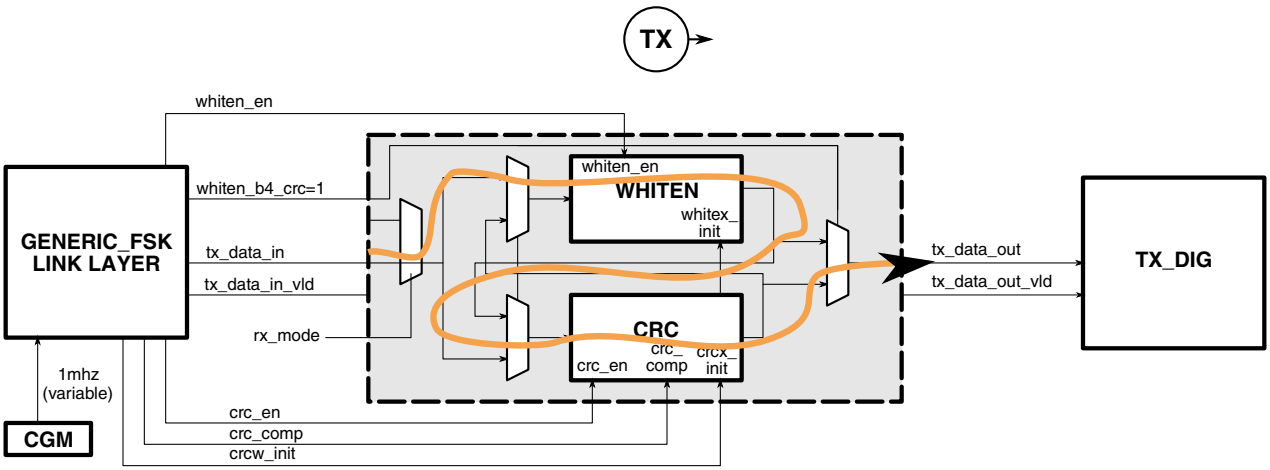



**Figure 64-179. Whitening Before CRC (TX)**

A diagram depicting an example of bitstream processing flow during packet reception, with WHITEN_B4_CRC=1, is shown below.

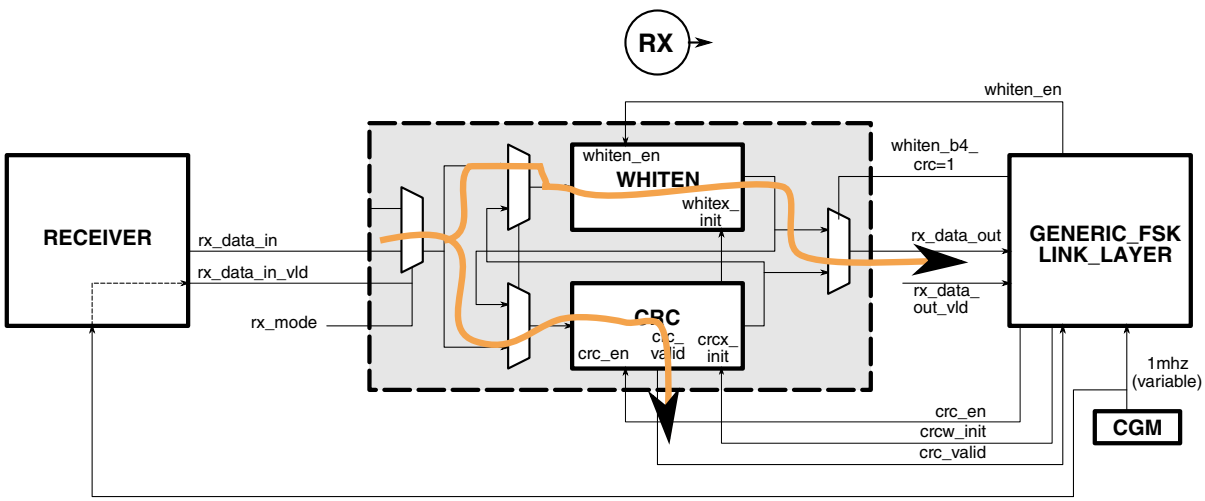



**Figure 64-180. Whitening Before CRC (RX)**

A timing diagram depicting a real-world example of TX bitstream processing timing is shown below. In this example, the desired start point for CRC shifting is the start of Network Address, so CRC_START_BYTE=0. The desired start point for whitening is the start of the header (H0), so WHITEN_START=1. It is also desired to stop whitening after the last bit of the last PAYLOAD byte, so WHITEN_END=0.

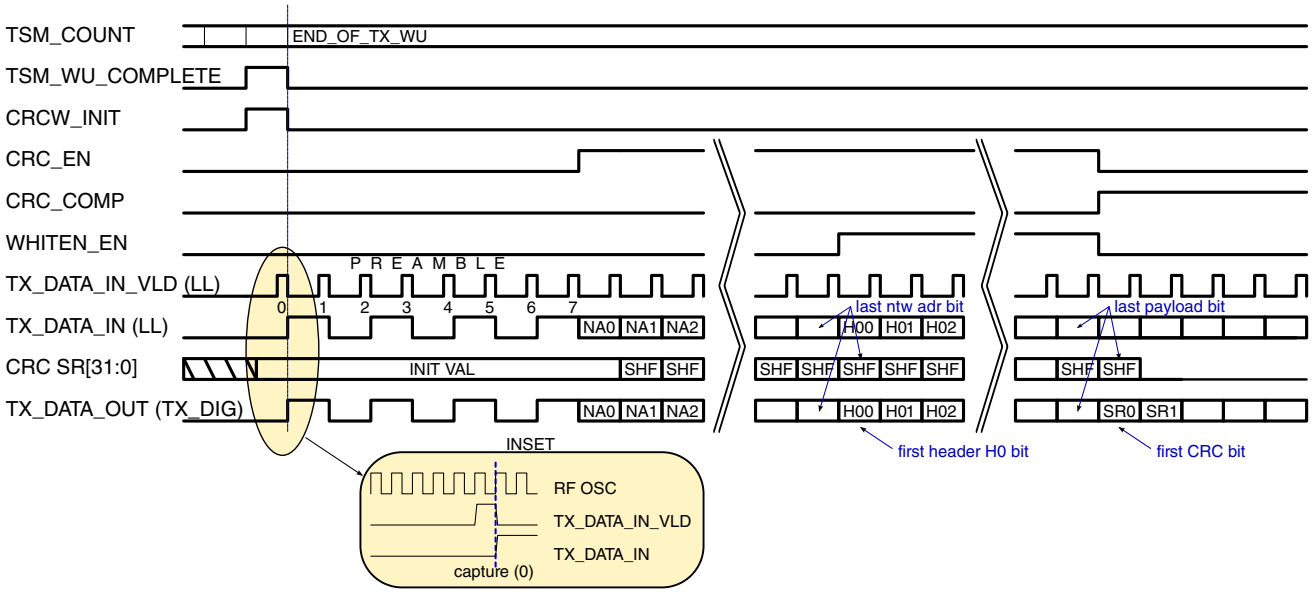



**Figure 64-181. BIT STREAM PROCESSING CRC AND WHITENING TX TIMING**

The bitstream depicted in the previous timing diagram, with CRC and whitening applied, is shifted out to the TX digital block, and transmitted over the air. A receiving device, will receive and process this same bitstream, by applying de-whitening, and verifying CRC on the de-whitened bitstream. The CRC and whitening parameters (registers) would be programmed to the receiving device, identically to how they are programmed for the transmitting device, specifically:

CRC_START_BYTE=0

WHITEN_START=1

WHITEN_END=0

A similar timing diagram depicting this RX bitstream processing timing, is shown below.

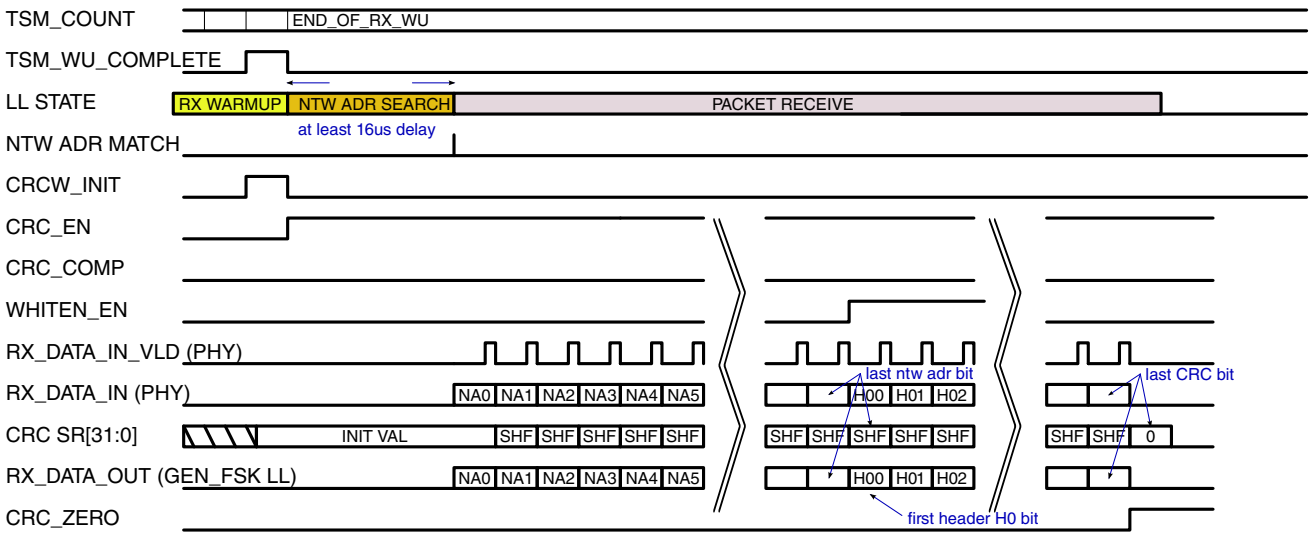



**Figure 64-182. BIT STREAM PROCESSING CRC AND WHITENING RX TIMING**

The GENERIC_FSK Link Layer controller provides a special whitening option, that causes the whitening LFSR to be re-initialized at the start of the PAYLOAD. The register bit WHITEN_PAYLOAD_REINIT, when 1, will result in the LFSR being re-initialized to its programmed initial state, determined by WHITEN_SEED[8:0], coincident with the first bit of the first byte of PAYLOAD. This applies to both TX and RX. When WHITEN_PAYLOAD_REINIT=0, there is no such re-initialization. This feature is only relevant when whitening/de-whitening is programmed to start before the PAYLOAD, i.e., WHITEN_START=1 or WHITEN_START=2.

The GENERIC_FSK Link Layer controller provides a number of other configurability options, which affect the internal operation of the CRC engine, and not the Link Layer controller itself. A listing of these registers which affect CRC internal operation, appears in the table below. For more details on how these registers affect CRC operation, see the CRC/Whitener Chapter.

| CRC-related Register | Brief Description |
|---|---|
| CRC_POLY[31:0] | CRC Polynomial<br>1: XOR exists in the bit's feedback path<br>0: no XOR in the bit's feedback path |
| CRC_SEED[31:0] | CRC Initialization Value |
| CRC_BYTE_ORD | 0: LS Byte First<br>1: MS Byte First |
| CRC_REF_IN | 0: do not manipulate input data stream<br>1: reflect each byte in the input stream bitwise |
| CRC_REF_OUT | 0: do not manipulate CRC result |

*Table continues on the next page...*

| CRC-related Register | Brief Description |
|---|---|
| | 1: CRC result is to be reflected bitwise (operated on entire word) |
| CRC_XOR_OUT[31:0] | XOR mask for CRC result (for no mask, should be 0) |

Similarly, there are a number of configurable whitening options provided by the Link Layer controller, which affect the internal operation of the whitening engine, and not the Link Layer controller itself. A listing of these registers which affect whitener internal operation, appears in the table below. For more details on how these registers affect whitener operation, see the CRC/Whitener Chapter.

| Whitener-related Register | Brief Description |
|---|---|
| WHITEN_ POLY[8:0] | Polynomial Value for Whitening/De-whitening |
| WHITEN_POLY_TYPE | Whiten polynomial type. A Fibonacci type LFSR is used with the whiten polynomial if this bit is asserted. Otherwise, a Galois type LFSR is used. |
| WHITEN_SEED[31:0] | Initialization Value for Whitening/De-whitening |
| WHITEN_SIZE[3:0] | Whitener Length |
| WHITEN_REF_IN | The input data stream is reflected, bit-wise, per byte, if this bit is asserted. Bit 7 becomes bit 0, bit 6 becomes bit 1, etc. This bit will only cause the reflection of the payload data bits as they are used in the whiten calculation and will not cause any change in the output bit order. |

## BITRATE

The GENERIC_FSK Link Layer controller provides multiple bitrate options. The register BITRATE[1:0] selects the bitrate, according to the following table.

| BITRATE[1:0] Register | BITRATE |
|---|---|
| 0 | 1 Mbit/sec |
| 1 | 500 Kbit/sec |
| 2 | 250 Kbit/sec |
| 3 | 2 Mbit/sec |

For transmission, BITRATE affects the rate at which the Link Layer controller shifts out bits, through the CRC/Whitener, and ultimately to the TX Digital Block. For reception, BITRATE affects the rate at which the PHY shifts out bits, through the CRC/Whitener, and ultimately to the Link Layer controller, which receives and processes the bitstream. The Link Layer is the originator of the packet bitstream for transmission; the PHY is the originator of the bitstream for reception. The Link Layer controller determines the bitrate by sending the **bitrate[1:0]** signal to the CGM (Clock Generation Module). The CGM decodes this signal and adjusts the rate at which pulses appear on **cg_vbr_en** to track the

programmed BITRATE. (Note: the 'vbr' in **cg_vbr_en** is an acronym for ‘variable bit rate’.) The CGM divides down the reference oscillator frequency to generate the frequency of the desired BITRATE, and then sends out pulses on cg_vbr_en which are one reference oscillator clock wide, at that rate. The **cg_vbr_en** shall be used as a gating signal (clock gating and/or data gating) inside the blocks which originate the bitstream (Link Layer, PHY), and those that consume it (CRC/Whitener, TX digital). The timing on cg_vbr_en manifests itself on the respective *_data_in_vld and *_data_out_vld signals to and from the 4 blocks that operate on the bitstream. This way, all 4 blocks operate in unison at all times at a single, desired bitrate.

The following diagram depicts how bitrate is controlled by the Link Layer and CGM, and how the respective blocks involved in bitstream processing stay in sync using **cg_vbr_en**.

**BITSTREAM PROCESSING WITH VARIABLE BIT RATE**

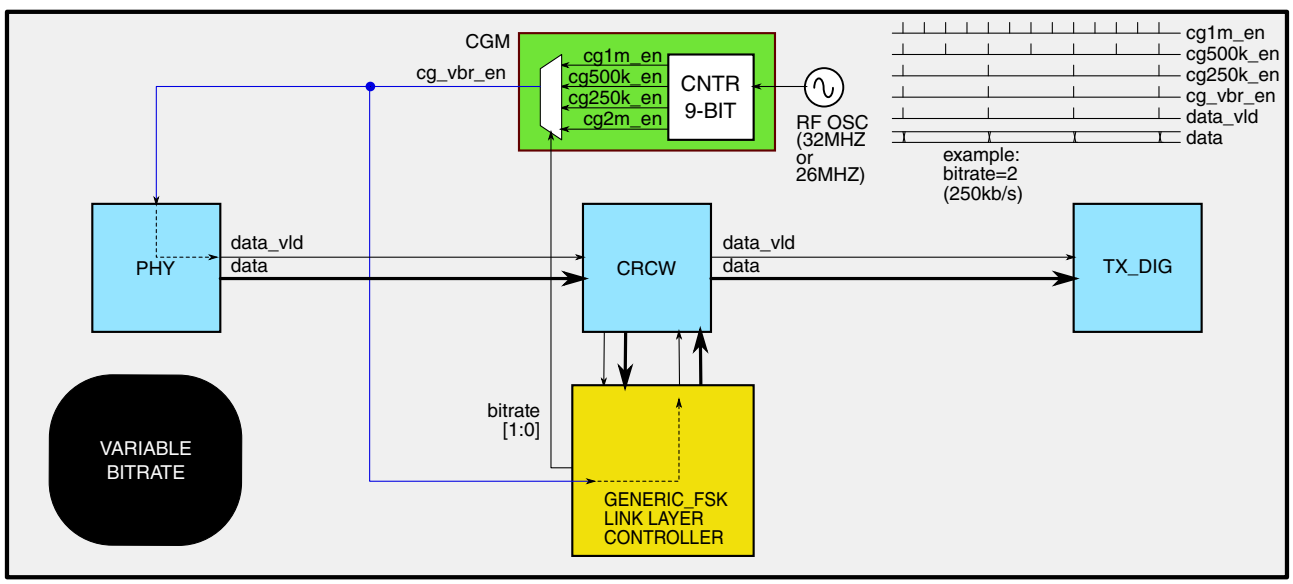



**Figure 64-183. BITSTREAM PROCESSING WITH VARIABLE BIT RATE**

## MANCHESTER ENCODING AND DECODING

The GENERIC_FSK Link Layer controller allows the packet bitstream to be Manchester-encoded for transmission, and decoded for reception. The Manchester encoding used by the controller conforms to IEEE 802.3, which is shown in the following diagram.

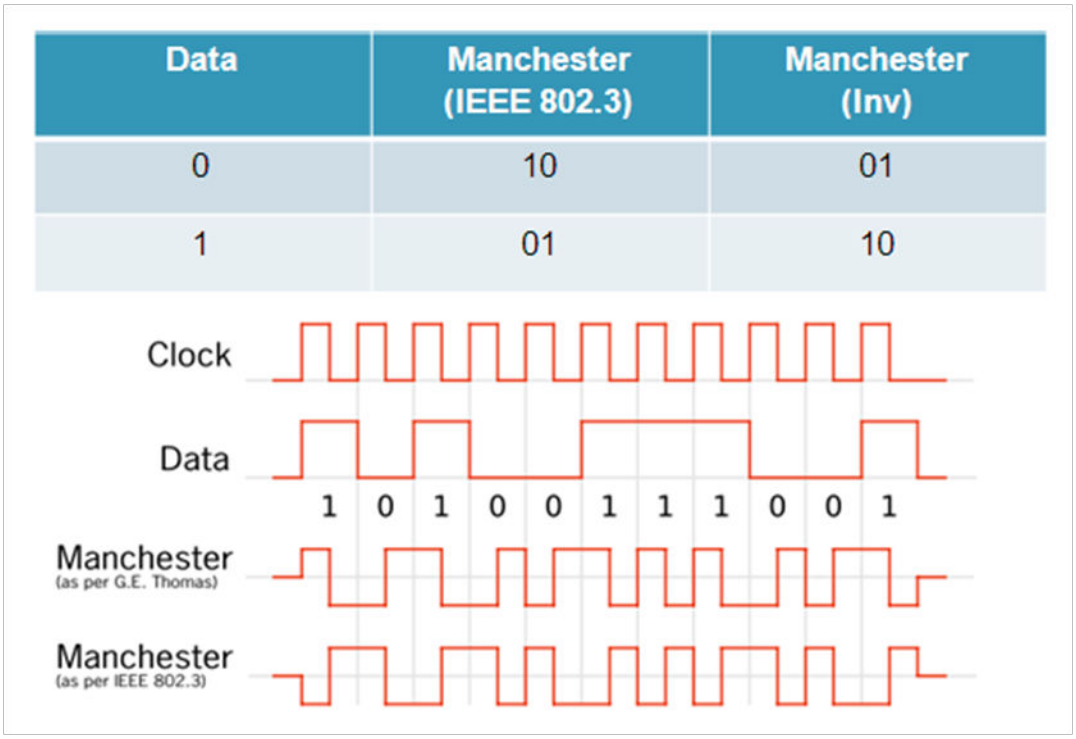



**Figure 64-184. Manchester Encoding**

Three programmable registers enable and control Manchester encoding and decoding, as described in the following table.

| | TX | RX |
|---|---|---|
| MANCHESTER_EN | 1: the entire packet bitstream, including CRC, is Manchester-encoded as it is transmitted to the TX digital block. CRC operates on the original, non-encoded bitstream.<br>0: No encoding applied | 1: the entire packet bitstream, including CRC, is Manchester-decoded, and the output of the decoder is transmitted simultaneous to the CRC engine and the Link Layer.<br>0: No decoding applied |
| MANCHESTER_INV | 1: Encoding is per 802.3, but with the coding signal inverted<br>0: Encoding is as per 802.3 | 1: Decoding is per 802.3, but with the coding signal inverted<br>0: Decoding is as per 802.3 |
| MANCHESTER_START | 1: Begin encoding at start-of-header<br>0: Begin encoding at start-of-payload | 1: Begin decoding at start-of-header<br>0: Begin decoding at start-of-payload |

As the encoding diagram implies, the post-encoded process bitstream has a bitrate of twice that of the pre-encoded data. The over-the-air bitrate must, of course, be constant. Thus, for transmission, the source of the bitstream data, i.e., the GENERIC_FSK Link Layer controller, must halve its bitrate whenever Manchester encoding is engaged. Since the CRC engine is also upstream of the encoder, it will also see the half bitrate. The output of the Manchester encoder, which feeds the TX digital block, will see the full bitrate. For reception, the source of the bitstream data, i.e, the PHY, will see the full

bitrate. After the PHY feeds the Manchester decoder with the full-rate data, the blocks downstream of the decoder, i.e., the Link Layer controller and the CRC engine, will need to operate at half bitrate. When Manchester encoding is used, the over-the-air bitrate, that is, the bitrate programmed via the BITRATE register, is restricted to either 1Mbit/sec (BITRATE=0), or 500Kbit/sec (BITRATE=1); other BITRATE options are not supported with Manchester. The following table summarizes the supported over-the-air bitrates, and specifies the effective bit rates seen by the 4 blocks involved in bitstream processing while Manchester is engaged.

| | PHY | GENERIC_FSK Link Layer | CRC LFSR | TX Digital |
|---|---|---|---|---|
| BITRATE=0 (ota bitrate = 1 Mbit/sec) | 1 Mbit/sec | 500 Kbit/sec | 500 Kbit/sec | 1 Mbit/sec |
| BITRATE=1 (ota bitrate = 500Kbit/sec) | 500 Kbit/sec | 250 Kbit/sec | 250 Kbit/sec | 500 Kbit/sec |

Because the Manchester encoding and decoding is not performed over the entire packet, e.g., the Network Address is never encoded, the Link Layer controller must switch its bitrate *dynamically*, reducing its own bitrate by half whenever Manchester is engaged, and restoring full bitrate when Manchester is disengaged. Since the PHY and TX digital must process bits at the full rate at all times, the bitstream-pacing signal, **cg_vbr_en**, must be split, such that the PHY and TX digital get the full rate gating signal, and the Link Layer and CRC get the variable rate gating signal. The following diagram describes how the CGM generates a special half-rate rendition of **cg_vbr_en** when Manchester is engaged (**encode_en**=1), and feeds this to the Link Layer (and indirectly, the CRC engine). The diagram assumes the default setting of BITRATE=0 (over-the-air rate is 1Mb/sec).

## BITSTREAM PROCESSING WITH MANCHESTER ENCODING

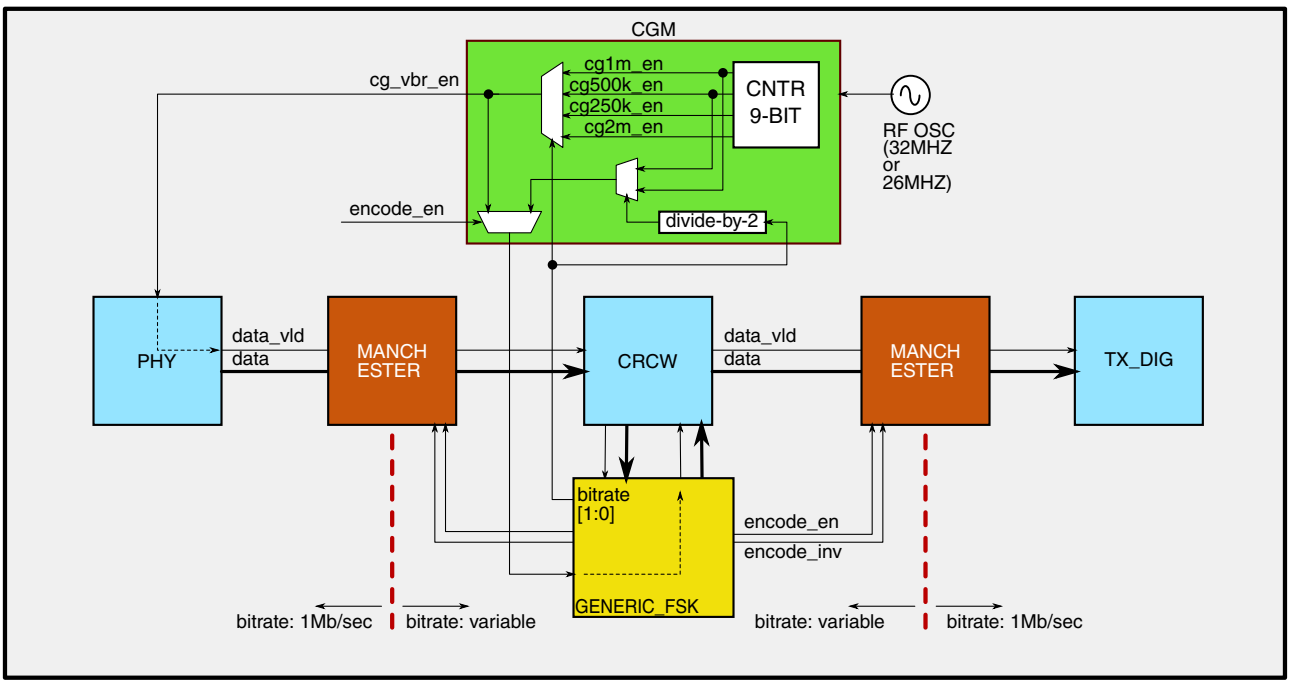



**Figure 64-185. BITSTREAM PROCESSING WITH MANCHESTER ENCODING**

A timing diagram depicting an example of bitstream processing, including Manchester encoding, for a packet transmission, is included below. The diagram shows the mid-packet switching of the encoding-enable signal, encode_en, and the halving of the bitrate as seen by the Link Layer controller. This example uses MANCHESTER_START=0, to initiate encoding at the start-of-payload.

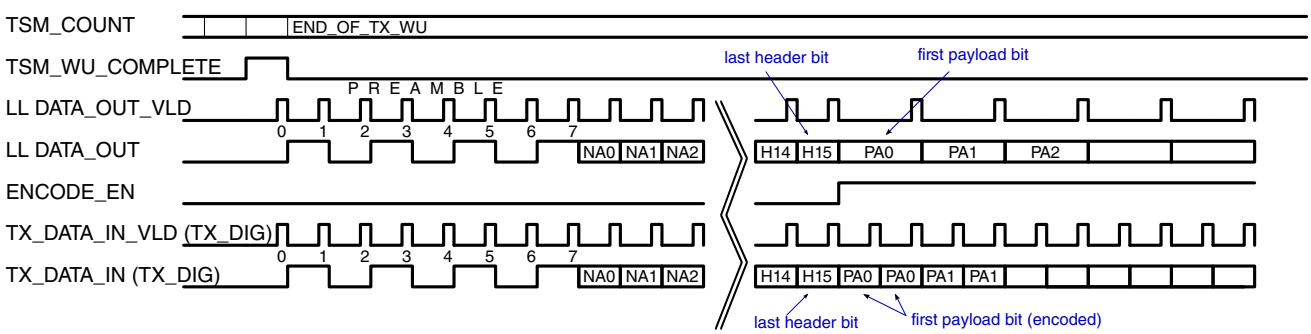



**Figure 64-186. BIT STREAM PROCESSING (WITH MANCHESTER) TX TIMING -- 1MBIT/SEC OTA**

An example of bitstream processing, including Manchester encoding, for a packet reception, is included below. The diagram shows the mid-packet switching of the decoding-enable signal, encode_en, and the halving of the bitrate as seen by the Link Layer controller. This example uses MANCHESTER_START=0, to initiate encoding at the start-of-payload.

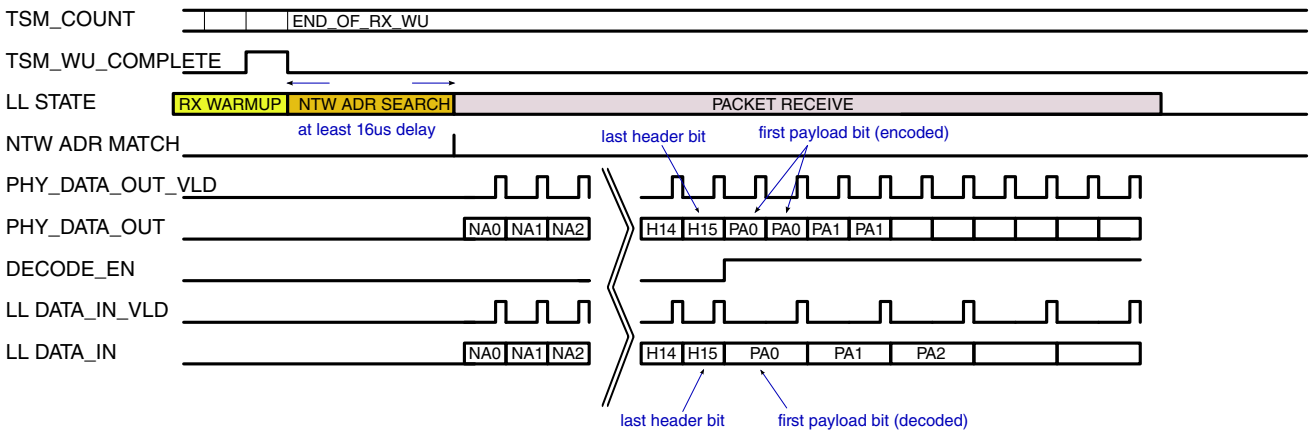



**Figure 64-187. BIT STREAM PROCESSING (WITH MANCHESTER) RX TIMING -- 1MBIT/SEC OTA**

**NOTE**

Whitening and Manchester encoding are mutually exclusive, and must not be enabled simultaneously. This is a software restriction on the setting of the register bits WHITEN_START and MANCHESTER_EN. Both may not be simultaneously non-zero. Failure to heed this restriction may lead to indeterminate behavior.

### 64.5.3.3.3 Fixed Packet Length

The GENERIC_FSK Link Layer controller expects a stream of consecutive bits specifying the length of the packet, to be transmitted and received over-the-air, as part of the packet bitstream, with the length specification typically encoded into the packet header, as shown below.

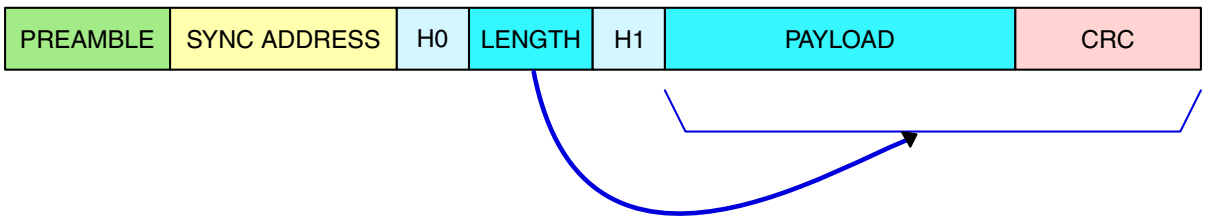



**Figure 64-188. PACKET STRUCTURE WITH LENGTH FIELD IN HEADER**

Some protocols don't include any length instructions in the packet header, as shown below.

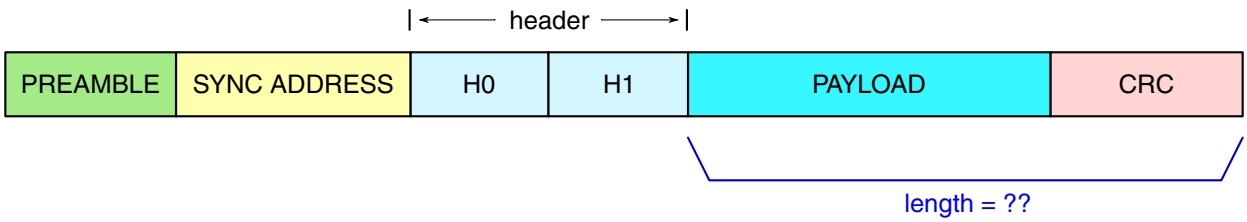



**Figure 64-189. PACKET STRUCTURE WITH NO LENGTH FIELD IN HEADER**

And, some protocols may not include a header at all.

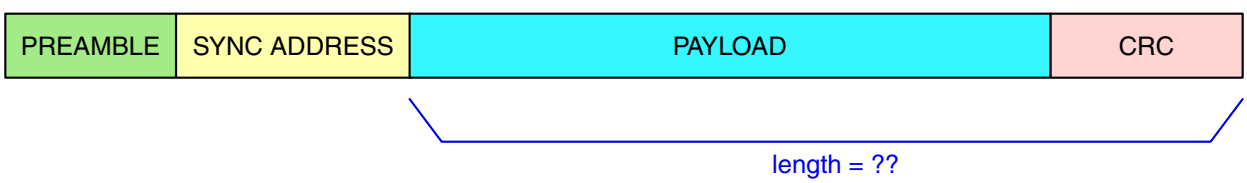



**Figure 64-190. PACKET STRUCTURE WITH NO HEADER**

In the aforementioned cases, another method of specifying length to the GENERIC_FSK Link Layer hardware must be employed. In both cases, if the length is not specified as part of the over-the-air bitstream, then GENERIC_FSK software must know the length in advance of the TX or RX operation, or know how to calculate it. So GENERIC_FSK software will provide this information to the hardware, in advance of the TX or RX operation, by programming the following registers:

- PACKET_CFG[LENGTH_SZ] = 0 : instructs the hardware that there is no LENGTH info in the packet header, and thus forces the hardware to use LENGTH_ADJ for packet length
- PACKET_CFG[LENGTH_ADJ] = ??? : program the desired (known) length of PAYLOAD + CRC

- PACKET_CFG[LENGTH_ADJ_UNSIGNED] = 1 : allows programmed length up to 255 octets
- CRC_CFG[CRC_SZ] = ??? : program the desired number of CRC octets

As an example, a packet is to be transmitted or received using the GENERIC_FSK Link Layer hardware. The packet has no header, but the length of the payload is known in advance to be 57 octets, and the number of CRC octets is 3. In this case, LENGTH_ADJ is programmed to 60 (57 + 3) and CRC_SZ is programmed to 3, as shown below.

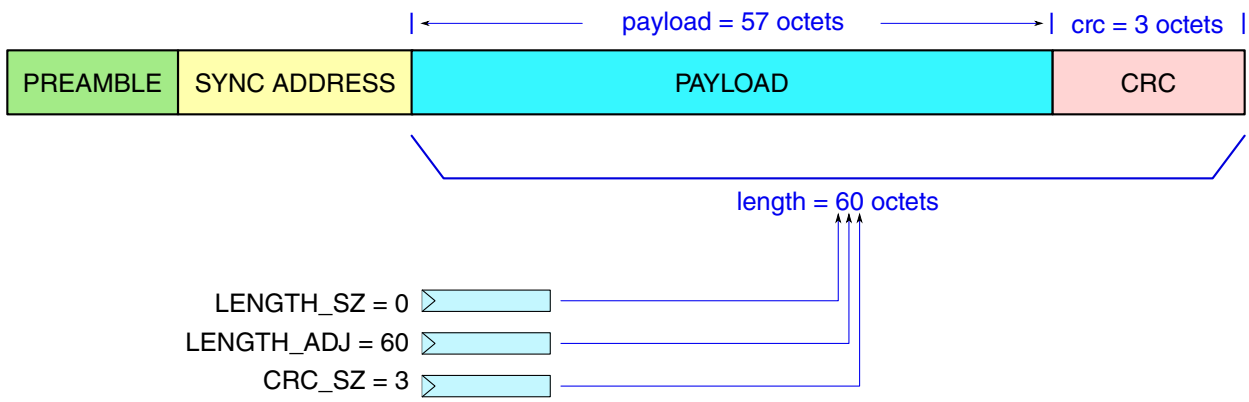



**Figure 64-191. PACKET STRUCTURE WITH NO HEADER AND FIXED PACKET LENGTH**

### 64.5.3.3.4 Packet Storage

The GENERIC_FSK Link Layer has access to a Packet Buffer (also known as Packet RAM), to store data to be transmitted, and to receive incoming packet data. The GENERIC_FSK Link Layer software prepares data to be transmitted, by loading the octets, in order, into the Packet Buffer. After reception, the octets received over the air are stored into the Packet Buffer, in the order they are received. The Packet Buffer contains separate spaces for TX and RX data, so incoming receive octets don't overwrite previously-loaded content intended for transmission. The Packet Buffer is large enough to accommodate the longest GENERIC_FSK packet, 2047 octets, plus header and CRC octets, for either transmit or receive.

The GENERIC_FSK Packet Buffer is contained within the multi-protocol Packet RAM. When operating in GENERIC_FSK mode, the entire Packet RAM is allocated to the GENERIC_FSK packet buffer. Because of the potentially large size of the GENERIC_FSK packets, and taking into account the high likelihood of asymmetry between the required lengths of packets for TX and RX, the 2 normally-separated RAM components of the Packet RAM are adjoined sequentially to form a double-depth (1088-entry) single storage space, still 16-bits wide. In addition, the consolidated RAM can be dedicated entirely to TX, entirely to RX, or split between TX and RX. The partitioning of the consolidated RAM between TX and RX is controlled by a configurable TX/RX

Partition Point. A Partition Point register, PB_PARTITION[10:0], has been provided. The Register can be programmed with any value between 0 and 1087. The consolidated RAM is partitioned such that the TX buffer resides before the Partition Point (RAM entries 0 -> (Partition_Point-1)), and the RX buffer resides after it (RAM entries Partition_Point -> 1087). Programming the Partition Point register to 0 dedicates the entire consolidated RAM to RX; programming the Partition Point register to 1088 dedicates the entire consolidated RAM to TX; programming the Partition Point register to any value between 1 and 1087 yields a split TX/RX buffer. Separate TX and RX buffers within the consolidated RAM means that incoming RX packets don't overwrite TX packet data. Because the Packet RAM is 16 bits wide, the GENERIC_FSK Link Layer controller hardware will address the consolidated RAM in an interleaved fashion as it stores octets to, or fetches octets from, the RAM-based packet buffer.

The partitioning of the consolidated RAM between transmit and receive sections for GENERIC_FSK is shown in the diagram below:

1. RAM0 and RAM1 abutted sequentially to form 1 contiguous space from Entry #0 - #1087
2. A Programmable Partition Point is defined: TX data before the partition, RX data after
3. The partition point can be programmed to any value from Entry #0 - #1087
4. Full 2 KB packet can be stored, for TX or RX
5. TX and RX packets are separated (RX does not overwrite TX)
6. Host access restricted to 16-bit

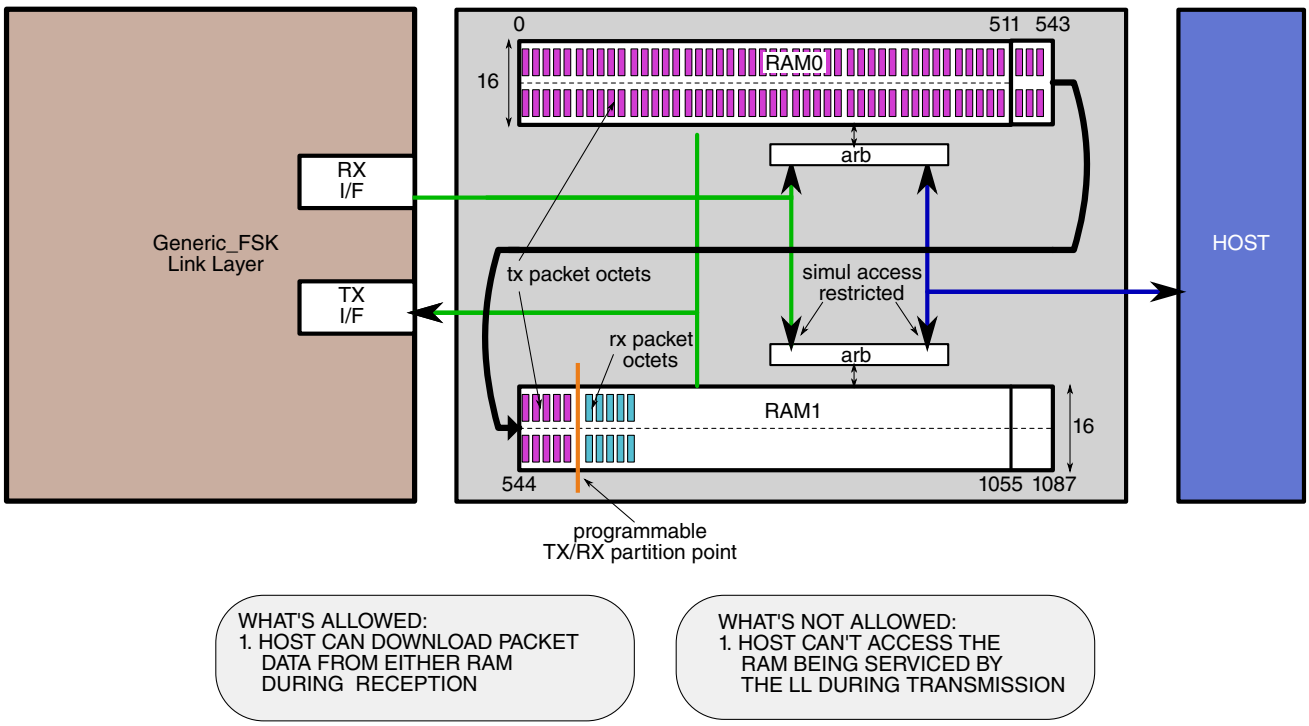



**Figure 64-192. RAM ALLOCATIONS FOR GENERIC_FSK**

During packet reception, the MCU is allowed to access the RX packet buffer of the consolidated RAM, in order to download the packet while it's being received, if so desired. In cases where both the MCU and the GENERIC_FSK Link Layer controller attempt to simultaneously access the RAM during reception, internal hardware arbitration is provided to delay the MCU access until the GENERIC_FSK Link Layer completes its access to the RAM. Link Layer accesses complete in a single reference oscillator clock period, so one and only 1 wait state shall be inserted on the IPS bus whenever access attempts coincide.

During packet transmission, MCU access to the consolidated RAM is not allowed. The GENERIC_FSK software must program the TX buffer before commanding a TX operation, and must not access the RAM during the transmission.

As mentioned previously, incoming receive data lands in the RX section of the Packet Buffer. Nominally, a newly received packet will overwrite the contents of the previously received packet (the TX section of the Packet Buffer remains intact). Software can inhibit receive-packet overwriting of the RX section of the Packet Buffer contents by setting the PB_PROTECT bit in the PACKET_RAM_CTRL register, in XCVR address space. When this bit is set, incoming receive data will be blocked from overwriting the existing contents of the RX section of the Packet Buffer. This will block RX packet overwriting, but will not inhibit TX content loading of the TX section of the Packet Buffer.

In the 2.4GHz Radio, the GENERIC_FSK Link Layer is the controlling Link Layer whenever the XCVR_CTRL[PROTOCOL] register is set to 8, 9, or 10. GENERIC_FSK software must not attempt to access the Packet RAM unless XCVR_CTRL[PROTOCOL] is set to 8, 9, or 10. When XCVR_CTRL[PROTOCOL] is set to one of the following: {0, 1, 2, 3, 4, or 5}, another Protocol Engine besides GENERIC_FSK, has exclusive access to the Packet RAM. That also means that the IPS bus assigned to that Protocol Engine, is the only IPS bus allowed to access the Packet RAM. If the GENERIC_FSK Link Layer software attempts to access the Packet RAM via the GENERIC_FSK IPS bus, when XCVR_CTRL[PROTOCOL] is set to one of the above settings, the access attempt will fail, as described below:

1. RAM contents shall not be altered on a GENERIC_FSK write attempt
2. Readback data on a GENERIC_FSK read attempt shall be indeterminate
3. The **ips_xfr_err** shall be asserted on the GENERIC_FSK IPS bus

If XCVR_CTRL[PROTOCOL] is set to neither {8, 9, or 10}, nor {0, 1, 2, 3, 4, or 5}, this means no Protocol Engine has exclusive access to the Packet RAM. GENERIC_FSK software should not attempt to access the Packet RAM under such conditions; results are indeterminate.

#### 64.5.3.3.5 Timebase

The GENERIC_FSK Link Layer includes a dedicated, 24-bit multi-purpose Event Timer (EVENT_TMR), that is updated at a 1 MHz rate. The EVENT_TMR maintains the timebase for all GENERIC_FSK-related activities, except during intervals of Deep Sleep Mode. The clock source for the EVENT_TMR is a high-precision, crystal-referenced RF Oscillator, which can be either 32MHz or 26MHz. The RF Oscillator is divided down to 1MHz for GENERIC_FSK in the Clocks Generation Module (CGM). To reduce power consumption to a minimum, most of the GENERIC_FSK Link Layer Controller runs off this 1MHz clock. The exceptions are the Register IP bus interface, Command Decoder, and Interrupt Control Unit, parts of which must run at the IP bus frequency.

The 24-bit EVENT_TMR running at 1 MHz rolls over (wraps around) approximately every 16.8 seconds.

The EVENT_TMR count can be read at any time, via the register of the same name. A 32-bit read is recommended to ensure coherency of the 3 bytes that make up the counter. No synchronization to the IP bus is required.

If the EVENT_TMR needs to be updated (changed), this can be done by writing the new, desired EVENT_TMR value to the register of the same name, and setting the EVENT_TMR_LD bit to 1. The update to EVENT_TMR will take effect at the next

1MHz clock edge. The EVENT_TMR[23:0] and EVENT_TMR_LD are all part of the same 32-bit register, and loading of the EVENT_TMR using this method should be performed as an atomic, 32-bit write.

If it is desired to increment the EVENT_TMR by a known amount, this can be done by writing the desired increment value to the EVENT_TMR register, and setting the EVENT_TMR_ADD bit to 1. The increment value is 24-bit signed, thus enabling a decrement function as well. The update to EVENT_TMR will take effect immediately. The EVENT_TMR_ADD avoids the necessity to perform an read-modify-write operation to the EVENT_TMR to adjust its value; this facilitates accurate restoration of the EVENT_TMR state after a period of Deep Sleep Mode. The EVENT_TMR[23:0] and EVENT_TMR_ADD are all part of the same 32-bit register, and incrementing of the EVENT_TMR using this method should be performed as an atomic, 32-bit write.

Two 24-bit timer-compare registers are provided, T1_CMP and T2_CMP.

GENERIC_FSK software may load T1_CMP with a future EVENT_TMR count value. When EVENT_TMR reaches the T1_CMP value, either (or both) of the following events can be triggered:

1. T1_IRQ (Interrupt)
2. Sequence Command Trigger (can automatically launch or stop a transceiver operation)

Similarly, GENERIC_FSK software may load T2_CMP with a future EVENT_TMR count value. When EVENT_TMR reaches the T2_CMP value, either (or both) of the following events can be triggered:

1. T2_IRQ(Interrupt)
2. Sequence Command Trigger (can automatically launch or stop a transceiver operation)

For either type of T1-triggered events to be enabled, i.e., interrupt or sequence-launch, set T1_CMP_EN=1 after loading T1_CMP with the desired value. If T1_CMP_EN=1, the T1_IRQ interrupt status bit will always become set on a T1 timer match. (Whether or not T1_IRQ generates an GENERIC_FSK Interrupt to the MCU depends on the settings of T1_IRQ_EN and GENERIC_FSK_IRQ_EN, see section: Interrupts). If T1_CMP_EN=0, neither an interrupt nor a sequence command can be triggered by a T1 match.

For either T2-triggered events to be enabled, i.e., interrupt or sequence launch, set T2_CMP_EN=1 after loading T2_CMP with the desired value. If T2_CMP_EN=1, the T2_IRQ interrupt status bit will always become set on a T2 timer match. (Whether or not T2_IRQ generates a GENERIC_FSK Interrupt to the MCU depends on the settings of T2_IRQ_EN and GENERIC_FSK_IRQ_EN, see Section: Interrupts). If T2_CMP_EN=0, neither an interrupt nor a sequence command can be triggered by a T2 match.

See Section Interrupts for a description of how the Link Layer Controller manages the T1 and T2 interrupts, and how software can clear them.

See Section Sequence Commands and Status for a description on how to launch and stop sequences on T1 and T2 timer matches.

### 64.5.3.3.6 Sequence Commands and Status

The GENERIC_FSK Link Layer Controller provides a command set of 12 transceiver commands, which can be used to launch and stop sequences. GENERIC_FSK software can issue sequence commands by writing to the SEQCMD[3:0] field of the XCVR_CTRL register. The SEQCMD field always reads back what was previously written by software, enabling software to quickly recall the last command written. The sequence commands are as listed in the table below:

| SEQCMD[3:0] | COMMAND | DESCRIPTION |
|---|---|---|
| 0x0 | NULL | No Action |
| 0x1 | TX_START_NOW | TX Start Now |
| 0x2 | TX_START_T1 | TX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP) |
| 0x3 | TX_START_T2 | TX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP) |
| 0x4 | TX_CANCEL | TX Cancel -- Cancels pending TX events but do not abort TX-in-progress[1] |
| 0x5 | RX_START_NOW | RX Start Now |
| 0x6 | RX_START_T1 | RX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP) |
| 0x7 | RX_START_T2 | RX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP) |
| 0x8 | RX_STOP_T1 | RX Stop @ T1 Timer Compare Match (EVENT_TMR = T1_CMP) |
| 0x9 | RX_STOP_T2 | RX Stop @ T2 Timer Compare Match (EVENT_TMR = T2_CMP) |
| 0xA | RX_CANCEL | RX Cancel -- Cancels pending RX events but do not abort RX-in-progress[2] |
| 0xB | ABORT | Abort All - Cancels all pending events and abort any sequence-in-progress |

1. TX-in-progress implies TX Warmup is complete, and packet transmission is underway (i.e., command decoder state = IN_TX)
2. RX-in-progress implies RX Warmup is complete, and Network Address search or packet reception is underway (i.e., command decoder state = IN_RX)

As described above, sequences can be launched immediately (actually on the next 1MHz clock edge) by issuing one of the following commands:

- TX_START_NOW
- RX_START_NOW

Alternatively, sequences can be scheduled to launch (or stop) in the future on a T1 timer compare, by loading T1_CMP with the desired launch time (relative to the EVENT_TMR), setting T1_CMP_EN=1, and then issuing one of the following commands:

- TX_START_T1
- RX_START_T1
- RX_STOP_T1

Similarly, sequences can be scheduled to launch (or stop) in the future on a T2 timer compare, by loading T2_CMP with the desired launch time (relative to the EVENT_TMR), setting T2_CMP_EN=1, and then issuing one of the following commands:

- TX_START_T2
- RX_START_T2
- RX_STOP_T2

Any transceiver sequence that has been launched, either immediately (by way of the *_NOW commands), or in the future (by way of the *_T1 or *_T2 commands), can be terminated immediately by issuing the ABORT command. This is true whether or not the sequence has actually begun, or is merely pending because the T1- (or T2-) match has not yet occurred. In either case, this is called a software abort. The ABORT command will terminate any active sequence and cancel any pending events.

Regarding the issuance of sequence commands, both immediate and timer-triggered, GENERIC_FSK software should be mindful of the following Guidelines and Restrictions:

| PROGRAMING GUIDELINES AND RESTRICTIONS | |
|---|---|
| 1 | T1 can only be assigned to 1 function at a time {TX_START, RX_START, RX_STOP}.<br>Once assigned to a function, attempts to change the assignment will be ignored |
| 2 | T2 can only be assigned to 1 function at a time {TX_START, RX_START, RX_STOP}.<br>Once assigned to a function, attempts to change the assignment will be ignored |
| 3 | To change a T1- or T2- assignment, do one of the following first:<br>**a.** issue an ABORT command<br>**b.** issue an TX_CANCEL command if T1 (or T2) is already assigned to TX_START, and the sequence has not yet started<br>**c.** issue an RX_CANCEL command if T1 (or T2) is already assigned to RX_START or RX_STOP, and the sequence has not yet started |
| 4 | T1 and T2 should not be assigned to the same function (results are indeterminate) |
| 5 | If T1 and T2 are both assigned to functions, then T1_CMP shall not be equal to T2_CMP (results are indeterminate) |

*Table continues on the next page...*

| PROGRAMING GUIDELINES AND RESTRICTIONS | |
|---|---|
| 6 | If any TX or RX sequence is underway (e.g., TSM not idle), the ongoing sequence will not be interrupted or perturbed by any of the following:<br>**a.** a T1 timer-match with T1 assigned to a function. The sequence trigger will be ignored, but the associated "pending" status bit will be cleared.<br>**b.** a T2 timer-match with T2 assigned to a function. The sequence trigger will be ignored, but the associated "pending" status bit will be cleared.<br>**c.** a TX_START_NOW or RX_START_NOW command. Both commands will be ignored |

To perform operations on an RF channel, the GENERIC_FSK Link Layer Controller interacts with the Transceiver Sequence Manager (TSM), which is responsible for the low-level signaling required to activate the individual RF blocks. To do this, the Link Layer Controller includes a Command Decoder. The Command Decoder translates sequence commands (e.g., TX_START_NOW, RX_STOP_T1) into TSM inputs required to start and stop sequences. Then, the TSM executes the warmup, TX, RX, and warmdown functions on the channel. The GENERIC_FSK Command Decoder sends 2 signals (mutually exclusive) to the TSM:

- generic_fsk_tx_en
- generic_fsk_rx_en

Upon receiving an assertion on either of these signals, TSM will launch into warmup on either a TX or RX sequence. When the TSM completes its warmup, it asserts the signal **tsm_wu_complete** back to the Command Decoder, which uses that signal to transition into full-on packet transmit or packet receive. When the packet transmission or reception is complete, the Command Decoder will deassert **generic_fsk_tx_en** (or **generic_fsk_rx_en**), and wait for the TSM to assert **tsm_idle**. At that point, the Command Decoder will return to its IDLE state.

To implement these functions, the Command Decoder includes a small (6-state) finite state machine. The state diagram is shown below, including the TSM interactions.

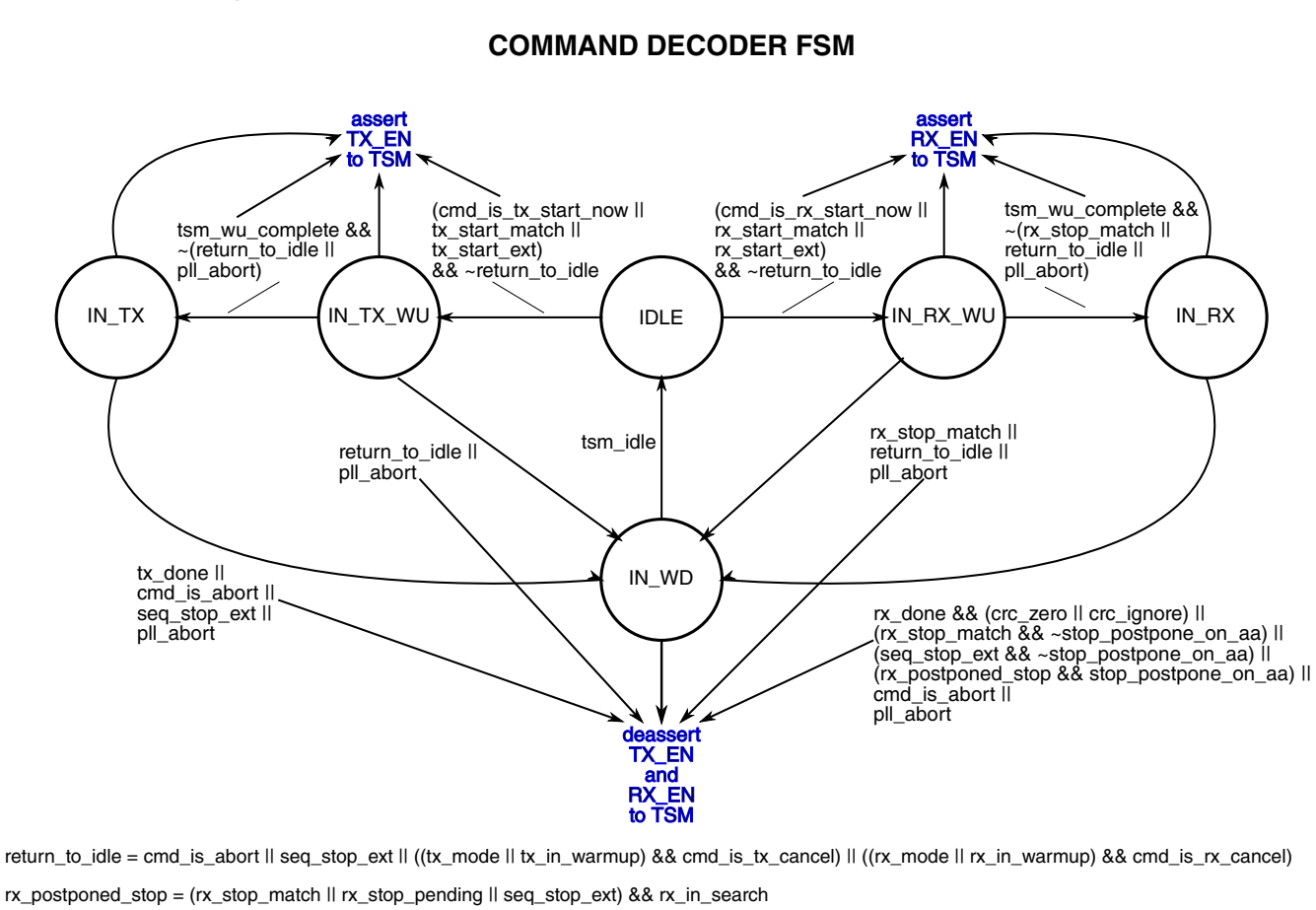



**Figure 64-193. GENERIC_FSK COMMAND DECODER FSM**

The state of the Command Decoder is carried by the state vector CMDDEC_CS[2:0]. The state can be monitored at all times by reading the CMDDEC_CS field of the XCVR_STATUS register, or via DTEST. The mapping of states to state vector is shown in the table below.

| CMDDEC_CS[2:0] | STATE NAME |
|---|---|
| 0b000 | IDLE |
| 0b001 | IN_TX_WU |
| 0b011 | IN_TX |
| 0b101 | IN_RX_WU |
| 0b111 | IN_RX |
| 0b010 | IN_WD |

The Command Decoder also maintains an set of 13 status bits, that make it easy for Link Layer software to track, not only state, but pending events; that is, timer-linked events scheduled for future triggering. These status bits reside in the XCVR_STS register. A description of these bits is provided in the table below.

| SEQSTS[12:0] | BIT | DESCRIPTION |
|---|---|---|
| TX_START_T1_PEND | [0] | TX Sequence will start @ next T1 Match |
| TX_START_T2_PEND | [1] | TX Sequence will start @ next T2 Match |
| TX_IN_WARMUP | [2] | TX Sequence in TSM Warmup |
| TX_IN_PROGRESS | [3] | TX Packet Transmission Currently Underway |
| TX_IN_WARMDN | [4] | TX Sequence in TSM Warmdown |
| RX_START_T1_PEND | [5] | RX Sequence will start @ next T1 Match |
| RX_START_T2_PEND | [6] | RX Sequence will start @ next T2 Match |
| RX_STOP_T1_PEND | [7] | RX Sequence will stop @ next T1 Match |
| RX_STOP_T2_PEND | [8] | RX Sequence will stop @ next T2 Match |
| RX_IN_WARMUP | [9] | RX Sequence in TSM Warmup |
| RX_IN_SEARCH | [10] | RX Sequence in Network Address Search |
| RX_IN_PROGRESS | [11] | RX Packet Reception Currently Underway |
| RX_IN_WARMDN | [12] | RX Sequence in TSM Warmdown |

### 64.5.3.3.7 Interrupts

Interrupt Status Bit Structure

The Generic FSK Link Layer Controller has 10 interrupt sources, each represented in the register map by an interrupt status bit. All interrupt status bits share a common, generic structure. If a particular interrupt source is enabled, a “TRIGGERING EVENT” for that interrupt source, will always set the status bit. A write-1-to-clear input from the IPS bus, will clear the status bit, but only if there is not a simultaneous triggering event. The triggering event prevails over a software clear attempt, if both events coincide. An integrated clock gate guarantees that the status bit only sees a clock when either a triggering event occurs, or a write-1-to-clear pulse arrives from the IPS bus. This reduces interrupt status bit power consumption to an absolute minimum. (Note: the TSM_IRQ does not have an associated status bit in the Link Layer Controller; its status bit resides in the Transceiver Sequence Manager, or TSM).

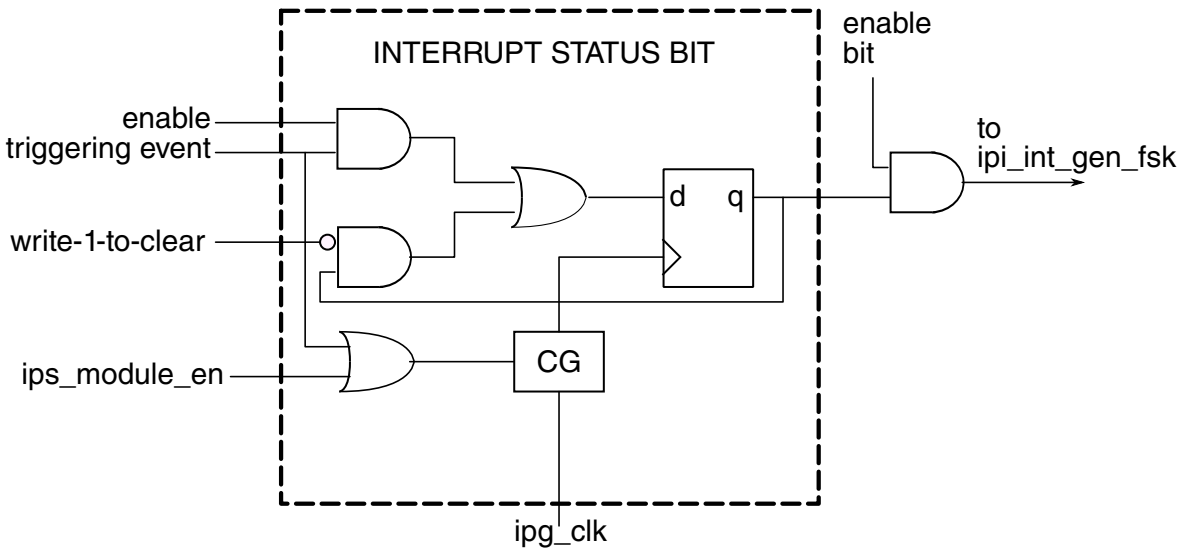



**Figure 64-194. Interrupt Status Bit**

Each GENERIC_FSK interrupt status bit, has an ENABLE bit associated with it. The name of the enable bit tracks the name of the interrupt source, with the string "_EN" appended at the end. Note that for any GENERIC_FSK interrupt source, if the triggering event occurs, the Interrupt Status Bit will be set regardless of the state of the corresponding ENABLE bit. If any of the 10 interrupts is to be ignored, software should clear the corresponding ENABLE bit, and apply the appropriate bit mask when reading the IRQ_CTRL register, to mask out the unwanted status bit.

GENERIC_FSK Interrupt Architecture

The 10 interrupt sources (status bits) are combined with their individual ENABLE bits, and then logically OR'ed together, in sum-of-products fashion, to generate single interrupt line to the MCU: **ipi_int_gen_fsk**. A global mask bit, GENERIC_FSK_IRQ_EN, can enable or disable **ipi_int_gen_fsk** altogether. There is no prioritization of interrupt sources; they have equal weight. The **ipi_int_gen_fsk** output is a level-sensitive indicator and will remain asserted until all interrupt status bits are cleared (or the corresponding ENABLE bits cleared). The following diagram depicts the GENERIC_FSK Interrupt Architecture.

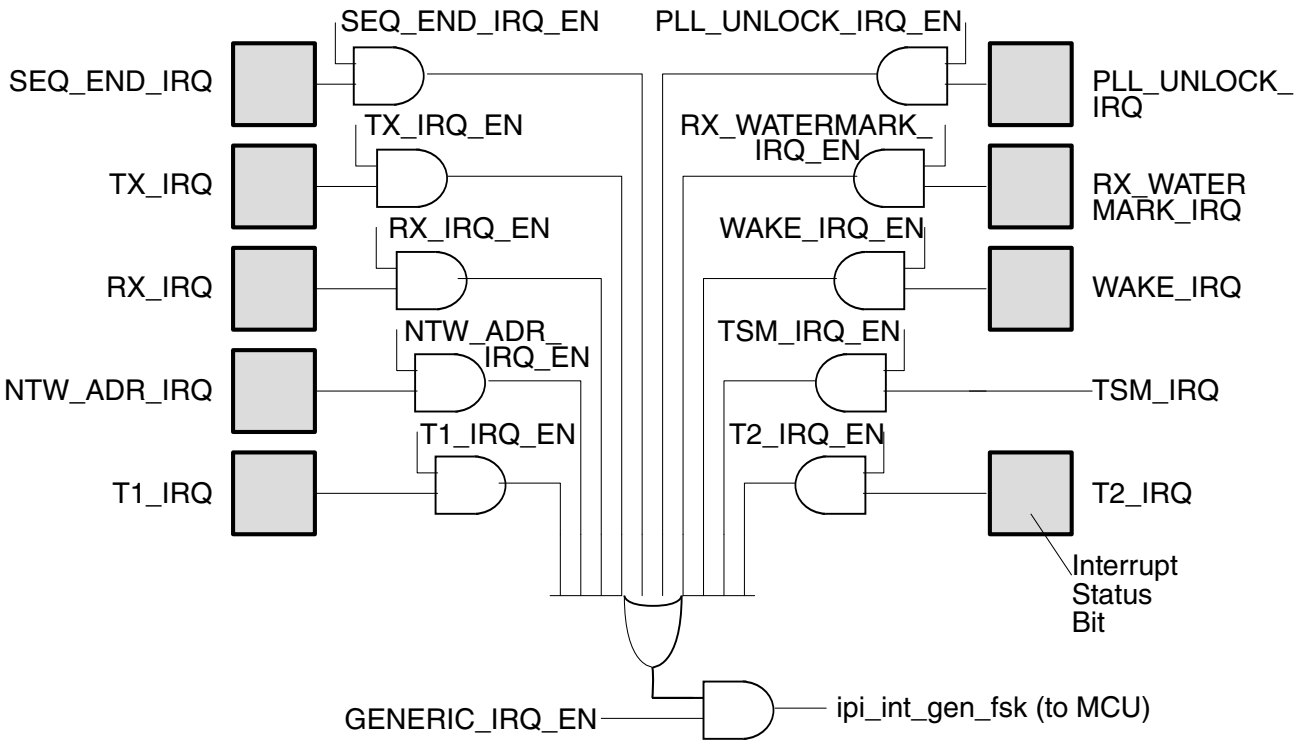



**Figure 64-195. GENERIC_FSK Interrupt Architecture**

## Clearing Interrupts

All interrupt status bit use a write-1-to-clear protocol. Writing a '1' to the interrupt status bit, in the IRQ_CTRL register, clears the offending interrupt. Writing a '0' to an interrupt status bit has no effect on the bit. Interrupt status bits are not affected by reads.

## GENERIC FSK Interrupt Sources

The following table describes the 10 GENERIC_FSK Interrupt Sources

| Interrupt | Description |
|---|---|
| T1_IRQ, T2_IRQ | The GENERIC_FSK Link Layer features a 24-bit Event Timer, which runs at a rate of 1MHz. The Link Layer has 2 Timer interrupts (T1_IRQ, T2_IRQ), each with its own 24-bit compare register (T1_CMP, T2_CMP), and each with its own compare enable (T1_CMP_EN, T2_CMP_EN). For each timer compare enable, if the bit is set, a match on the respective 24-bit compare value to the Event Timer will cause the corresponding interrupt status bit to become set. If the compare enable is not set, Event Timer matches won't cause the corresponding interrupt status bit to become set. |
| TX_IRQ | TX interrupt occurs at the end of a TX operation. |
| RX_IRQ | RX interrupt occurs at the end of a successful RX packet reception. A successful packet reception implies CRC was verified to be good, H0 matching passed, H1 matching passed, and received packet length did not violate pre-set |

*Table continues on the next page...*

| Interrupt | Description |
|---|---|
| | limits. Nominally, RX interrupts are not generated on packets which fail CRC check, or which failure the H0, H1 or LENGTH limits. A received packet with a failed CRC check or a H0, H1, or LENGTH violation, results in an RX recycle back to the Network Address Search state. To receive a Data Indication (RX_IRQ) on packets which fail CRC, set the bit CRC_IGNORE=1. H0 and H1 matching, and LENGTH limiting, are under SW control. |
| NTW_ADR_IRQ | A Network Address Match has occurred on an enabled Network Address. At NTW_ADR_IRQ, a timestamp is captured by transferring the current contents of the EVENT_TMR into the TIMESTAMP register. Software can determined which Network Address matched by querying the NTW_ADR_MCH[3:0] field of the NTW_ADR_CTRL register. |
| SEQ_END_IRQ | The Sequence End Interrupt (SEQ_END_IRQ), indicates that a transceiver operation (TX or RX) has completed, and the Command Decoder State Machine, and the Transceiver Sequence Manager (TSM), have returned to their respective IDLE states. A SEQ_END_IRQ will always occur at the end of a sequence, even if the sequence terminated abnormally (such as a Software Abort, or a PLL Unlock Abort). A SEQ_END_IRQ always occurs whenever the Command Decoder FSM transitions from any non-idle to IDLE state. When SEQ_END_IRQ occurs, software can be sure that the Command Decoder and TSM are in their idle state, and a new sequence can be programmed. |
| RX_WATERMARK_IRQ | RX Watermark interrupt will occur during packet reception, when the number of received bytes, as indicated by the read-only BYTE_COUNTER register, matches the contents of the RX_WATERMARK register. For the purpose of defining RX_WTR_MARK, the first byte received (BYTE_COUNTER = 0) is the first byte of Network Address; the second byte received (BYTE_COUNTER = 1) is the second byte of Network Address, etc. To cause an RX Watermark Interrupt to occur after a 2-byte Network Address has been received, set RX_WTR_MARK=1. By default, RX_WATERMARK is set to a value larger than the longest supported GENERIC_FSK packet length, so an RX_WATERMARK_IRQ won't be triggered under these conditions. |
| PLL_UNLOCK_IRQ | When an PLL unlock event occurs during an transceiver operation, and the transceiver has been configured to automatically abort sequences on PLL unlock events, the PLL_UNLOCK_IRQ status bit will become set. The Transceiver Sequence Manager (TSM) will begin monitoring for PLL unlock only after the PLL has been given sufficient time to achieve lock; A PLL unlock which occurs after the warmup period, can be enabled to cause a sequence abort. Individual enables are provided for TX and RX sequences. (See the TSM Chapter for more details). |
| WAKE_IRQ | **For Manual DSM:** This is WAKE_IRQ. A WAKE_IRQ will be triggered when the GENERIC_FSK Link Layer Controller has awoken from a Manual DSM (Deep Sleep Mode) cycle. WAKE_IRQ indicates that the RF Oscillator has been restarted, and the GENERIC_FSK EVENT_TMR has resumed counting. |

*Table continues on the next page...*

| Interrupt | Description |
|---|---|
| | **For Wake-On-Radio:**: This is WOR_IRQ. See Section Wake-On-Radio |
| TSM_IRQ | TSM_IRQ is a debug feature, enabling the Transceiver Sequence Manager (TSM) to generate an interrupt at any point in a TX or RX Warmup. TSM is a multipurpose hardware resource shared by all of the protocol engines in the SoC. Thus, TSM does not have its own interrupts; its interrupts are assigned to whichever link layer controller is currently executing an RF operation. TSM Interrupt Status bits reside in Transceiver (XCVR) Address Space, and can be cleared there. There is no intended mission-mode use for TSM_IRQ. See the TSM Chapter for more details. |

#### 64.5.3.3.8 Memory Mapping

In the SoC memory map, space has been allocated to GENERIC_FSK. Within the SoC, peripherals are allocated memory space in 4KB blocks, called IPS slots. GENERIC_FSK occupies one such slot. GENERIC_FSK address space consists of the GENERIC_FSK registers. The partitioning of address space for GENERIC_FSK is shown in the diagram below.

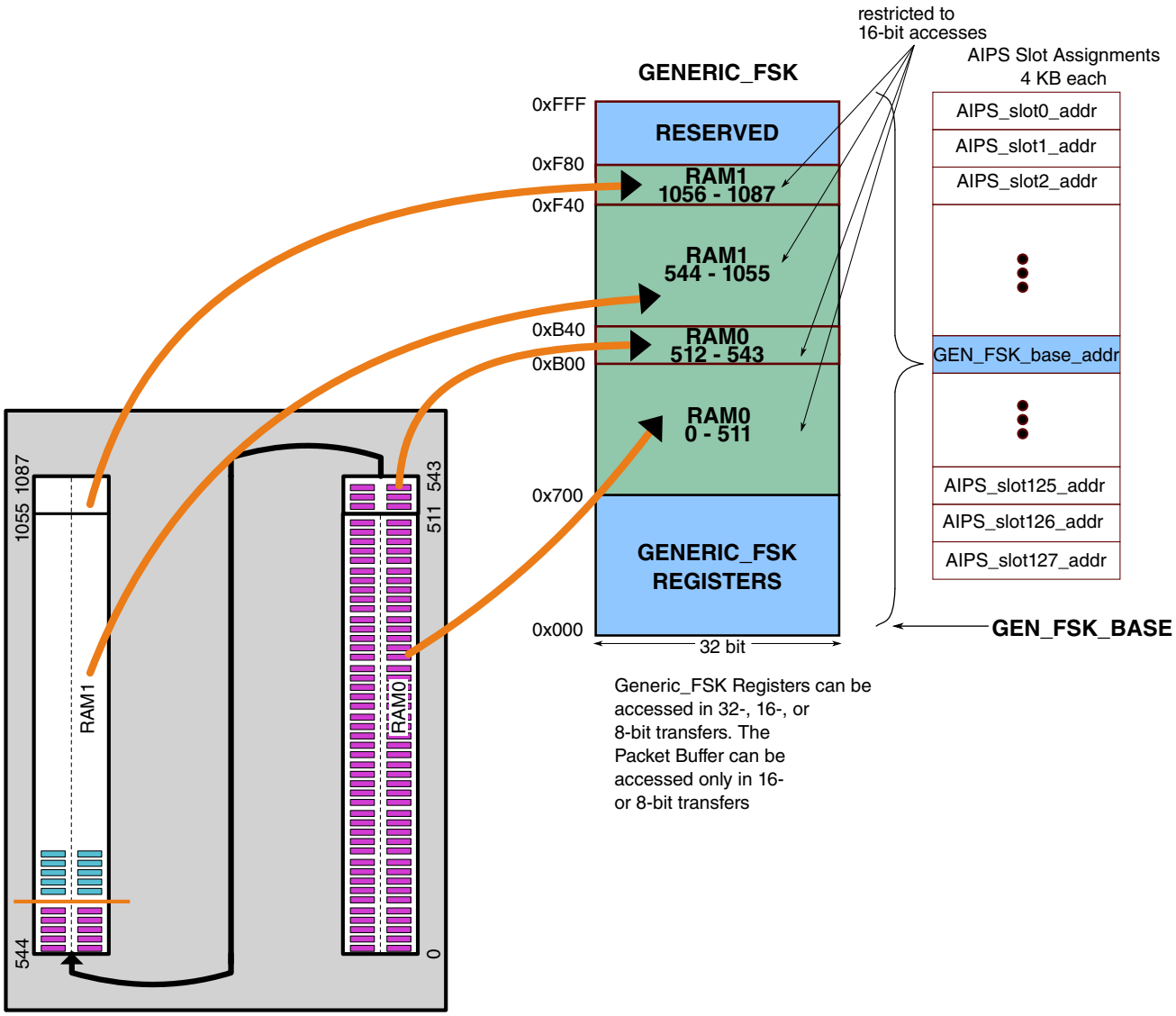



**Figure 64-196. GENERIC_FSK MEMORY MAPPING**

**NOTE**

See AIPS Memory Map for specific memory assignments details.

All registers within GENERIC_FSK memory space, are accessible by the MCU in 8-, 16-, or 32-bit accesses. For efficiency, 32-bit accesses are recommended.

### 64.5.3.3.9 Sequence Timing Diagrams

The following timing diagram depicts an example of GENERIC_FSK packet transmission, highlighting the following events:

1. Software stores packet octets into Packet Buffer, to be transmitted

2. Software loads T1_CMP with desired TX start time, sets T1_CMP_EN=1
3. Software schedules TX sequence with TX_START_T1 command
4. EVENT_TMR matches T1_CMP, TX sequence starts automatically
5. Generic_FSK Link Layer reads octets from Packet Buffer. Packet is transmitted
6. TX_IRQ asserts after last bit of packet transmitted
7. SEQ_END_IRQ asserts after TSM returns to IDLE (TX warmdown complete)

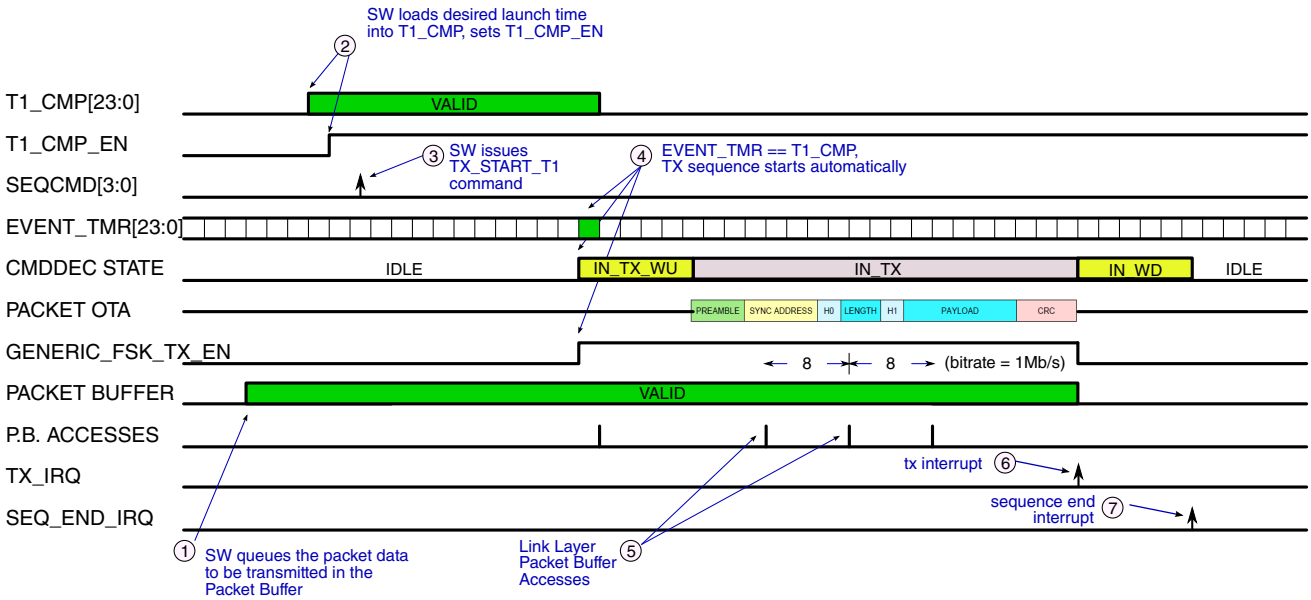



**Figure 64-197. GENERIC FSK TX SEQUENCE: LAUNCH VIA T1 TIMER**

The following timing diagram depicts an example of GENERIC_FSK “good” packet reception, highlighting the following events:

1. Software loads T2_CMP with desired RX start time, sets T2_CMP_EN=1
2. Software schedules RX sequence with RX_START_T2 command
3. EVENT_TMR matches T2_CMP, RX sequence starts automatically
4. Network Address search, followed by NTW_ADR_IRQ interrupt, TIMESTAMP capture
5. Packet reception, Generic_FSK LL write octets to Packet Buffer
6. CRC verification passes, and no H0, H1, or LENGTH_MAX violations, RX_IRQ asserted, Packet RX Buffer valid, RSSI valid
7. SEQ_END_IRQ asserts after TSM returns to IDLE (RX warmdown complete)

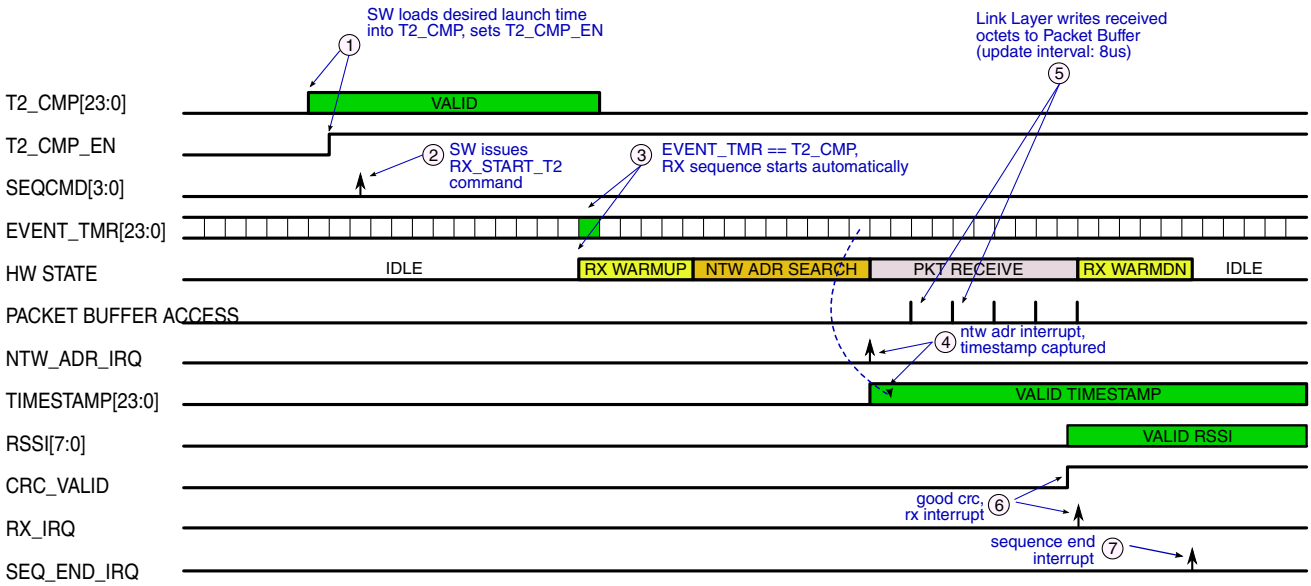



**Figure 64-198. GENERIC FSK RX SEQUENCE: LAUNCH VIA T2 TIMER, GOOD PACKET**

The following timing diagram depicts an example of GENERIC_FSK “bad” packet reception, highlighting the following events:

1. Software loads T2_CMP with desired RX start time, sets T2_CMP_EN=1
2. Software schedules RX sequence with RX_START_T2 command
3. EVENT_TMR matches T2_CMP, RX sequence starts automatically
4. Network Address search, followed by NTW_ADR_IRQ interrupt, TIMESTAMP capture
5. Packet reception, Generic_FSK LL write octets to Packet Buffer
6. CRC verification fails, no CRC_VALID, no RX_IRQ, RX Recycle
7. No SEQ_END_IRQ, return to Network Address Search

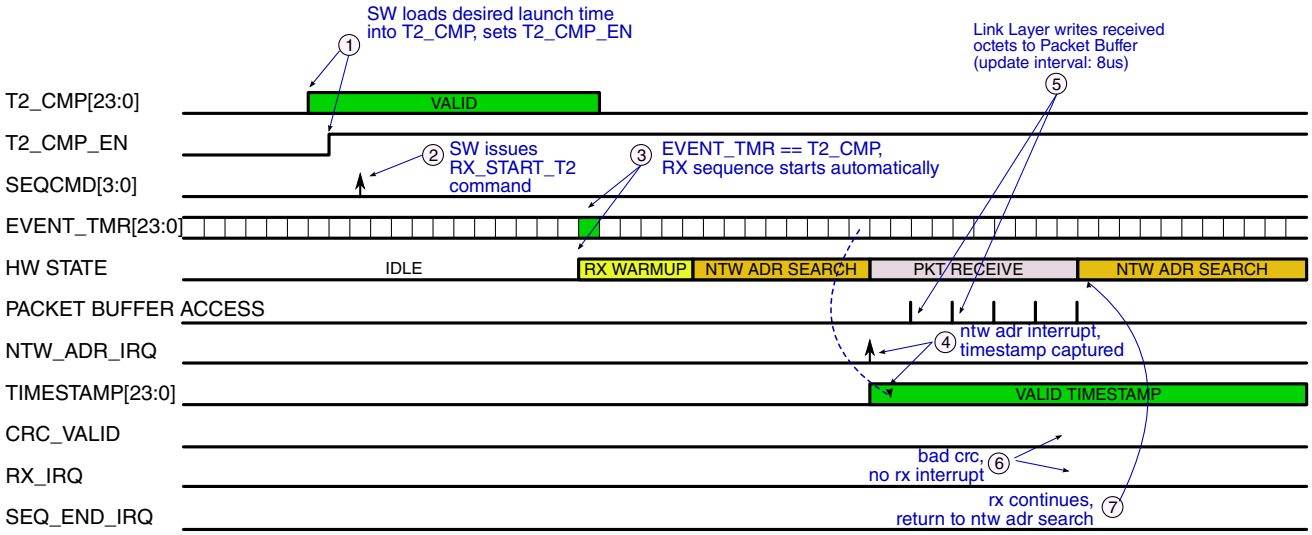



**Figure 64-199. GENERIC FSK RX SEQUENCE: LAUNCH VIA T2 TIMER, BAD PACKET, RX RECYCLE**

The following timing diagram depicts a GENERIC_FSK RX search operation, which fails to detect a Network Address before timing out. The following events are highlighted:

1. Software loads T1_CMP with desired RX start time, sets T1_CMP_EN=1
2. Software loads T2_CMP with desired RX stop time, sets T2_CMP_EN=1
3. Software schedules RX sequence start with RX_START_T1 command
4. Software schedules RX sequence end with RX_START_T2 command
5. EVENT_TMR matches T1_CMP, RX sequence starts automatically
6. EVENT_TMR matches T2_CMP, RX sequence goes into warmdown automatically
7. SEQ_END_IRQ asserts to indicate TSM has returned to IDLE. No NTW_ADR_IRQ, No RX_IRQ.

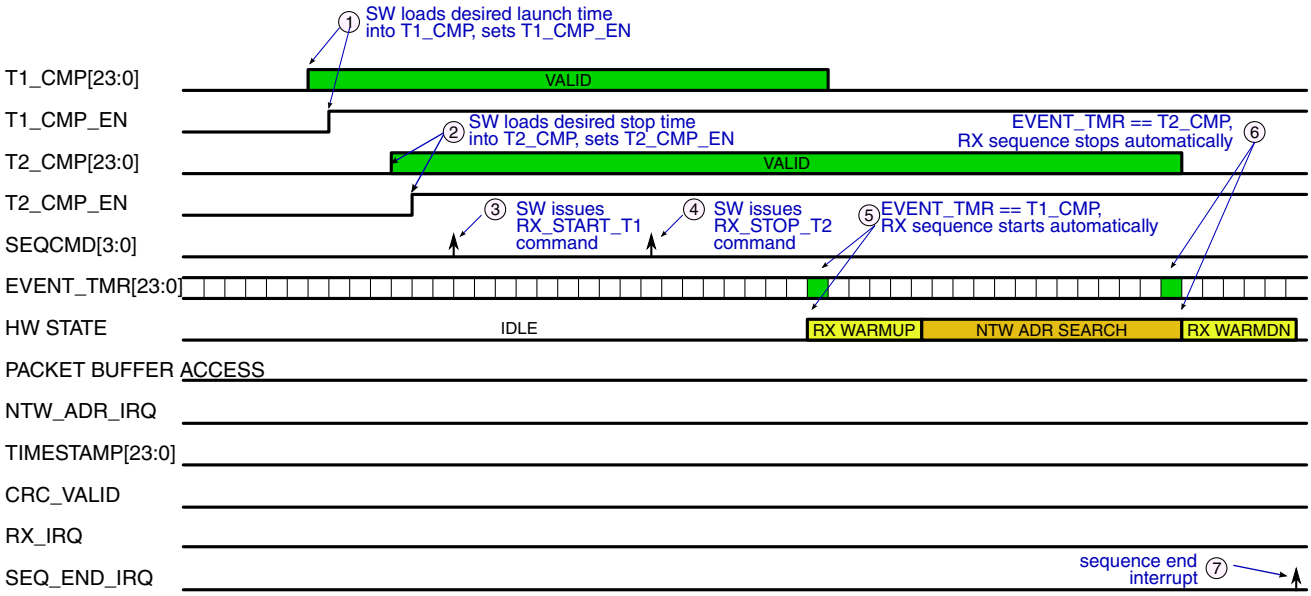



**Figure 64-200. GENERIC FSK RX SEQUENCE: SLAVE CHANNEL SEARCH, LAUNCH VIA T1, STOP VIA T2**

## 64.5.3.3.10 Manual DSM

When the GENERIC_FSK Link Layer anticipates long periods of inactivity, the Link Layer controller can be put into a Deep Sleep Mode (DSM), where all of its clocks are turned off. In addition, the GENERIC_FSK Link Layer can be configured to optionally turn off the RF Oscillator, if it is not otherwise needed by the SoC. Deep Sleep Mode results in dramatic power savings. The procedure described below, is called Manual DSM.

To make Manual DSM possible, the transceiver incorporates a low-frequency, high-precision Deep Sleep Timer, DSM_TIMER. This timer is clocked by a crystal-referenced 32.768KHz oscillator. The timer is 24-bit wide, yielding a 8.5 minute rollover (the maximum length of a deep sleep period).

The hardware to implement DSM within the SoC, is partitioned across several modules, and several power domains. This is to allow the maximum achievable power savings, by retaining full voltage for only those modules which need to be active in DSM, and placing the remaining modules in “state retention” mode. The diagram below depicts the partitioning of DSM hardware across the various modules, including the GENERIC_FSK Link Layer Controller.

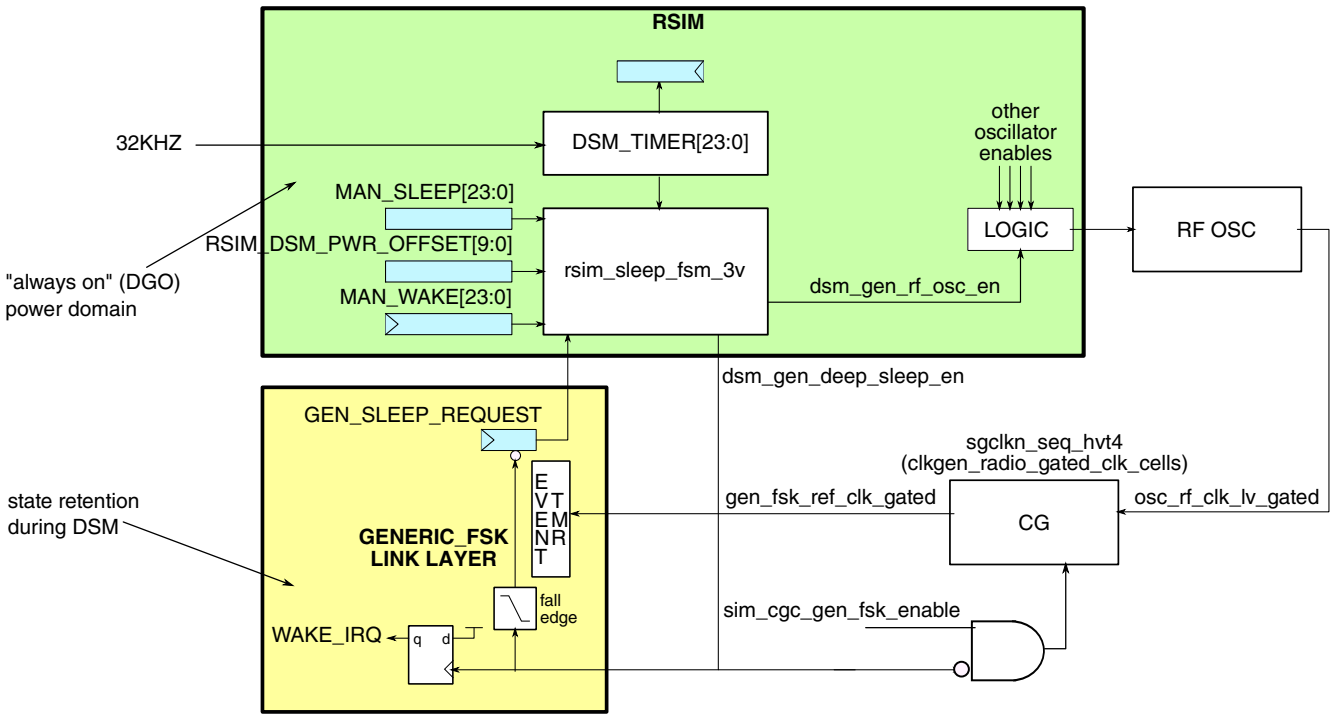



**Figure 64-201. MANUAL DSM BLOCK DIAGRAM**

The concept behind Manual DSM, as it is implemented for GENERIC_FSK, is to allow the GENERIC_FSK timebase to be maintained for a period of time, while the RF Oscillator is turned off and much of the SoC is placed in state-retention. During DSM, the timebase management is temporarily transferred from the GENERIC_FSK EVENT_TMR to the low-power DSM_TIMER, and after DSM exit, timebase management is transferred back to the EVENT_TMR, under software control. During DSM, the EVENT_TMR is frozen. The total time spent in DSM is known to software, so that upon exiting DSM, the EVENT_TMR can be updated to correct for the precise amount of time it was frozen.

DSM entry and exit, and RF Oscillator re-start during Manual DSM, are governed by 5 registers, as described in the table below.

| Field | R/W | Description |
|---|---|---|
| MAN_SLEEP[23:0] | rw | If GEN_SLEEP_REQUEST=1, enter DSM & freeze EVENT_TMR when MAN_SLEEP matches DSM_TIMER.<br>**NOTE:** This register resides in RSIM Address Space |
| RSIM_DSM_PWR_OFFSET[9:0] | rw | Determines the power and RF oscillator re-start times relative to MAN_WAKE<br>**NOTE:** This register resides in RSIM Address Space |

*Table continues on the next page...*

| Field | R/W | Description |
|---|---|---|
| MAN_WAKE[23:0] | rw | Exit Deep Sleep Mode, and resume EVENT_TMR, when MAN_WAKE matches DSM_TIMER<br>**NOTE:** This register resides in RSIM Address Space |
| GEN_SLEEP_REQUEST | w | Enable a match on MAN_SLEEP[23:0] to DSM_TIMER[23:0], to enter Deep Sleep Mode, by writing a 1 to this bit. Cancel a pending sleep request by writing a 0 to it, if already set, at least 2 32.768KHz clock cycles prior to MAN_SLEEP[23:0] |
| DSM_TIMER[23:0] | r | Current State of the 32KHz Sleep Timer<br>**NOTE:** This register resides in RSIM Address Space |

## Procedure

The procedure to schedule a Manual DSM cycle (entry/exit) as well as to program the desired RF Oscillator re-start time, and the calculation to restore the GENERIC_FSK EVENT_TMR after a DSM cycle, is described below.

1. Software determines the future time at which it would like to enter DSM, by reading the DSM_TIMER, computing the number of 32KHz clock cycles remaining until the desired DSM start-time, and writing this value into MAN_SLEEP register. The value programmed into MAN_SLEEP should be no fewer than 4 clocks greater (in the future) than the current time as read from DSM_TIMER.
2. Software determines the future time at which it would like to exit DSM, by computing the number of 32KHz clock cycles remaining until the desired DSM exit-time, and writing this value into MAN_WAKE register.
3. Software writes a 1 to DSM_CTRL[GEN_SLEEP_REQUEST]. This bit resides in GENERIC_FSK address space. Writing this bit to 1 enables a DSM cycle to commence using the values programmed into MAN_SLEEP and MAN_WAKE.
4. MCU can go into a low power state (e.g., STOP, VLPS, or VLLSx)
5. When DSM_TIMER = MAN_SLEEP, the EVENT_TMR will be clock-gated so that it will remain frozen at its current count. The RF Oscillator will be turned off (unless overridden due to a non-Generic-related SoC requirement)
6. All hardware not in the low-voltage domain will be placed into state-retention, or other low-power state.
7. When DSM_TIMER = MAN_WAKE, clocking of the GENERIC_FSK Link Layer Controller will resume and the EVENT_TMR will be ungated (allowed to resume counting).
8. Software computes the time that the SoC was in DSM (and hence the time the EVENT_TMR was frozen), in microseconds, with the equation:

   **(MAN_WAKE – MAN_SLEEP) / 32768 * 1000000**

9. Software increments the GENERIC_FSK EVENT_TMR by this amount, by writing this value to EVENT_TMR[23:0] register with EVENT_TMR_ADD=1.
10. On the next 1MHz clock edge, the EVENT_TMR has been restored to where it would have been, had no DSM occurred, with 1.0-microsecond accuracy.

Note that the 1.0-microsecond accuracy quoted above, assumes an ideal 32.768KHz source. Any ppm error in the 32.768KHz source will degrade the microsecond accuracy proportionally to the magnitude of the error.

Once GEN_SLEEP_REQUEST has been set to 1 to request Manual DSM, the request can be cancelled by writing a 0 to the bit. The cancellation must be performed at least 2 32.768KHz before the MAN_SLEEP[23:0] match to DSM_TIMER[23:0] to avoid a race condition to enter DSM.

The following timing diagram depicts an GENERIC_FSK-driven DSM entry/exit cycle.

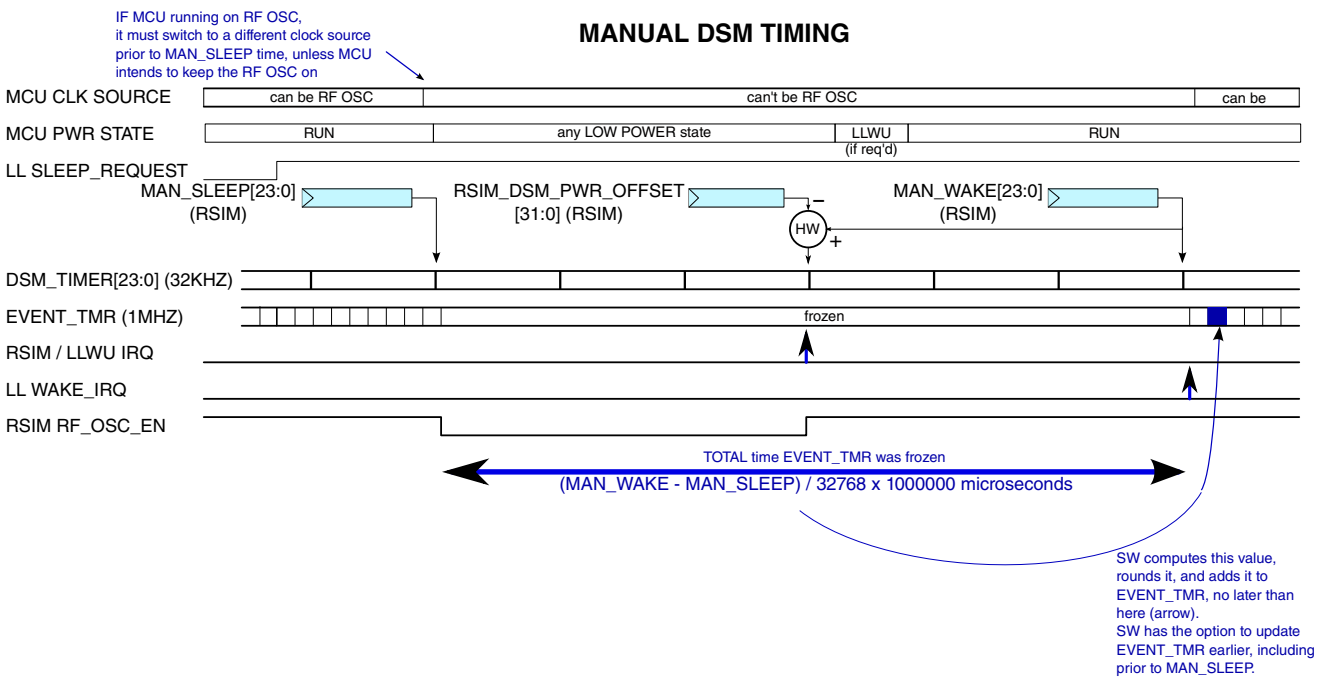



**Figure 64-202. GENERIC_FSK DEEP SLEEP TIMING**

## DSM Early Exit

Circumstances may arise where the Manual DSM cycle may occasionally need to be cut short, due to circumstances beyond the control of the radio. When this happens, the SoC, starting with the MCU, is awakened out of its low power state, long before the RSIM DSM_TIMER reaches MAN_WAKE. Therefore, software handling the event which triggered the wakeup, will determine if the radio needs to be awakened as well. If this is the case, software will advance the MAN_WAKE setting to a point in the near future, but no fewer than 4 32KHz clock cycles beyond the current time ascertained by reading

DSM_TIMER. So, to execute a Manual DSM Early Exit, software shall read DSM_TIMER, add +4 to the value, and write the sum to the RSIM MAN_WAKE regiter. Software shall then wait for the GENERIC_FSK WAKE_IRQ interrupt, which will assert once the full radio wakeup from the low power state has completed. Software should then take whatever steps are necessary to configure the radio to respond to the conditions which triggered the early exit. Servicing of a DSM early exit interrupt is beyond the scope of this document.

#### 64.5.3.3.11 Wake-On-Radio

The GENERIC_FSK Link Layer can take advantage of the automated sequence- and DSM-scheduling provided by Wake-On-Radio. Wake-On-Radio is one of two DSM control schemes that can be selected for GENERIC_FSK, as an alternative to Manual DSM. See Chapter *Wake-On-Radio*.

Several changes to the GENERIC_FSK Link Layer Controller have been made to support Wake-On-Radio.

Interrupt WAKE_IRQ has been made dual-purpose. If Wake-On-Radio is disabled, i.e., WOR_CTRL[WOR_EN]=0, or Wake-On-Radio not assigned to GENERIC_FSK, WAKE_IRQ performs its legacy function for Manual DSM. (See Section Manual DSM). If Wake-On-Radio is enabled, i.e., WOR_EN=1, and assigned to GENERIC_FSK, this interrupt becomes WOR_IRQ, and has new functionality specific to Wake-On-Radio. See Chapter *Wake-On-Radio*, Section *Interrupt Management*, for more details on the meaning, and handling of WOR_IRQ. The Interrupts section of this chapter describes the dual-purpose nature of WAKE_IRQ / WOR_IRQ.

In addition to WOR_IRQ, numerous interface changes have been made to support Wake-On-Radio. The changes allow, for example, WOR to manage GENERIC_FSK's RF interrupts, inhibit timestamp updates, etc. The added port signals are listed in the table below, along with a description or each:

| WAKE-ON-RADIO INTERFACE SIGNAL | DIRECTION | DESCRIPTION |
|---|---|---|
| wor_event_tmr_add[23:0] | WOR -> GENERIC_FSK | The Event Timer "Addback" value computed by WOR to compensate for the amount of time Event Timer was frozen during DSM |
| wor_addback | WOR -> GENERIC_FSK | A 1µs-long pulse commanding the GENERIC_FSK Event Timer to auto-increment by the the amount appearing on wor_event_tmr_add[23:0] |
| wor_force_wake_irq | WOR -> GENERIC_FSK | Force WOR_IRQ to assert, due to an occurrence of one of the 3 WOR_IRQ interrupt triggers |

*Table continues on the next page...*

| WAKE-ON-RADIO INTERFACE SIGNAL | DIRECTION | DESCRIPTION |
|---|---|---|
| wor_block_wake_irq | WOR -> GENERIC_FSK | Inhibit the legacy WAKE_IRQ assertion when WOR_EN=1 (this interrupt becomes WOR_IRQ) |
| tx_start_ext | WOR -> GENERIC_FSK | A 1µs-long pulse to the GENERIC_FSK command decoder to initiate a WOR-driven TX sequence |
| rx_start_ext | WOR -> GENERIC_FSK | A 1µs-long pulse to the GENERIC_FSK command decoder to initiate a WOR-driven RX sequence |
| seq_stop_ext | WOR -> GENERIC_FSK | A 1µs-long pulse to the GENERIC_FSK command decoder to stop (abort) a WOR-driven RX sequence |
| wor_allow_irq | WOR -> GENERIC_FSK | Inhibits RF interrupts, so that interrupt assertions can be managed based on WOR descriptor settings. Controlled dynamically by the WOR state machine. |
| timestamp_inhibit | WOR -> GENERIC_FSK | In Wake-On-Radio, the legacy GENERIC_FSK TIMESTAMP register becomes WOR TIMESTAMP0 (timestamp for SLOT0). This signal blocks this register from updating on SLOTS 1-3 |
| wake_irq_trig | WOR <- GENERIC_FSK | WOR uses this signal to cancel GEN_SLEEP_REQUEST and transition from IN_DSM -> COMP_ADDBACK state |
| event_tmr[23:0] | WOR <- GENERIC_FSK | GENERIC_FSK Event Timer, used by WOR for timestamp capture |
| event_tmr_plus_1[23:0] | WOR <- GENERIC_FSK | GENERIC_FSK Event Timer + 1, used by WOR to schedule RF operation starts and stops |
| tx_irq_trig | WOR <- GENERIC_FSK | TXIRQ trigger, indicates the completion of a packet transmission, used by WOR to transition out of WAKE_SEQ_END state |
| rx_irq_trig | WOR <- GENERIC_FSK | RXIRQ trigger, indicates the completion of a good packet reception, used by WOR to transition out of WAKE_SEQ_END state |
| rx_status_for_wor[2:0] | WOR <- GENERIC_FSK | Internal RX status signals used by WOR to formulate TIMESTAMPx_STS[2:0].<br>**NOTE:** See Section Timestamp Status for Wake-On-Radio following this table |
| timestamp_clk_en | WOR <- CGM <- GENERIC_FSK | Timestamp clock gate enable, the resultant clock is used by WOR to update the TIMESTAMP1,2, and 3 registers |

# Timestamp Status for Wake-On-Radio

The GENERIC_FSK Link Layer has 3 methods for filtering and screening packets as they are being received:

H0 filtering
H1 filtering
LENGTH_MAX filtering

In Normal operation, when one or more of the filtering rules are violated, an RX recycle will occur immediately after the header, and the cause of the filtering violation (status bit) will not be preserved. In addition, the final CRC status can't be known at the point of recycle. In Wake-On-Radio operation, The TIMESTAMPx_STS[1] bit reflects CRC status, and TIMESTAMPx_STS[2] bit reflects Packet Filter status. For an RX recycle at the end of the packet header, the GENERIC_FSK Link Layer won't be able to provide the necessary signalling to Wake-On-Radio in order to properly set those 2 bits of each TIMESTAMPx_STS register, so these bits should be considered invalid. To render them valid, set the GENERIC_FSK PACKET_CFG[REC_BAD_PKT]=1, which will force reception of the entire packet, including CRC, before recycling. This will allow the Wake-On-Radio TIMESTAMPx_STS[2:1] bits to yield valid information.

#### 64.5.3.3.12 Multi-protocol Arbitration

The GENERIC_FSK Link Layer controller may be included in an SoC that includes similar controllers for other protocols (e.g., Bluetooth Low Energy). GENERIC_FSK may be required to operate simultaneously with other protocols on the SoC, and there are scenarios where both protocol engines require access to the RF channel at the same time.

To assist in software arbitration between GENERIC_FSK and other protocols, the read-only status but XCVR_BUSY has been included in the XCVR_STS register. When this bit is set, this indicates that the RF channel is currently in use (TSM is busy), by one of the Radio’s four link layer controllers. When this bit it clear, the RF channel is available to GENERIC_FSK (TSM is idle).

#### 64.5.3.3.13 Wifi Coexistence

Provisions have been made to allow for the 2.4GHz transceiver to coexist in the same space as a WiFi transceiver IC, which shares the same frequency band. The coexistence scheme designates the WiFi transceiver as the master, and the 2.4GHz transceiver as the slave. The objective of the coexistence strategy is to prevent both the WiFi and 2.4GHz transceivers transmitting simultaneously; a configuration option exists to also prevent the 2.4GHz transceiver from receiving when the WiFi transceiver owns the RF channel.

A very basic mechanism is used in cases where 2.4GHz radio does not generate requests, but instead defers to the WiFi transceiver to grant permission to use the medium. The WiFi IC generates a signal, RF_NOT_ALLOWED. If this signal is asserted, then 2.4GHz radio does not perform any communication. When this signal is de-asserted, then 2.4GHz radio is free to perform communications. In case the signal RF_NOT_ALLOWED is de-asserted when 2.4GHz radio has already initiated the transmission/reception of a packet then the 2.4GHz radio must stop its activity immediately .

When RF_NOT_ALLOWED aborting is enabled, the GENERIC_FSK Link Layer hardware must respond to RF_NOT_ALLOWED assertions, by aborting any TX or RX sequence which is currently underway, and Link Layer software must not initiate any new sequences until RF_NOT_ALLOWED has been deasserted. The hardware response must be handled autonomously by the Link Layer, with no MCU intervention required. This is because the RF_NOT_ALLOWED assertions can occur during a Wake-On-Radio timeslot which has been designated as "no-MCU-wakeup" in the slot's descriptor. In this scenario, the aborting of the sequence occurs without any assistance from, or notification of, the MCU.

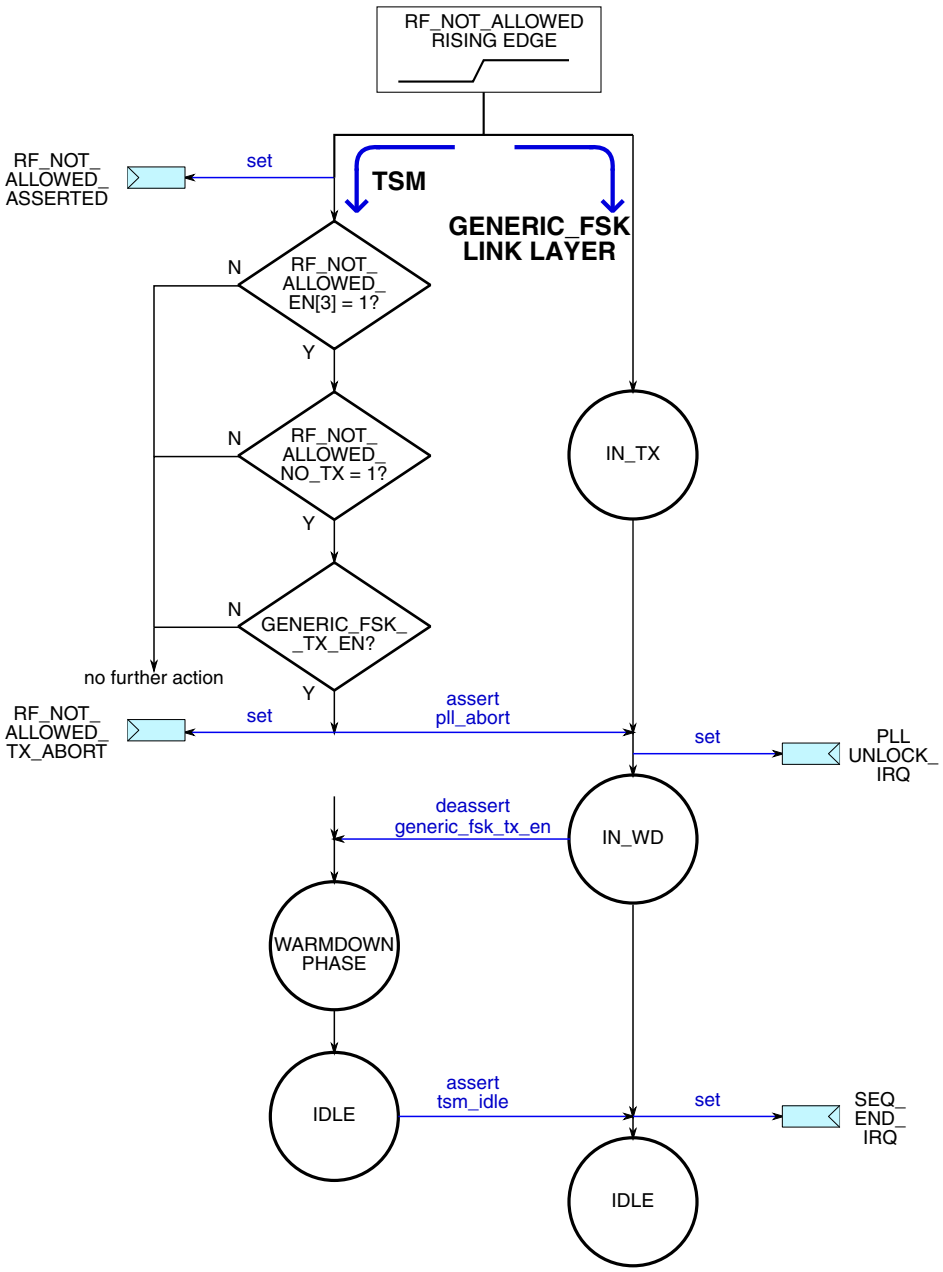
In the multi-protocol 2.4GHz radio, RF_NOT_ALLOWED aborting can be individually enabled/disabled for each protocol engine. For GENERIC_FSK, RF_NOT_ALLOWED_EN[3] is the associated enable control bit. This bit resides in the COEX_CTRL register in XCVR address space. When this bit is 0, transitions on RF_NOT_ALLOWED are ignored by the GENERIC_FSK Link Layer hardware; when this bit is 1, the GENERIC_FSK Link Layer hardware will monitor RF_NOT_ALLOWED at all times, and abort any active sequence which is underway when an assertion on the pin occurs. The complete hardware response to RF_NOT_ALLOWED assertions is described below.

Additional control over RF_NOT_ALLOWED aborting is provided by the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits. If RF_NOT_ALLOWED_NO_TX=1, then an RF_NOT_ALLOWED abort will occur only if a GENERIC_FSK TX sequence is underway (generic_fsk_tx_en=1). GENERIC_FSK TX sequences will not be aborted if RF_NOT_ALLOWED_NO_TX=0. If RF_NOT_ALLOWED_NO_RX=1, then an RF_NOT_ALLOWED abort will occur only if GENERIC_FSK RX sequence is underway (generic_fsk_rx_en=1). GENERIC_FSK

RX sequences will not be aborted if RF_NOT_ALLOWED_NO_RX=0. The RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits reside in the COEX_CTRL register.

For the purposes of triggering a hardware abort, the *pll_abort* input to the GENERIC_FSK Link Layer hardware is used. This is because the hardware response to the RF_NOT_ALLOWED assertion is identical to that of a PLL unlock event. This also means that, when RF_NOT_ALLOWED aborting is enabled for GENERIC_FSK the PLL_UNLOCK_IRQ interrupt status bit, will be dual purpose: it will not only indicate a PLL unlock condition, but also a RF_NOT_ALLOWED abort. PLL aborting and RF_NOT_ALLOWED aborting are enabled separately, with the COEX_CTRL register maintaining the control bits required for the latter. The status bits for RF_NOT_ALLOWED aborting are also available in COEX_CTRL, so that software will be able to distinguish the source of the PLL_UNLOCK_IRQ, in case both PLL and RF_NOT_ALLOWED aborting are enabled.

The sequence of events which results in a hardware abort of an GENERIC_FSK TX operation triggered by an assertion on RF_NOT_ALLOWED, is a collaboration between the TSM (Transceiver Sequence Manager) and the GENERIC_FSK Link Layer hardware, as shown in the following diagram.



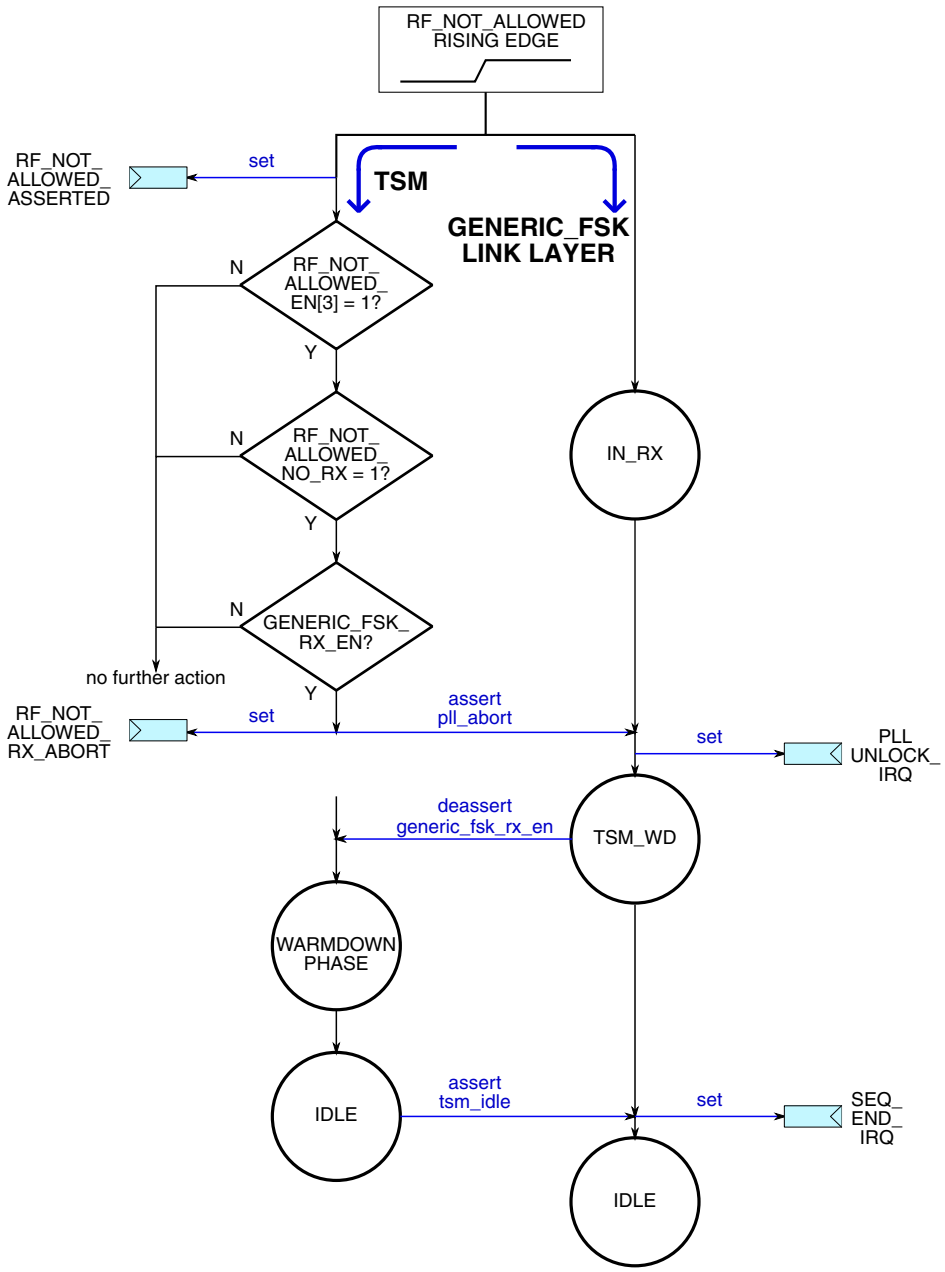

Upon assertion on RF_NOT_ALLOWED, the TSM sets the RF_NOT_ALLOWED_ASSERTED status bit in COEX_CTRL, then checks that the conditions for a hardware abort are all met:

- RF_NOT_ALLOWED_EN[3] = 1, which enables GENERIC_FSK to respond to RF_NOT_ALLOWED events
- RF_NOT_ALLOWED_NO_TX = 1, which enables TX operations to be aborted
- generic_fsk_tx_en = 1, TX request to TSM from GENERIC_FSK, indicating TX operation in progress

If all conditions are not met, no further action is taken. Otherwise, the TSM sets COEX_CTRL[RF_NOT_ALLOWED_TX_ABORT], and also asserts *pll_unlock* to the GENERIC_FSK Link Layer hardware. GENERIC_FSK responds by asserting IRQ_CTRL[PLL_UNLOCK_IRQ] in GENERIC_FSK address space. GENERIC_FSK enters IN_WD state, which deasserts generic_fsk_tx_en to the TSM. This initiates the TSM TX warmdown. GENERIC_FSK holds in IN_WD to wait for TSM to return to idle. Once this occurs, GENERIC_FSK asserts IRQ_CTRL[SEQ_END_IRQ] and returns to its IDLE state. Three status bits are now set to indicate to software that the source of the abort was an RF_NOT_ALLOWED assertion:

- IRQ_CTRL[PLL_UNLOCK_IRQ]
- COEX_CTRL[RF_NOT_ALLOWED_ASSERTED]
- COEX_CTRL[RF_NOT_ALLOWED_TX_ABORT]

The sequence of events which results in a hardware abort of an GENERIC_FSK RX sequence triggered by an assertion on RF_NOT_ALLOWED, is shown in the following diagram.




Upon assertion on RF_NOT_ALLOWED, the TSM sets the RF_NOT_ALLOWED_ASSERTED status bit in COEX_CTRL, then checks that the conditions for a hardware abort are all met:

- RF_NOT_ALLOWED_EN[3] = 1, which enables GENERIC_FSK to respond to RF_NOT_ALLOWED events
- RF_NOT_ALLOWED_NO_RX = 1, which enables RX operations to be aborted
- generic_fsk_rx_en = 1, RX request to TSM from GENERIC_FSK, indicating RX operation in progress

If all conditions are not met, no further action is taken. Otherwise, the TSM sets COEX_CTRL[RF_NOT_ALLOWED_RX_ABORT], and also asserts *pll_unlock* to the GENERIC_FSK Link Layer hardware. GENERIC_FSK responds by asserting IRQ_CTRL[PLL_UNLOCK_IRQ] in GENERIC_FSK address space. GENERIC_FSK enters IN_WD state, which deasserts generic_fsk_rx_en to the TSM. This initiates the TSM TX warmdown. GENERIC_FSK holds in IN_WD to wait for TSM to return to idle. Once this occurs, GENERIC_FSK asserts IRQ_CTRL[SEQ_END_IRQ] and returns to its IDLE state. Three status bits are now set to indicate to software that the source of the abort was an RF_NOT_ALLOWED assertion:

- IRQSTS1[PLL_UNLOCK_IRQ]
- COEX_CTRL[RF_NOT_ALLOWED_ASSERTED]
- COEX_CTRL[RF_NOT_ALLOWED_RX_ABORT]

The complete set of control and status bits in the COEX_CTRL register is described in the following table:

| Field | R/W | Description |
|---|---|---|
| RF_NOT_ALLOWED_EN | RW | The coexistence input RF_NOT_ALLOWED can be enabled to selectively abort TX or RX sequences, with individual enables for each supported protocol, subject also to the state of the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits, according to the table below.<br>**xxx1**: RF_NOT_ALLOWED assertions are enabled to abort BLE TX and RX sequences<br>**xxx0**: RF_NOT_ALLOWED assertions are not enabled to abort BLE TX and RX sequences<br>**x1xx**: RF_NOT_ALLOWED assertions are enabled to abort 802.15.4 TX and RX sequences<br>**x0xx**: RF_NOT_ALLOWED assertions are not enabled to abort 802.15.4 TX and RX sequences<br>**1xxx**: RF_NOT_ALLOWED assertions are enabled to abort GENERIC_FSK TX and RX sequences<br>**0xxx**: RF_NOT_ALLOWED assertions are not enabled to abort GENERIC_FSK TX and RX sequences |
| RF_NOT_ALLOWED_NO_TX | RW | TX Sequences will be aborted by RF_NOT_ALLOWED assertions, if the appropriate RF_NOT_ALLOWED_EN[x] bit is set. |
| RF_NOT_ALLOWED_NO_RX | RW | RX Sequences will be aborted by RF_NOT_ALLOWED assertions, if the appropriate RF_NOT_ALLOWED_EN[x] bit is set. |
| RF_NOT_ALLOWED_ASSERTED | R/W1TC | RF_NOT_ALLOWED has gone from not asserted, to asserted, since the last time this bit was cleared |
| RF_NOT_ALLOWED_TX_ABORT | R/W1TC | An TX sequence has been aborted since the last time this bit was cleared |
| RF_NOT_ALLOWED_TX_ABORT | R/W1TC | An TX sequence has been aborted since the last time this bit was cleared |
| RF_NOT_ALLOWED | R | Reflects the instantaneous state of the RF_NOT_ALLOWED pin, |

# Appendix A
# Radio Register Summary

## A.1 RSIM Memory Map and Register Definition

### A.1.1 RSIM register descriptions

#### A.1.1.1 RSIM Memory map

RSIM base address: 4102_F000h

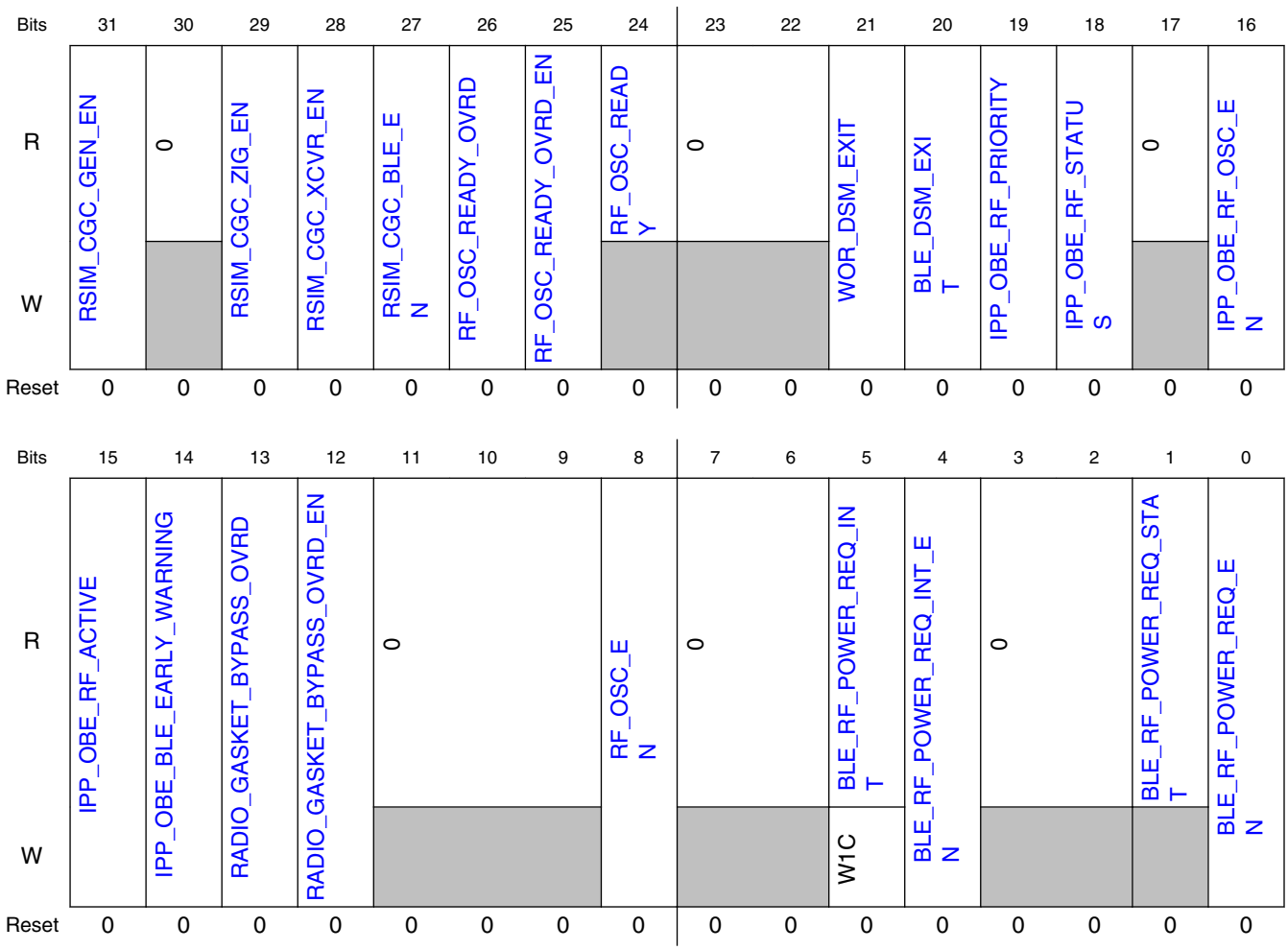
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Radio System Control (CONTROL) | 32 | RW | 0000_0000h |
| 10h | Radio Miscellaneous (MISC) | 32 | RW | 0500_0000h |
| 14h | RSIM Power Control (POWER) | 32 | RW | 7000_0307h |
| 18h | Radio Software Configuration (SW_CONFIG) | 32 | RW | 0000_0020h |
| 100h | Deep Sleep Timer (DSM_TIMER) | 32 | RO | 0000_0000h |
| 104h | Deep Sleep Timer Control (DSM_CONTROL) | 32 | RW | 0000_0000h |
| 108h | Deep Sleep Wakeup Sequence (DSM_WAKEUP) | 32 | RW | 0000_0000h |
| 10Ch | WOR Deep Sleep Duration (WOR_DURATION) | 32 | RO | 0000_0000h |
| 110h | WOR Deep Sleep Wake Time (WOR_WAKE) | 32 | RW | 0000_0000h |
| 11Ch | MAN Deep Sleep Time (MAN_SLEEP) | 32 | RW | 0000_0000h |
| 120h | MAN Deep Sleep Wake Time (MAN_WAKE) | 32 | RW | 0000_0000h |
| 124h | Radio Oscillator Control (RF_OSC_CTRL) | 32 | RW | A020_3806h |
| 128h | Radio Analog Test Registers (ANA_TEST) | 32 | RW | 0000_0000h |
| 12Ch | Radio Analog Trim Registers (ANA_TRIM) | 32 | RW | 784B_0000h |

# A.1.1.2 Radio System Control (CONTROL)

## A.1.1.2.1 Offset

| Register | Offset |
|---|---|
| CONTROL | 0h |

## A.1.1.2.2 Diagram




## A.1.1.2.3 Fields

| Field | Function |
|---|---|
| 31<br>RSIM_CGC_GEN_EN | GEN Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the Generic Link Layer.<br>0b - Clock disabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - Clock enabled |
| 30<br>— | Reserved |
| 29<br>RSIM_CGC_ZIG_EN | ZIG Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the 802.15.4 Link Layer.<br>0b - Clock disabled<br>1b - Clock enabled |
| 28<br>RSIM_CGC_XCVR_EN | XCVR Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the Transceiver.<br>0b - Clock disabled<br>1b - Clock enabled |
| 27<br>RSIM_CGC_BLE_EN | BLE Clock Gate Control<br>This RSIM Clock Gate Enable bit controls the clock gate to the BLE Link Layer.<br>0b - Clock disabled<br>1b - Clock enabled |
| 26<br>RF_OSC_READY_OVRD | RF Ref Osc Ready Override<br>This bit directly controls the Radio RF Ref Osc Ready signal when the RF Ref Osc Ready Override is enabled. All Radio and SoC signals that are derived from the RF Ref Osc Ready signal can be overridden using this bit. |
| 25<br>RF_OSC_READY_OVRD_EN | RF Ref Osc Ready Override Enable<br>This bit enables the RF Ref Osc Ready Override bit. |
| 24<br>RF_OSC_READY | RF Ref Osc Ready<br>The RF Reference Oscillator has an internal counter that gates off the RF Ref Osc clock until the selected count is reached. This bit shows the status of the RF Ref Osc ready signal that comes from that counter. The RF Ref Osc Ready signal can be overridden using the RF_OSC_READY_OVRD bit. |
| 23-22<br>— | Reserved |
| 21<br>WOR_DSM_EXIT | Wake on Radio Force Deep Sleep Mode Exit<br>This bit forces the Wake on Radio State Machine to wakeup from Deep Sleep Mode. |
| 20<br>BLE_DSM_EXIT | BLE Force Deep Sleep Mode Exit<br>This bit forces the BLE link layer to wakeup from Deep Sleep Mode. |
| 19<br>IPP_OBE_RF_PRIORITY | IPP_OBE_RF_PRIORITY<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Priority signal as a pin interface option. |
| 18<br>IPP_OBE_RF_STATUS | IPP_OBE_RF_STATUS<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Status signal as a pin interface option. |
| 17<br>— | Reserved |
| 16<br>IPP_OBE_RF_OSC_EN | IPP_OBE_RF_OSC_EN<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Osc Enable signal as a pin interface option. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>IPP_OBE_RF_ACTIVE | IPP_OBE_RF_ACTIVE<br>This bit enables the Output Driver (OBE) on the SoC port that provides the RF Active signal as a pin interface option. |
| 14<br>IPP_OBE_BLE_EARLY_WARNING | IPP_OBE_BLE_EARLY_WARNING<br>This bit enables the Output Driver (OBE) on the SoC port that provides the BLE Early Warning signal as a pin interface option. |
| 13<br>RADIO_GASKET_BYPASS_OVRD | Radio Gasket Bypass Override<br>This bit directly controls the SoC platform asynchronous gasket bypass signal when the Gasket Bypass Override is enabled.The default behavior of the SoC Asynchronous Gasket is Not Bypassed, which means the Radio registers are accessed using the RF Ref Osc clock. The Radio sends the RF Ref Osc to the asynchronous gasket which then uses that clock for SoC register accesses of the Radio registers. This requires the RF Ref Osc to be Enabled and Ready. If the RF Ref Osc is not Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, which forces the use of the SoC IPG clock as the register access clock. If the RF Ref Osc Is Enabled and Ready, then the IPS Gasket can be overridden to be in Bypass Mode, but this is an ILLEGAL ACCESS if the Radio is operational, DO NOT write Radio registers with the Asynchronous Gasket Bypassed when the Radio is Operating. Test mode access is allowed but glitches may occur. The intent of Bypass Mode is to allow software to configure the Radio before the RF Ref Osc is Enabled and Ready. Note that the BLE Link Layer registers can only be accessed when the RF Ref Osc is Enabled and Ready.<br>0b - XCVR and Link Layer Register Clock is the RF Ref Osc Clock<br>1b - XCVR and Link Layer Register Clock is the SoC IPG Clock |
| 12<br>RADIO_GASKET_BYPASS_OVRD_EN | Radio Gasket Bypass Override Enable<br>The SoC platform has an asynchronous gasket that allows register access for the Radio registers in all SoC clocking modes. This bit allows software to directly control the SoC platform asynchronous gasket bypass signal. |
| 11-9<br>— | Reserved |
| 8<br>RF_OSC_EN | RF Ref Osc Enable<br>The RF Reference Oscillator can be enabled by a Radio link layer (Including the Link Layer use of Deep Sleep Modes), by an internal SoC clock mode, by an External Pin request, or by this bit.<br>If this bit is 0, then in Radio Run Mode, the RF Ref Osc can be turned on by the SoC, or by a Radio link layer using RRun Mode (Including the Link Layer use of Deep Sleep Modes), or by an External Pin request. Warning: The RF Ref Osc can not be enabled by the SoC until the Radio has entered RRun Mode.<br>If this bit is 0, then in Radio Stop Modes, the RF Ref Osc will be off (except that the external pin request can turn the RF Osc on even if the RF Power is off). Warning: Radio will not go into Stop mode while the RF Ref Osc is being used by the SoC, or by a Radio link layer unless the Link Layer is using a Deep Sleep State Machine.<br>If this bit is 1, then the RF Ref Osc will be on in Radio Run Mode. Warning: The RF Ref Osc can not be enabled until the Radio has entered RRun Mode.<br>If this bit is 1, then the RF Ref Osc will be off in Radio Stop Modes (except that the external pin request can turn the RF Osc on even if the RF Power is off). Warning: Radio will not go into Stop mode while the RF Ref Osc is On. |
| 7-6<br>— | Reserved |
| 5 | BLE RF Power Request Interrupt Flag<br>This bit is the BLE_RF_POWER_REQ_INT interrupt flag. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BLE_RF_POWER_REQ_INT | This interrupt flag is cleared by writing a 1 to it. |
| 4<br>BLE_RF_POWER_REQ_INT_EN | BLE RF Power Request Interrupt Enable<br>This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the BLE RF Power Request is asserted high. |
| 3-2<br>— | Reserved |
| 1<br>BLE_RF_POWER_REQ_STAT | BLE RF Power Request Status<br>This bit indicates the current status of the BLE link layer request to turn on the RF Power. |
| 0<br>BLE_RF_POWER_REQ_EN | BLE RF Power Request Enable<br>This bit resets on POR only.<br>If this bit is cleared (the default state), then all BLE link layer requests to turn on the RF Power will be blocked and ignored.<br>In BLE protocols the BLE link layer will always restart when exiting reset by first Requesting the RF Power, this bit blocks that behavior until software configures the Radio and enables the requests. |

## A.1.1.3 Radio Miscellaneous (MISC)

### A.1.1.3.1 Offset

| Register | Offset |
|---|---|
| MISC | 10h |

### A.1.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RADIO_VERSION | | | | | | | | 0 | | | | Reserved | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | Reserved | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.1.1.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RADIO_VERSION | Radio Version ID number<br>This register value can be read to identify the Version of the Radio. Note that the value will read as zero unless the Radio is in Run Mode.<br>B = 2.4 GHz Radio 2.0<br>8 = 2.4 GHz Radio 2.1<br>4 = 2.4 GHz Radio 3.0<br>5 = 2.4 GHz Radio 3.1 |
| 23-20<br>— | Reserved |
| 19-16<br>— | Reserved |
| 15-5<br>— | Reserved |
| 4-0<br>— | Reserved |

## A.1.1.4 RSIM Power Control (POWER)

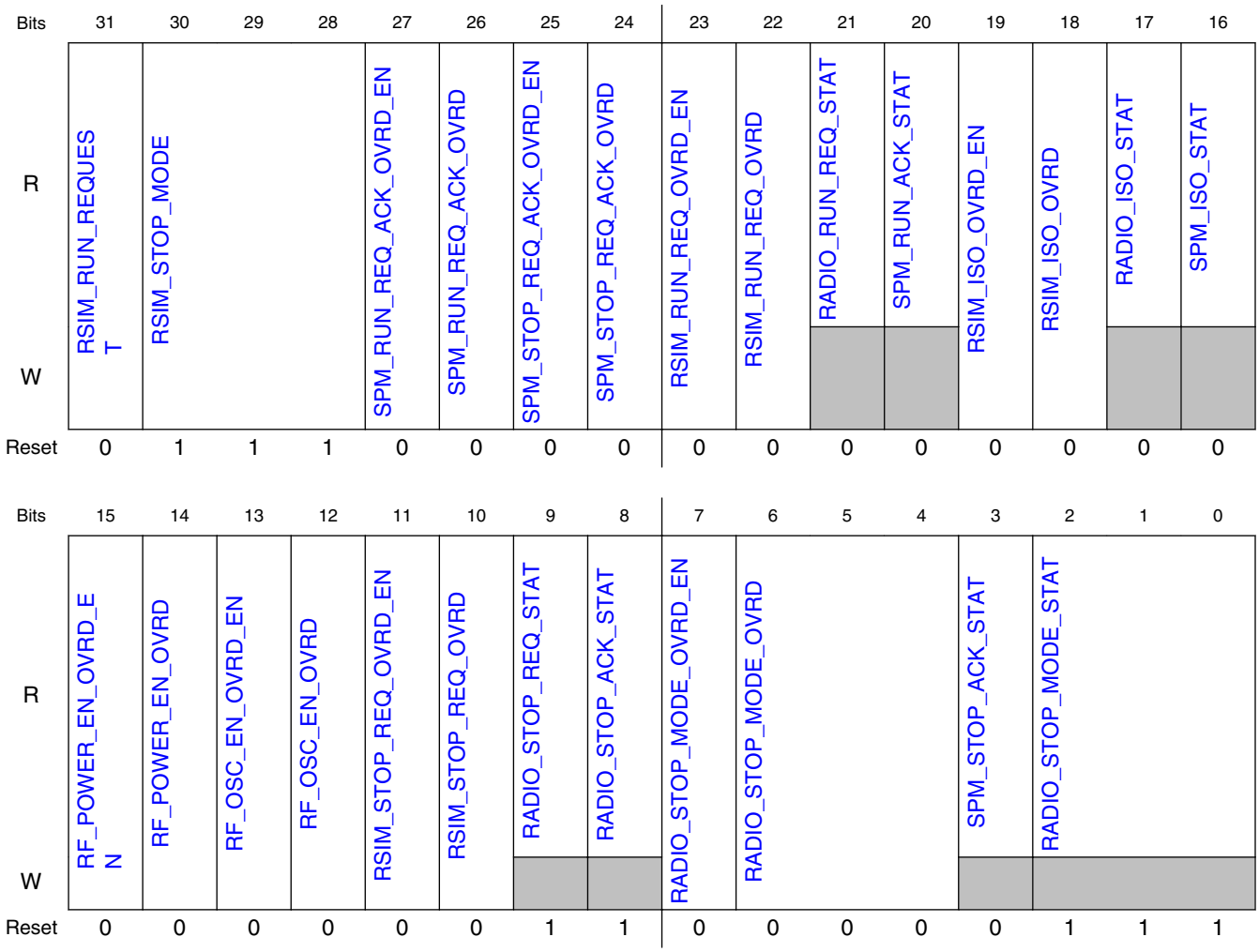
### A.1.1.4.1 Offset

| Register | Offset |
|---|---|
| POWER | 14h |

### A.1.1.4.2 Function

This register contains the power control and status bits.

## A.1.1.4.3 Diagram




## A.1.1.4.4 Fields

| Field | Function |
|---|---|
| 31<br>RSIM_RUN_REQUEST | RSIM Run Regulator Request<br>When this bit is set, the RSIM will request the SPM Run Regulator to be turned on and the RSIM will then stop requesting Stop after the Run Regulator Acknowledge signal is received from the SPM module. |
| 30-28<br>RSIM_STOP_MODE | RSIM lowest allowed Stop Mode<br>This is the Stop Mode that will be requested when the RSIM asserts a Stop request to the SPM module.<br>000b - Reserved<br>001b - Reserved<br>011b - RLLS mode (Radio State Retention mode)<br>111b - RVLLS mode (This is the POR setting) |
| 27 | SPM Run Request Acknowledge Override Enable<br>This bit enables the SPM Run Request Acknowledge Override bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SPM_RUN_REQ_ACK_OVRD_EN | |
| 26<br>SPM_RUN_REQ_ACK_OVRD | SPM Run Request Acknowledge Override<br>This bit directly controls the SPM Run Request Acknowledge signal when the SPM Run Request Acknowledge Override is enabled. |
| 25<br>SPM_STOP_REQ_ACK_OVRD_EN | SPM Stop Request Acknowledge Override Enable<br>This bit enables the SPM Stop Request Acknowledge Override bit. |
| 24<br>SPM_STOP_REQ_ACK_OVRD | SPM Stop Request Acknowledge Override<br>This bit directly controls the SPM Stop Request Acknowledge signal when the SPM Stop Request Acknowledge Override is enabled. |
| 23<br>RSIM_RUN_REQ_OVRD_EN | RSIM Run Request Override Enable<br>This bit enables the RSIM Run Request Override bit. |
| 22<br>RSIM_RUN_REQ_OVRD | RSIM Run Request Override<br>This bit directly controls the RSIM Run Request signal to the SPM when the RSIM Run Request Override is enabled. |
| 21<br>RADIO_RUN_REQ_STAT | Radio Run Request Status<br>This is the value of the Radio Run Request signal being sent to the SPM module, including any overrides. |
| 20<br>SPM_RUN_ACK_STAT | SPM Run Request Acknowledge Status<br>This is the value of the SPM Run request Acknowledge signal being sent to the Radio. |
| 19<br>RSIM_ISO_OVRD_EN | RSIM ISO Override Enable<br>This bit enables the RSIM ISO Override bit. |
| 18<br>RSIM_ISO_OVRD | RSIM ISO Override<br>This bit directly controls the Radio ISO signal when the RSIM ISO Override is enabled. |
| 17<br>RADIO_ISO_STAT | Radio Isolation Status<br>This is the value of the RSIM ISO signal being sent to the Radio Isolation logic, including any overrides. |
| 16<br>SPM_ISO_STAT | SPM ISO Status<br>This is the value of the SPM ISO signal being sent to the Radio. |
| 15<br>RF_POWER_EN_OVRD_EN | Radio Power Enable Override Enable<br>This bit enables the Radio Power Enable Override bit. |
| 14<br>RF_POWER_EN_OVRD | Radio Power Enable Override<br>This bit directly controls the RF_POWER_EN signal when the Radio Power Enable Override is enabled. |
| 13 | Radio Osc Enable Override Enable<br>This bit enables the Radio Osc Enable Override bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RF_OSC_EN_OVRD_EN | |
| 12<br>RF_OSC_EN_OVRD | Radio Osc Enable Override<br>This bit directly controls the RF_OSC_EN SoC pin signal when the Radio Osc Enable Override is enabled. |
| 11<br>RSIM_STOP_REQ_OVRD_EN | Radio Stop Request Override Enable<br>This bit enables the Radio Stop Request Override bit. |
| 10<br>RSIM_STOP_REQ_OVRD | Radio Stop Request Override<br>This bit directly controls the Radio Stop Request signal when the Radio Stop Request Override is enabled. |
| 9<br>RADIO_STOP_REQ_STAT | Radio Stop Request Status<br>This is the value of the Radio Stop Request signal being sent to the SPM module, including any overrides. |
| 8<br>RADIO_STOP_ACK_STAT | Radio Stop Acknowledge Status<br>This is the status of the Stop request Acknowledge signal being used by the Radio, including any override of the SPM module signal. |
| 7<br>RADIO_STOP_MODE_OVRD_EN | Radio Stop Mode Override Enable<br>This bit enables the Radio Stop Mode Override bit. |
| 6-4<br>RADIO_STOP_MODE_OVRD | Radio Stop Mode Override<br>This bit directly controls the Radio Stop Mode signal when the Radio Stop Mode Override is enabled. |
| 3<br>SPM_STOP_ACK_STAT | SPM Stop Acknowledge Status<br>This is the status of the Stop request Acknowledge signal received from the SPM module. |
| 2-0<br>RADIO_STOP_MODE_STAT | Radio Stop Mode Status<br>This is the value of the Radio Stop Mode bus that is sent to the SPM module. |

## A.1.1.5 Radio Software Configuration (SW_CONFIG)

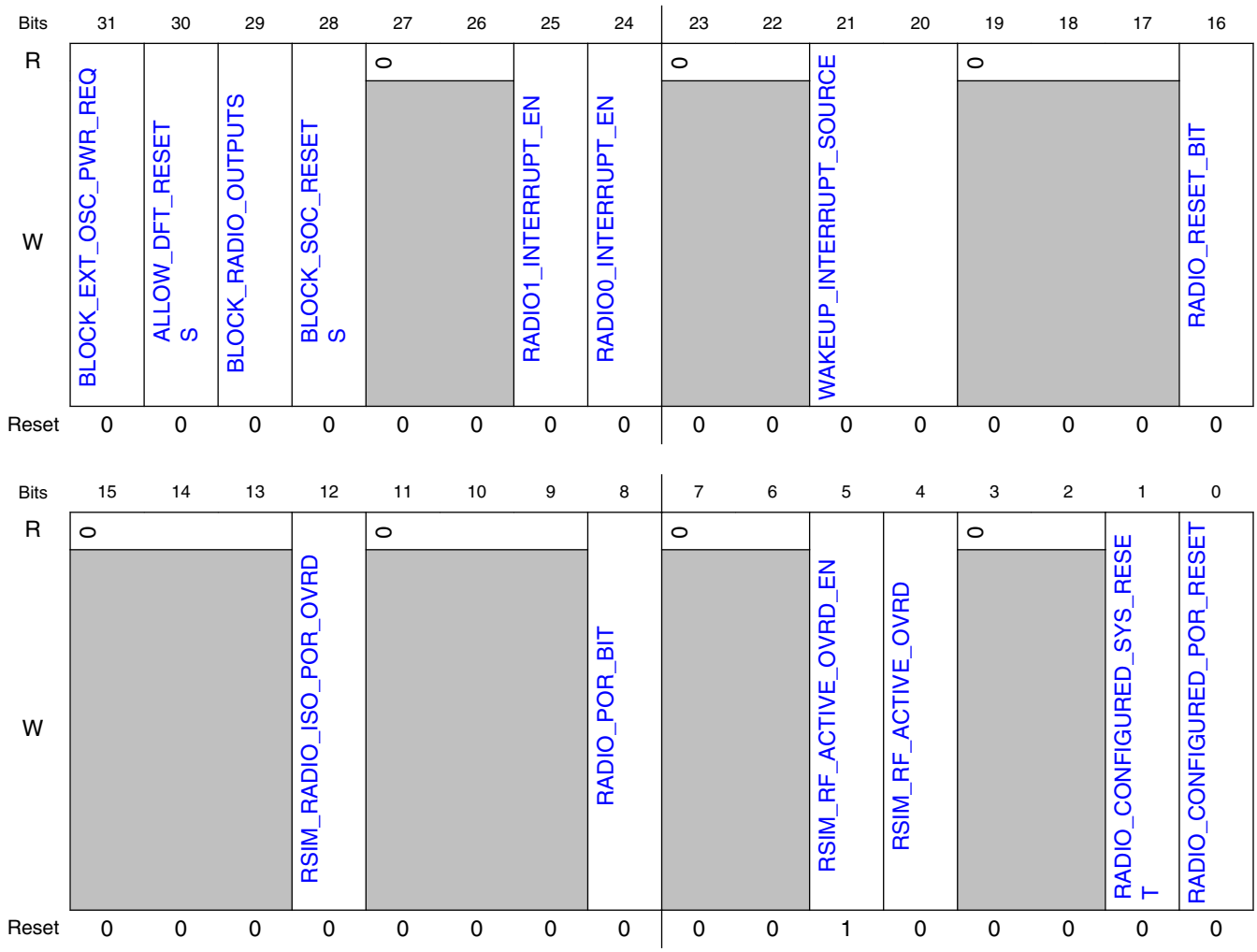
### A.1.1.5.1 Offset

| Register | Offset |
|---|---|
| SW_CONFIG | 18h |

### A.1.1.5.2 Function

This register contains the software configuration and control bits.

### A.1.1.5.3 Diagram




### A.1.1.5.4 Fields

| Field | Function |
|---|---|
| 31<br>BLOCK_EXT_OSC_PWR_REQ | Block External Requests for RF OSC from starting a Radio Power Wakeup Sequence<br>If this bit is set then any External request for the OSC will not start a Radio Power Wakeup Sequence, otherwise any such request does request full Run Regulation mode. |
| 30<br>ALLOW_DFT_RESETS | Allow the DFT Reset Pin to Reset the Radio<br>The Radio provides a port that can be connected to a test mode pin in order to provide a method of resetting the Radio independently from SoC resets. This bit enables that DFT functionality. |
| 29<br>BLOCK_RADIO_OUTPUTS | Block Radio Outputs<br>This bit resets on POR only. This bit is intended to allow the SoC to perform concurrent testing of various SoC logic while the Radio is operating independently. Any Radio output signals that go to the SoC will be blocked so as to not affect the SoC testing when this bit is set. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 28<br>BLOCK_SOC_RESETS | Block SoC Resets of the Radio, cleared by Radio System Reset<br>Most SoC System or Power On Resets will be blocked and the Radio will not be affected by them when this bit is set.<br>The Radio Power Mode logic is not blocked by this bit and will reset on any SoC Power On Reset. |
| 27-26<br>— | Reserved |
| 25<br>RADIO1_INTERRUPT_EN | Radio1 Interrupt Enable<br>This bit enables the Radio1 interrupt source, as selected by the Radio1 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal (typically connected to the MCU LLWU). |
| 24<br>RADIO0_INTERRUPT_EN | Radio0 Interrupt Enable<br>This bit enables the Radio0 interrupt source, as selected by the Radio0 IRQ Select bits, to be added to the interrupt sources that are used to generate the RSIM Asynchronous MCU Wakeup signal (typically connected to the MCU LLWU). |
| 23-22<br>— | Reserved |
| 21-20<br>WAKEUP_INTERRUPT_SOURCE | RSIM Wakeup Interrupt Source Selector<br>The Radio Power-On Sequence can generate an interrupt at three times in the sequence, these bits make that selection.<br>00b - No Radio Power-On Sequence interrupt will be generated.<br>01b - A Power-On Sequence interrupt will be generated when the RF Power Request occurs, including unblocked requests from an external source to use the RF OSC.<br>10b - A Power-On Sequence interrupt will be generated when the RF OSC Request occurs, but not if the RF OSC request was from an external source.<br>11b - A Power-On Sequence interrupt will be generated when the RSIM RF Active Warning occurs |
| 19-17<br>— | Reserved |
| 16<br>RADIO_RESET_BIT | Software System Reset for the Radio<br>This bit only clears if there is an SoC Power On Reset.<br>Setting this bit will reset the Global Radio and RSIM logic, including all Link Layer logic.<br>The logic will stay in reset until this bit is cleared.<br>This bit will not reset the Power Mode logic or the Deep Sleep Mode related logic. |
| 15-13<br>— | Reserved |
| 12<br>RSIM_RADIO_ISO_POR_OVRD | RSIM ISO_POR Override<br>This bit directly controls the Radio ISO_POR signal when the RSIM_ISO_OVRD_EN bit is set. |
| 11-9<br>— | Reserved |
| 8<br>RADIO_POR_BIT | Software Power On Reset for the Radio<br>This bit only clears if there is an SoC Power On Reset.<br>Setting this bit will reset all the radio logic that uses the Radio Power On Reset or Radio VLLS Reset signal, it will be held in reset until this bit is cleared. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-6<br>— | Reserved |
| 5<br>RSIM_RF_ACTIVE_OVRD_EN | RF Active Internal Override Enable<br>This bit enables the RF Active Internal Override bit. |
| 4<br>RSIM_RF_ACTIVE_OVRD | RF Active Internal Override<br>If the RSIM_RF_ACTIVE_OVRD_EN bit is set, then this bit controls the Internal version of RF Active instead of the Link Layers. The Internal version of RF Active whether overridden or not has a lower priority and can always be overridden by software using the SW_RF_ACTIVE_BIT or the TSM Spare 1 bit if it is selected using the WIFI_COEXIST_1 select bit. |
| 3-2<br>— | Reserved |
| 1<br>RADIO_CONFIGURED_SYS_RESET | Radio Configuration Bit, cleared by Radio System Reset<br>This is a bit software can write when it wants to be able to check if a Radio System Reset occurs. |
| 0<br>RADIO_CONFIGURED_POR_RESET | Radio Configuration Bit, cleared by Radio Power On Reset<br>This is a bit software can write when it wants to be able to check if a Radio Power On Reset occurs. |

## A.1.1.6 Deep Sleep Timer (DSM_TIMER)

### A.1.1.6.1 Offset

| Register | Offset |
|---|---|
| DSM_TIMER | 100h |

### A.1.1.6.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | DSM_TIMER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DSM_TIMER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.1.1.6.3 Fields

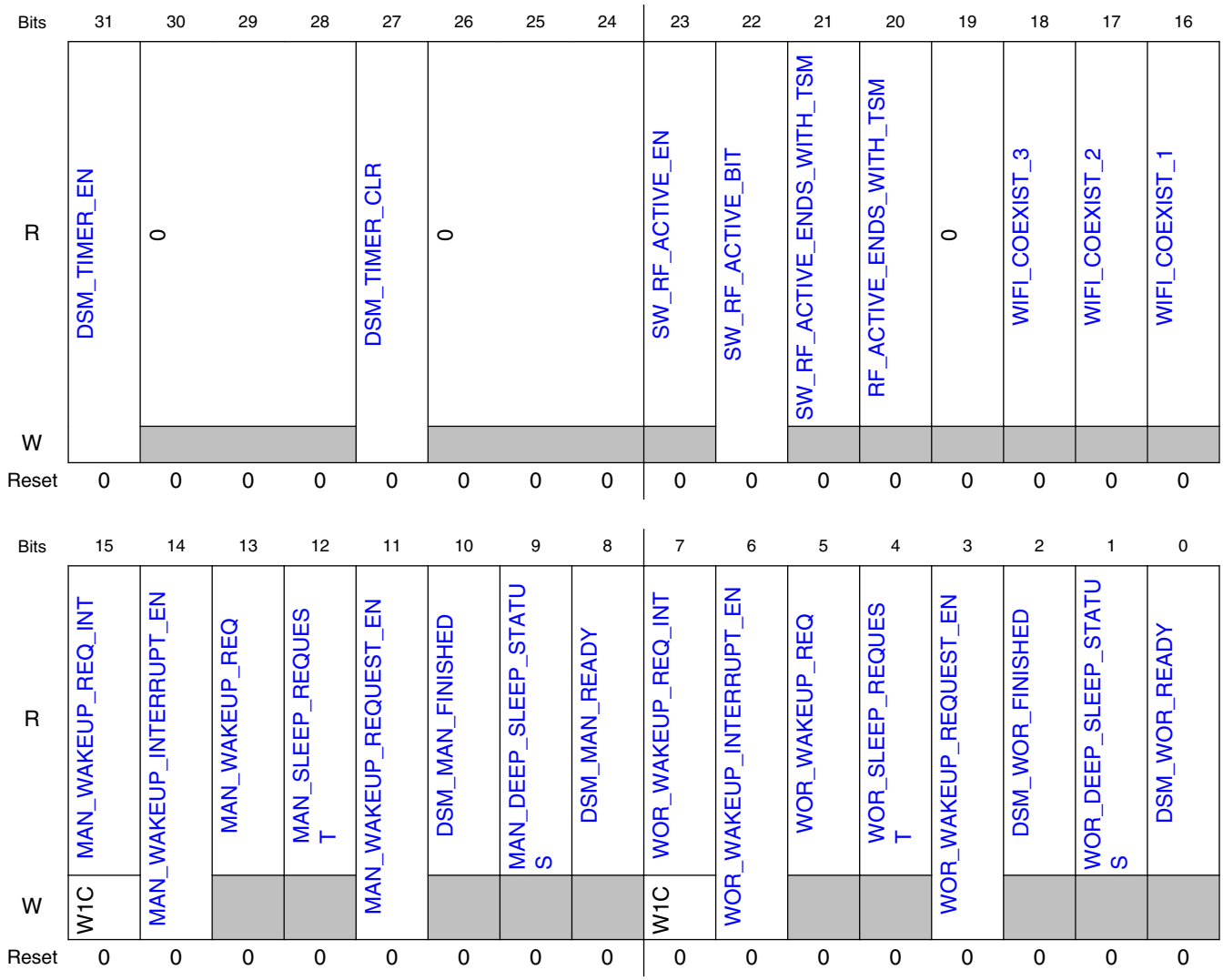
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>DSM_TIMER | Deep Sleep Mode Timer<br>This read only register shows the value of the Deep Sleep Timer. The timer counts clock cycles of the 32 kHz SoC oscillator whenever the DSM_TIMER_EN bit is set. The timer is reset to zero whenever the DSM_TIMER_CLR bit is set. |

## A.1.1.7 Deep Sleep Timer Control (DSM_CONTROL)

### A.1.1.7.1 Offset

| Register | Offset |
|---|---|
| DSM_CONTROL | 104h |

## A.1.1.7.2 Diagram




## A.1.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>DSM_TIMER_EN | Deep Sleep Mode Timer Enable<br>Whenever this bit is set the Deep Sleep Mode Timer counts clock cycles of the 32 kHz SoC oscillator. |
| 30-28<br>— | Reserved |
| 27<br>DSM_TIMER_CLR | Deep Sleep Mode Timer Clear<br>Whenever this bit is set the Deep Sleep Mode Timer is reset to zero. |
| 26-24 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 23<br>SW_RF_ACTIVE_EN | Software RF_ACTIVE Control Enable<br>If this bit is set the SW_RF_ACTIVE_BIT acts as an OR source for the RF_ACTIVE pin. |
| 22<br>SW_RF_ACTIVE_BIT | Software RF_ACTIVE Control Bit<br>If the SW_RF_ACTIVE_EN bit is set this bit acts as an OR source for the RF_ACTIVE pin. |
| 21<br>SW_RF_ACTIVE_ENDS_WITH_TSM | Software RF_ACTIVE clearing mechanism<br>If this bit is set along with the SW_RF_ACTIVE_EN bit, then the Software RF_ACTIVE OR source will clear when the TSM sequence ends. |
| 20<br>RF_ACTIVE_ENDS_WITH_TSM | RF_ACTIVE clearing mechanism<br>If this bit is set then the RSIM RF_ACTIVE OR source will clear when the TSM sequence ends. |
| 19<br>— | Reserved |
| 18<br>WIFI_COEXIST_3 | RF_EARLY_WARNING Source<br>If this bit is set the RF_EARLY_WARNING pin will be controlled by TSM Spare 3, otherwise it will be controlled by the BLE Link Layer. |
| 17<br>WIFI_COEXIST_2 | RF_STATUS Source<br>If this bit is set the RF_STATUS pin will be controlled by TSM Spare 3, otherwise it will be controlled by TSM Spare 2. |
| 16<br>WIFI_COEXIST_1 | RF_ACTIVE Source<br>If this bit is set the RF_ACTIVE pin will be controlled by TSM Spare 1, otherwise it will be controlled by the RSIM RF Active Sequence along with any override by the SW_RF_ACTIVE_BIT. |
| 15<br>MAN_WAKEUP_REQ_INT | Interrupt Flag from an MAN Deep Sleep Module Radio Wakeup<br>If the MAN_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an MAN Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| 14<br>MAN_WAKEUP_INTERRUPT_EN | MAN Deep Sleep Module Radio Wakeup Interrupt Enable<br>This bit enables an interrupt request, that is also added to the RSIM of the WOR Wakeup Request is asserted high. Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version |
| 13<br>MAN_WAKEUP_REQ | MAN Deep Sleep Module Radio Wakeup Status<br>This bit shows the status of the MAN Deep Sleep Module Request for the Radio to Wakeup. If the MAN_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| 12<br>MAN_SLEEP_REQUEST | MAN Deep Sleep Requested<br>From the MAN Sleep Enable register. This enables a match of MAN_SLEEP[23:0] to SLEEP_TMR[23:0]; when the match occurs Deep Sleep Mode is entered.<br>Warning: This bit will read as X if the Radio is not in RRun mode. |
| 11<br>MAN_WAKEUP_REQUEST_EN | Enable MAN Deep Sleep Module to initiate a Radio Wakeup<br>This bit allows the MAN Deep Sleep Module to initiate a Radio Wakeup. |
| 10 | MAN Deep Sleep Time Finished |

*Table continues on the next page...*

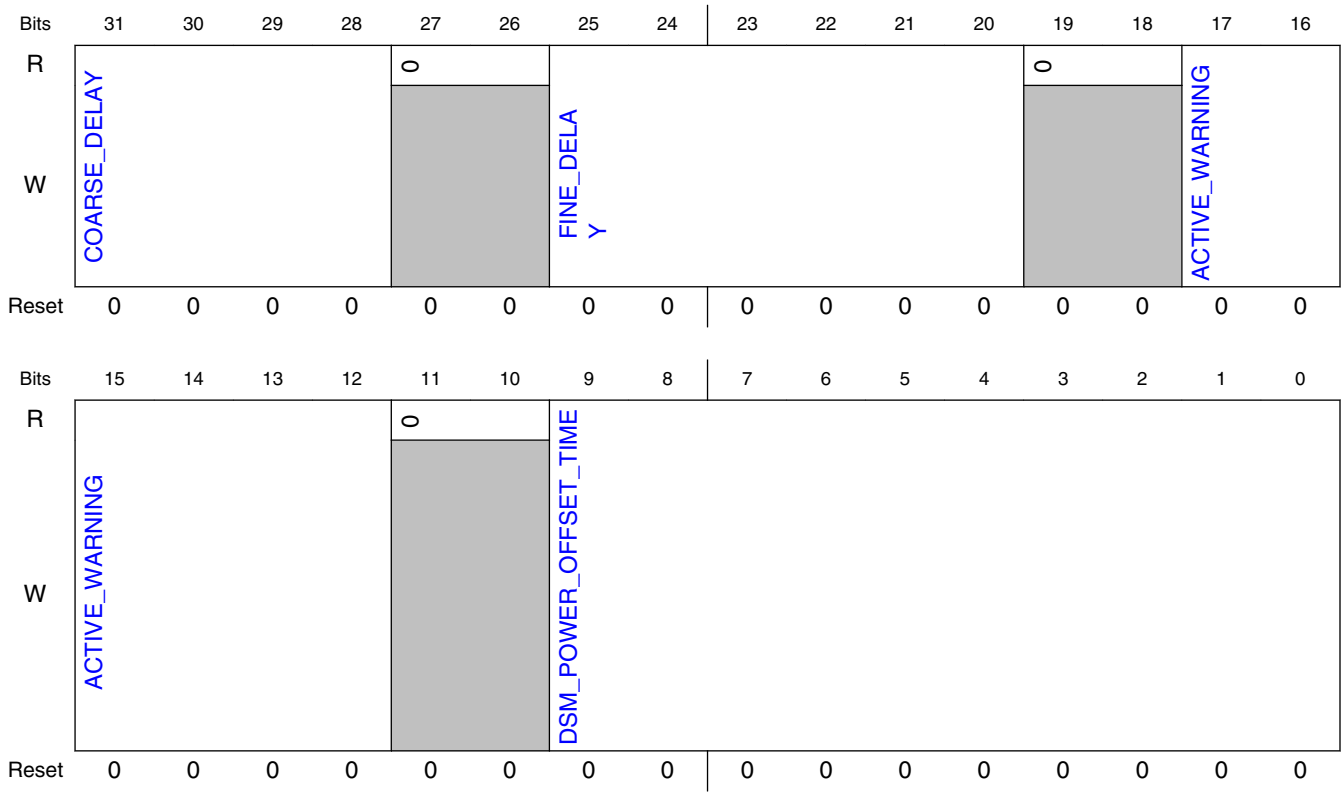
| Field | Function |
|---|---|
| DSM_MAN_FINISHED | This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| 9<br>MAN_DEEP_SLEEP_STATUS | MAN Deep Sleep Mode Status<br>This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |
| 8<br>DSM_MAN_READY | MAN Ready for Deep Sleep Mode<br>This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |
| 7<br>WOR_WAKEUP_REQ_INT | Interrupt Flag from an WOR Deep Sleep Module Radio Wakeup<br>If the WOR_WAKEUP_INTERRUPT_EN bit is set, then this flag indicates that an WOR Radio Wakeup has occurred. The flag will persist until this bit is written with a 1 to clear it. |
| 6<br>WOR_WAKEUP_INTERRUPT_EN | WOR Deep Sleep Module Radio Wakeup Interrupt Enable<br>This bit enables an interrupt request, that is also added to the RSIM Asynchronous MCU Wakeup signal, when the WAKEUP_INTERRUPT_SOURCE selected version of the WOR Wakeup Request is asserted high. |
| 5<br>WOR_WAKEUP_REQ | WOR Deep Sleep Module Radio Wakeup Status<br>This bit shows the status of the WOR Deep Sleep Module Request for the Radio to Wakeup. If the WOR_WAKEUP_REQUEST_EN bit is set, then this signal is used by the RSIM to Wakeup the Radio if the Radio is not already awake. |
| 4<br>WOR_SLEEP_REQUEST | WOR Deep Sleep Requested<br>From the WOR Sleep Enable register. This enables a match of WOR_SLEEP[23:0] to SLEEP_TMR[23:0]; when the match occurs Deep Sleep Mode is entered.<br>**warning:** This bit will read as X if the Radio is not in RRun mode. |
| 3<br>WOR_WAKEUP_REQUEST_EN | Enable WOR Deep Sleep Module to initiate a Radio Wakeup<br>This bit allows the WOR Deep Sleep Module to initiate a Radio Wakeup. |
| 2<br>DSM_WOR_FINISHED | WOR Deep Sleep Time Finished<br>This is the SLEEP_FINISHED state in the Deep Sleep Module State Machine, the state machine holds here until the Link Layer de-asserts the sleep_request. |
| 1<br>WOR_DEEP_SLEEP_STATUS | WOR Deep Sleep Mode Status<br>This is the signal from the Deep Sleep Module State Machine telling the Link Layer to enter and exit Deep Sleep Mode. If this bit is set then the Link Layer is in Deep Sleep Mode. |
| 0<br>DSM_WOR_READY | WOR Ready for Deep Sleep Mode<br>This is the SLEEP_READY state in the Deep Sleep Module State Machine, the state machine holds here until the sleep time is reached on the 32 kHz clock counter (Deep Sleep Timer). |

## A.1.1.8 Deep Sleep Wakeup Sequence (DSM_WAKEUP)

### A.1.1.8.1 Offset

| Register | Offset |
|---|---|
| DSM_WAKEUP | 108h |

### A.1.1.8.2 Diagram




### A.1.1.8.3 Fields

| Field | Function |
|---|---|
| 31-28<br>COARSE_DELAY | Deep Sleep Wakeup Coarse Delay Time<br>This is the first delay after the beginning of the Power-On sequence, the range is 0-15 and the resulting delay is 1.953 ms x COARSE_DELAY + 61 us, with a minimum delay of 61 us. |
| 27-26<br>— | Reserved |
| 25-20<br>FINE_DELAY | Deep Sleep Wakeup Fine Delay Time<br>This is the second delay after the beginning of the Power-On sequence, the range is 0-63 and the resulting delay is 122.07 us x FINE_DELAY + 61 us, with a minimum delay of 61 us. This is when the RF OSC is enabled and so the Radio Analog Power must be stable at this milestone. |
| 19-18<br>— | Reserved |
| 17-12<br>ACTIVE_WARNING | Deep Sleep Wakeup RF Active Warning Time<br>This is the delay after RF OSC has been enabled, the range is 0-63 and the resulting delay is 122.07 us x ACTIVE_WARNING + 30.5 us. This is typically when the RF Osc should be stable. |
| 11-10 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 9-0<br>DSM_POWER_OFFSET_TIME | Deep Sleep Wakeup Power Offset Time<br>This register is the calculated Radio Power-On Sequence Time. |

## A.1.1.9 WOR Deep Sleep Duration (WOR_DURATION)

### A.1.1.9.1 Offset

| Register | Offset |
|---|---|
| WOR_DURATION | 10Ch |

### A.1.1.9.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | WOR_DSM_DURATION | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WOR_DSM_DURATION | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.1.1.9.3 Fields

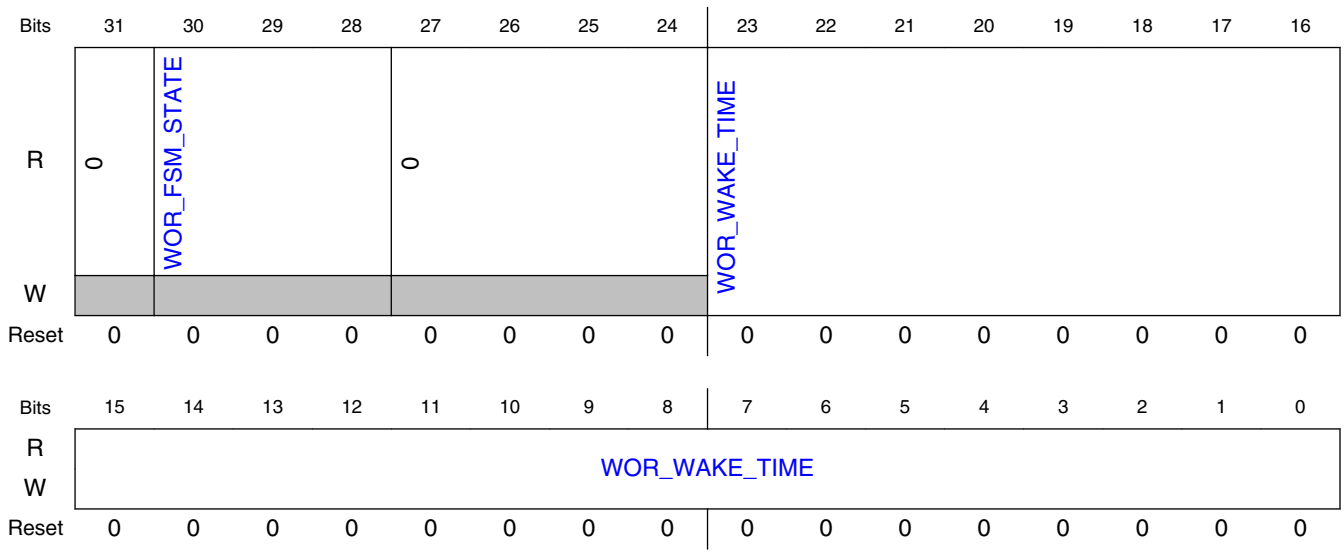
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>WOR_DSM_DURATION | WOR Deep Sleep Time Elapsed<br>The RSIM calculates the duration of the WOR deep sleep time and presents it here. |

# A.1.1.10 WOR Deep Sleep Wake Time (WOR_WAKE)

## A.1.1.10.1 Offset

| Register | Offset |
|---|---|
| WOR_WAKE | 110h |

## A.1.1.10.2 Diagram




## A.1.1.10.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>WOR_FSM_STATE | WOR Deep Sleep State Machine State<br>This is the current State that the WOR Deep Sleep Finite State Machine is in. |
| 27-24<br>— | Reserved |
| 23-0<br>WOR_WAKE_TIME | WOR Deep Sleep Module Wake Time<br>Hardware determines the future time at which to exit Deep Sleep Mode by reading the DSM_TIMER and computing the number of 32 kHz clock cycles needed to match the desired DSM Duration. The calculated value of this Wake Time is shown in this register. |

## A.1.1.11 MAN Deep Sleep Time (MAN_SLEEP)

### A.1.1.11.1 Offset

| Register | Offset |
|---|---|
| MAN_SLEEP | 11Ch |

### A.1.1.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MAN_SLEEP_TIME | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAN_SLEEP_TIME | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.1.1.11.3 Fields

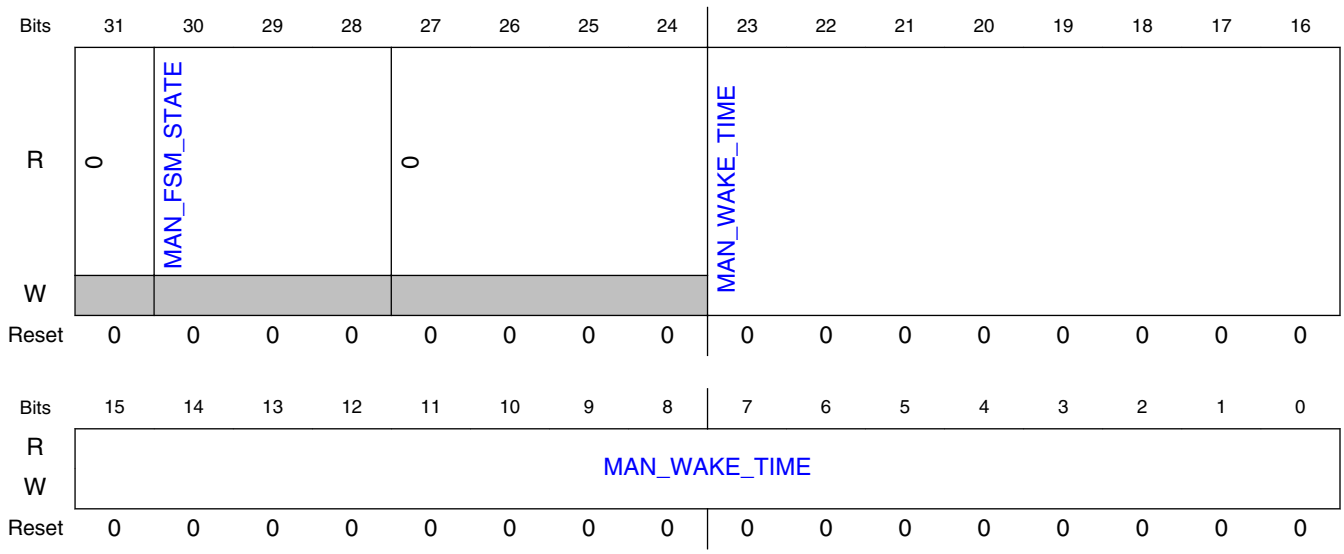
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>MAN_SLEEP_TIME | MAN Deep Sleep Module Sleep Time<br>Software determines the future time at which it would like to enter Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM start-time, and writing that value into this register. The value programmed into this register should be no fewer than 4 clocks greater (in the future) than the current time as read from DSM_TIMER. |

## A.1.1.12 MAN Deep Sleep Wake Time (MAN_WAKE)

### A.1.1.12.1 Offset

| Register | Offset |
|---|---|
| MAN_WAKE | 120h |

### A.1.1.12.2 Diagram




### A.1.1.12.3 Fields

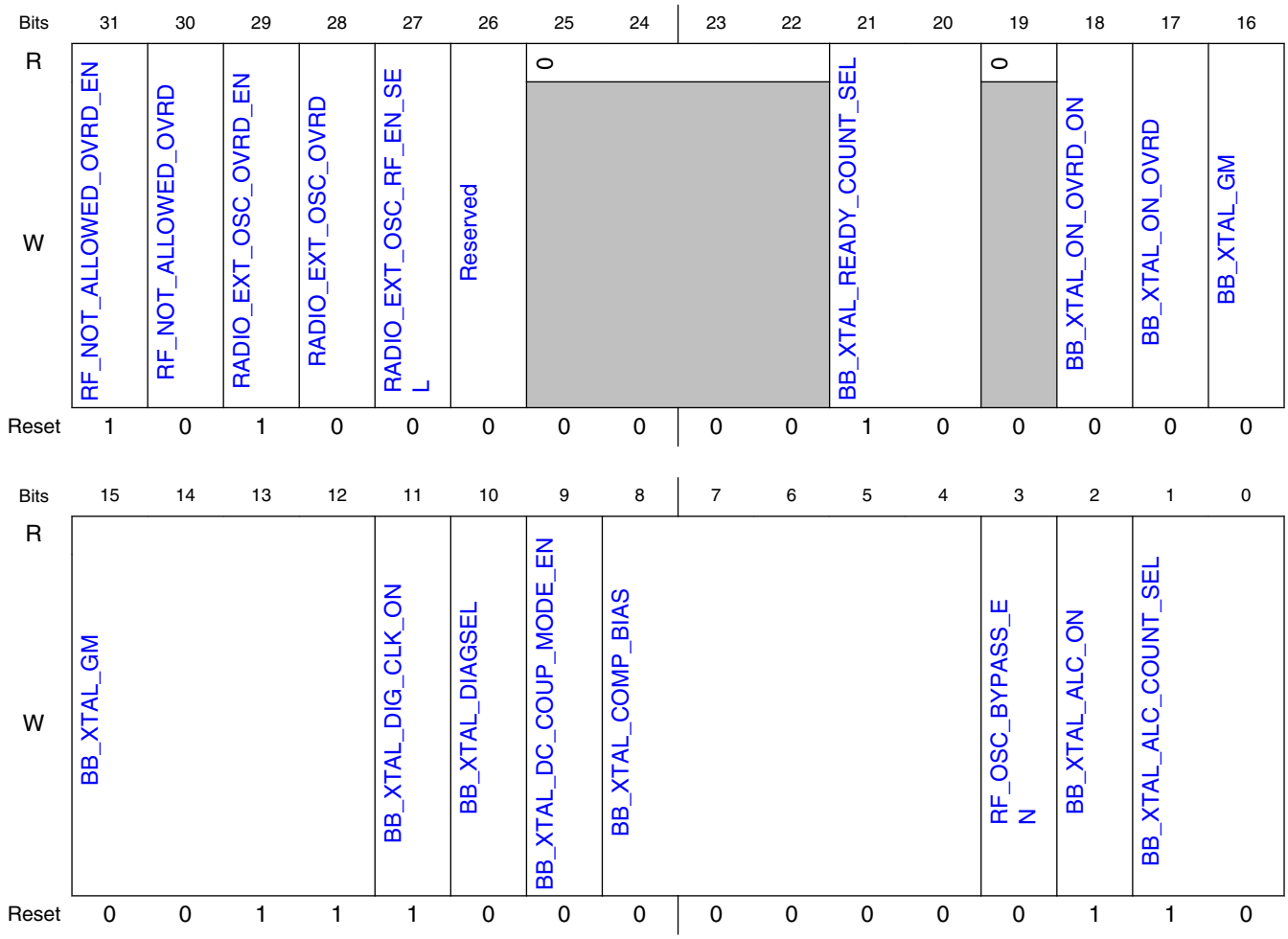
| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>MAN_FSM_STATE | MAN Deep Sleep State Machine State<br>This is the current State that the MAN Deep Sleep Finite State Machine is in. |
| 27-24<br>— | Reserved |
| 23-0<br>MAN_WAKE_TIME | MAN Deep Sleep Module Wake Time<br>Software determines the future time at which it would like to exit Deep Sleep Mode by reading the DSM_TIMER, computing the number of 32 kHz clock cycles remaining until the desired DSM exit time, and writing that value into this register. |

## A.1.1.13 Radio Oscillator Control (RF_OSC_CTRL)

### A.1.1.13.1 Offset

| Register | Offset |
|---|---|
| RF_OSC_CTRL | 124h |

## A.1.1.13.2 Diagram




## A.1.1.13.3 Fields

| Field | Function |
|---|---|
| 31<br>RF_NOT_ALLOWED_OVRD_EN | RF Not Allowed Override Enable<br>This bit enables the RF Not Allowed Override bit. |
| 30<br>RF_NOT_ALLOWED_OVRD | RF Not Allowed Override<br>If the RF_NOT_ALLOWED_OVRD_EN bit is set, then this bit controls the RF Not Allowed functionality instead of the SoC pin. |
| 29<br>RADIO_EXT_OSC_OVRD_EN | Radio External Request for RF OSC Override Enable<br>This bit enables the Radio External RF OSC Override bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 28<br>RADIO_EXT_OSC_OVRD | Radio External Request for RF OSC Override<br>If the RADIO_EXT_OSC_OVRD_EN bit is set, then this bit controls that External RF OSC Request functionality. |
| 27<br>RADIO_EXT_OSC_RF_EN_SEL | Radio External Request for RF OSC Select<br>The Radio has two pins which may be controlled by external pins in the SoC in such a way as to allow an external device, such as another radio, to enable this Radio's RF Oscillator. This bit selects which of the two pins has control of that external request. |
| 26<br>— | Reserved |
| 25-22<br>— | Reserved |
| 21-20<br>BB_XTAL_READY_COUNT_SEL | rmap_bb_xtal_ready_count_sel_hv[1:0]<br>Program counter for xtal ready signal<br>Sets up count value for XO startup time.<br>00b - 1024 counts (32 us @ 32 MHz)<br>01b - 2048 (64 us @ 32 MHz)<br>10b - 4096 (128 us @ 32 MHz)<br>11b - 8192 (256 us @ 32 MHz) |
| 19<br>— | Reserved |
| 18<br>BB_XTAL_ON_OVRD_ON | rmap_bb_xtal_on_ovrd_on_hv<br>Enable override XO enable bit<br>Enable selector:<br>0b - rfctrl_bb_xtal_on_hv is asserted<br>1b - rfctrl_bb_xtal_on_ovrd_hv is asserted |
| 17<br>BB_XTAL_ON_OVRD | rmap_bb_xtal_on_ovrd_hv<br>Override XO enable |
| 16-12<br>BB_XTAL_GM | rmap_bb_xtal_gm_hv[4:0]<br>Amplifier current bumps, bit [4] not used. Current values assume ALC is off<br>0 - Min setting: 8 current sources on (TT27:60uA)<br>...<br>15 - Max setting: 128 current sources on (TT27:960uA) |
| 11<br>BB_XTAL_DIG_CLK_ON | rmap_bb_xtal_dig_clk_on_hv<br>Enable digital clk output |
| 10<br>BB_XTAL_DIAGSEL | rmap_bb_xtal_diagsel_hv<br>Enable diagnostics for XO block |
| 9<br>BB_XTAL_DC_COUP_MODE_EN | rmap_bb_xtal_dc_coup_mode_en_hv<br>This bit enables the external dc coupled mode. This bit powers down the XO amplifier to enable DC coupled input. |

*Table continues on the next page...*

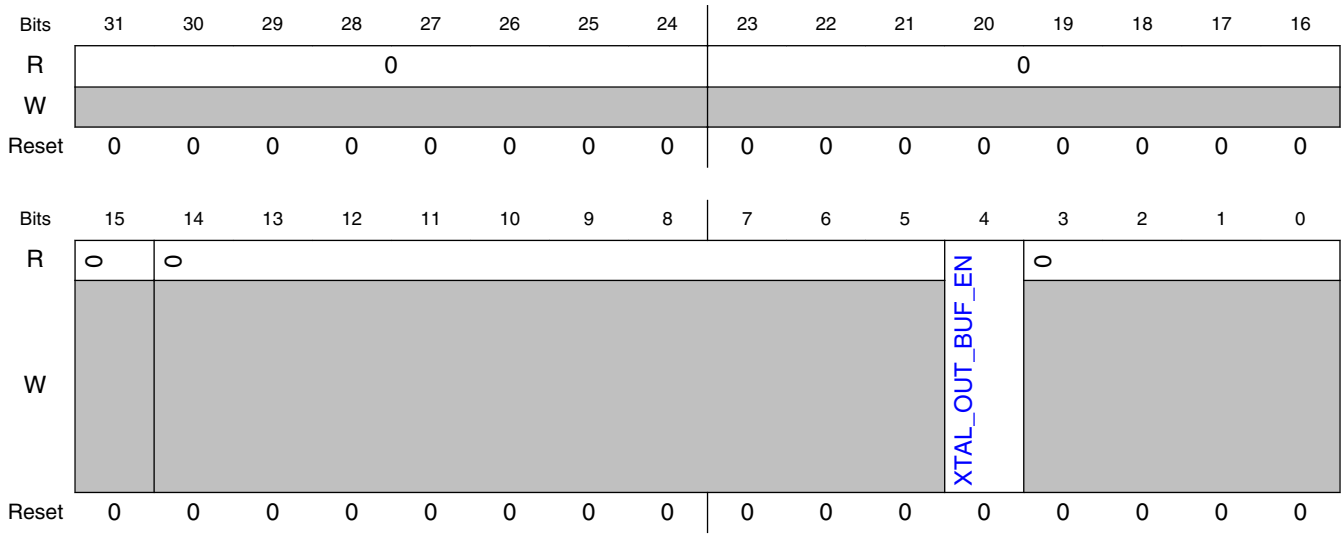
| Field | Function |
|---|---|
| 8-4<br>BB_XTAL_COMP_BIAS | rmap_bb_xtal_comp_bias_hv[4:0]<br>Not used. |
| 3<br>RF_OSC_BYPASS_EN | RF Ref Osc Bypass Enable<br>This bit engages the RF Ref Osc analog bypass circuit if the RF Ref Osc is enabled. When the RF Ref Osc is in bypass mode it passes the RF EXTAL clock as the RF Ref Osc clock. Note that the RF Ref Osc Ready signal functions normally in RF OSC Bypass mode, unless overridden with the RF_OSC_READY_OVRD_EN bit. |
| 2<br>BB_XTAL_ALC_ON | rmap_bb_xtal_alc_on_hv<br>Enable ALC |
| 1-0<br>BB_XTAL_ALC_COUNT_SEL | rmap_bb_xtal_alc_count_sel_hv[1:0]<br>Program counter for alc ready signal<br>Sets up count value for fastcharge to turn off:<br>00b - 2048 (64 us @ 32 MHz)<br>01b - 4096 (128 us @ 32 MHz)<br>10b - 8192 (256 us @ 32 MHz)<br>11b - 16384 (512 us @ 32 MHz) |

## A.1.1.14 Radio Analog Test Registers (ANA_TEST)

### A.1.1.14.1 Offset

| Register | Offset |
|---|---|
| ANA_TEST | 128h |

### A.1.1.14.2 Diagram




### A.1.1.14.3 Fields

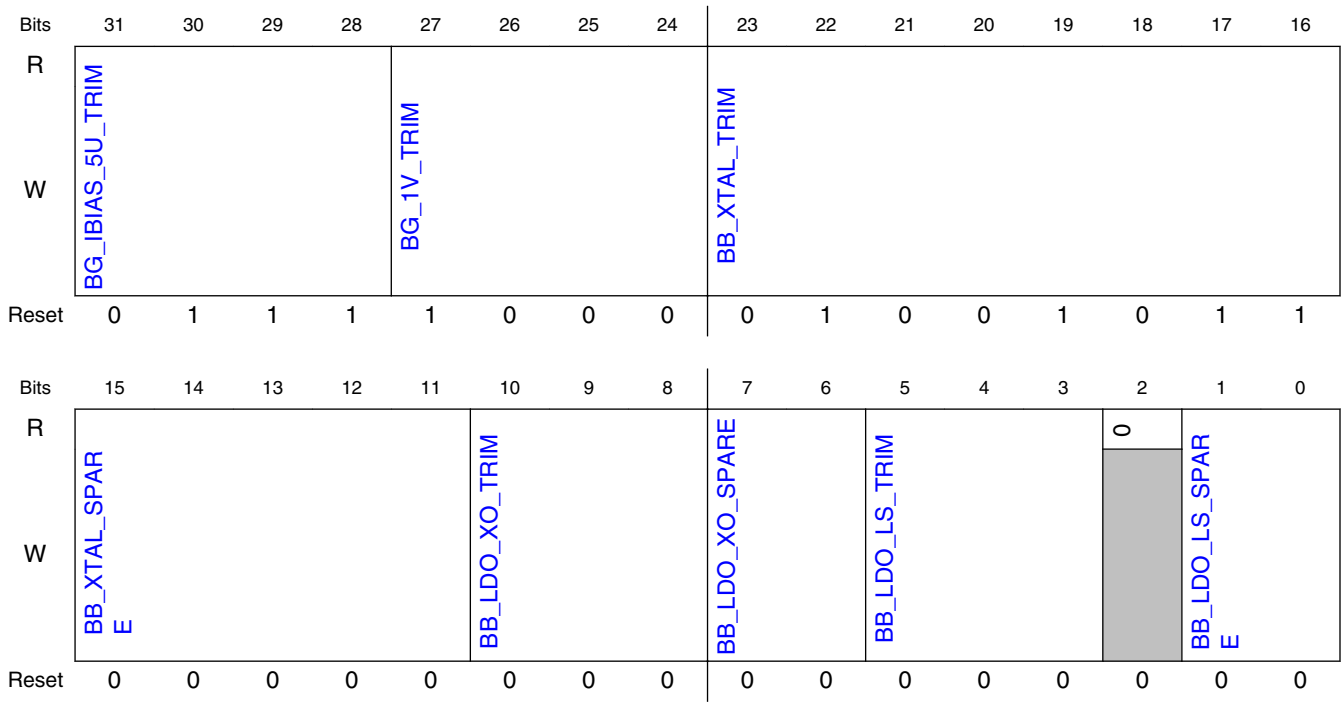
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-15<br>— | Reserved |
| 14-5<br>— | Reserved. |
| 4<br>XTAL_OUT_BUF_EN | XTAL Output Buffer Enable<br>The RF Analog Oscillator XTAL_OUT buffer for the XTAL_OUT SoC Pin is enabled when this bit is set. |
| 3-0<br>— | Reserved. |

## A.1.1.15 Radio Analog Trim Registers (ANA_TRIM)

### A.1.1.15.1 Offset

| Register | Offset |
|---|---|
| ANA_TRIM | 12Ch |

## A.1.1.15.2 Diagram




## A.1.1.15.3 Fields

| Field | Function |
|---|---|
| 31-28<br>BG_IBIAS_5U_TRIM | rmap_bg_ibias_5u_trim_hv[3:0]<br>5uA current trim bits. Default setting is 0111<br>Trim bits for VBG output current<br>0000b - 3.55 uA<br>0001b - 3.73 uA<br>0010b - 4.04 uA<br>0011b - 4.22 uA<br>0100b - 4.39 uA<br>0101b - 4.57 uA<br>0110b - 4.89 uA<br>0111b - 5.06 (Default)<br>1000b - 5.23 uA<br>1001b - 5.41 uA<br>1010b - 5.72 uA<br>1011b - 5.9 uA<br>1100b - 6.07 uA<br>1101b - 6.25 uA<br>1110b - 6.56 uA<br>1111b - 6.74 uA |
| 27-24<br>BG_1V_TRIM | rmap_bg_1v_trim_hv[3:0]<br>Trim bits for VBG output voltage |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Trim bits for VBG output voltage<br>0000b - 954.14 mV<br>0001b - 959.26 mV<br>0010b - 964.38 mV<br>0011b - 969.5 mV<br>0100b - 974.6 mV<br>0101b - 979.7 mV<br>0110b - 984.8 mV<br>0111b - 989.9 mV<br>1000b - 995 mV (Default)<br>1001b - 1 V<br>1010b - 1.005 V<br>1011b - 1.01 V<br>1100b - 1.015 V<br>1101b - 1.02 V<br>1110b - 1.025 V<br>1111b - 1.031 V |
| 23-16<br>BB_XTAL_TRIM | rmap_bb_xtal_trim_hv[7:0]<br>Bump XO load capacitor<br>Load capacitor bumps. bit [7] not used.<br>0 - Min C1: 5.7pF. Min C2: 7.1pF<br>...<br>127 - Max C1: 22.6pF. Max C2: 28.2pF |
| 15-11<br>BB_XTAL_SPARE | rmap_bb_xtal_spare_hv[4:0]<br>Bit 4: Override RF Analog XTAL Ready signal ; 0 - No override, BB_XTAL_READY_COUNT_SEL is used ; 1- Force RF Analog XTAL Ready = 1. Note that this RF Analog signal can be also be overridden using the RSIM RF_OSC_READY_OVRD bits.<br>Bit 3: XTAL_OUT output Polarity invert bit; 0 - No Polarity inversion; 1 - Polarity inverted<br>Bit 2: This bit selects the clock to be output on the XTAL_OUT SoC pin. The default is to select the XTAL as the clock source. If this bit is written to a 1 then the AuxPLL output will be selected instead.<br>Bits 1:0 are unused. |
| 10-8<br>BB_LDO_XO_TRIM | rmap_bb_ldo_xo_trim_hv[2:0]<br>Trim settings for LDO.<br>Trim settings for LDO.<br>000b - 1.20 V (Default)<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 7-6<br>BB_LDO_XO_SPARE | rmap_bb_ldo_xo_spare_hv[1:0]<br>Spare bits for LDO, not used. |
| 5-3<br>BB_LDO_LS_TRIM | rmap_bb_ldo_ls_trim_hv[2:0]<br>Trim settings for LDO. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Trim settings for LDO.<br>000b - 1.20 V (Default)<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 2<br>— | Reserved |
| 1-0<br>BB_LDO_LS_SPARE | rmap_bb_ldo_ls_spare_hv[1:0]<br>Spare bits. Not used so far. |

# A.2 Transceiver Memory Map and Register Definition

## A.2.1 XCVR_TX_DIG register descriptions

### A.2.1.1 XCVR_TX_DIG_ADDR Memory map

XCVR_TX_DIG base address: 4103_0200h

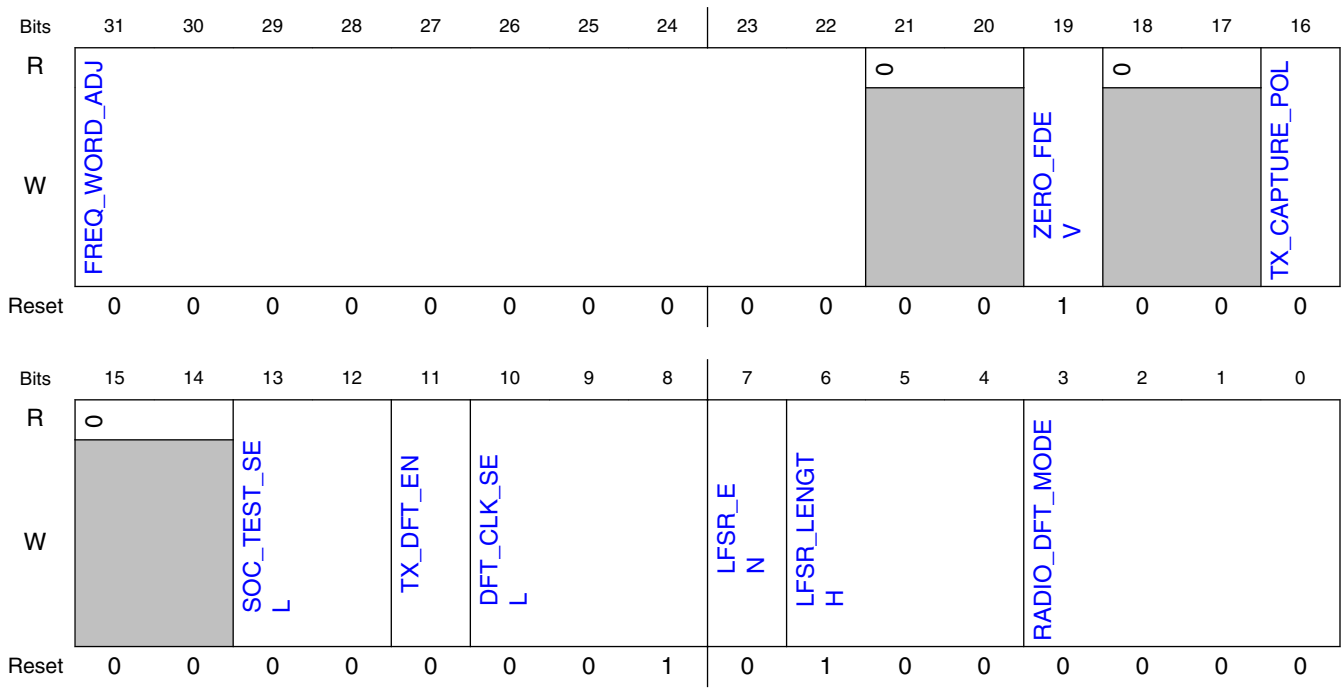
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TX Digital Control (CTRL) | 32 | RW | 0008_0140h |
| 4h | TX Data Padding (DATA_PADDING) | 32 | RW | 7FFF_55AAh |
| 8h | TX GFSK Modulator Control (GFSK_CTRL) | 32 | RW | 0001_4000h |
| Ch | TX GFSK Filter Coefficients 2 (GFSK_COEFF2) | 32 | RW | C063_0401h |
| 10h | TX GFSK Filter Coefficients 1 (GFSK_COEFF1) | 32 | RW | BB29_960Dh |
| 14h | TX FSK Modulation Levels (FSK_SCALE) | 32 | RW | 0800_1800h |
| 18h | TX DFT Modulation Pattern (DFT_PATTERN) | 32 | RW | 0000_0000h |

### A.2.1.2 TX Digital Control (CTRL)

### A.2.1.2.1 Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

### A.2.1.2.2 Diagram




### A.2.1.2.3 Fields

| Field | Function |
|---|---|
| 31-22<br>FREQ_WORD_ADJ | Frequency Word Adjustment<br>This register is a signed 9 bit number that is added to the TX Digital output before it is presented to the PLL as the baseband frequency word. This allows the baseband frequency word to be adjusted, or skewed, by a range of -512 to +511.<br>This frequency adjustment is applied to the final modulation word from any source (GFSK, FSK, DFT) and there is no protection from a math overflow. So the baseband frequency word range of -4096 to 4095 must be considered when adding this frequency word adjustment. |
| 21-20<br>— | Reserved |
| 19<br>ZERO_FDEV | On-Air at Zero FDev<br>If this bit is set, then in Gaussian Modulation modes the modulation will start at the Carrier Frequency instead of at the negative modulation value. |
| 18-17 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 16<br>TX_CAPTURE_POL | Polarity of the Input Data for the Transmitter<br>If this bit is set, the TX data presented to the Transmitter will be inverted before it is processed. |
| 15-14<br>— | Reserved |
| 13-12<br>SOC_TEST_SEL | Radio Clock Selector for SoC RF Clock Tests<br>This register selects the Radio clock source for the SoC Clock Tests Frequency Measurement.<br>00b - No Clock Selected<br>01b - PLL Sigma Delta Clock, divided by 2<br>10b - Auxiliary PLL Clock, divided by 2<br>11b - RF Ref Osc clock, divided by 2 |
| 11<br>TX_DFT_EN | DFT Modulation Enable<br>If the Radio is in a DFT Pattern Register mode, then this bit is used to turn on and off the modulation that is shifted out from the pattern register. |
| 10-8<br>DFT_CLK_SEL | DFT Clock Selection<br>This register selects the frequency of the DFT clock that is used to shift out the DFT Modulation Pattern in DFT Pattern Register modes, and the same frequency is also used to clock the LFSR and generate the pseudo-random modulation in DFT LFSR modes.<br>000b - 62.5 kHz<br>001b - 125 kHz<br>010b - 250 kHz<br>011b - 500 kHz<br>100b - 1 MHz<br>101b - 2 MHz<br>110b - 4 MHz<br>111b - RF OSC Clock |
| 7<br>LFSR_EN | LFSR Enable<br>If the Radio is in a DFT LFSR mode, then this bit is used to turn on and off the LFSR that is used to generate the modulation.<br>Note that the LFSR is clocked at the DFT Clock frequency. |
| 6-4<br>LFSR_LENGTH | LFSR Length<br>This register selects the length of the DFT LFSR and the associated LFSR Tap Mask.<br>The Mask is in the form of [MSB...LSB]<br>000b - LFSR 9, tap mask 100010000<br>001b - LFSR 10, tap mask 1001000000<br>010b - LFSR 11, tap mask 11101000000<br>011b - LFSR 13, tap mask 1101100000000<br>100b - LFSR 15, tap mask 111010000000000<br>101b - LFSR 17, tap mask 11110000000000000<br>110b - Reserved<br>111b - Reserved |
| 3-0<br>RADIO_DFT_MODE | Radio DFT Modes<br>This register selects the Radio DFT mode as described below.<br>In addition to setting the Radio DFT mode, the DFT LFSR needs to be configured, and the Radio Protocol needs to be chosen. |

| Field | Function |
|---|---|
| | For LFSR modes the LFSR_EN needs to be set to turn on the LFSR.<br>0000b - Normal Radio Operation, DFT not engaged.<br>0001b - Carrier Frequency Only<br>0010b - Pattern Register GFSK<br>0011b - LFSR GFSK<br>0100b - Pattern Register FSK<br>0101b - LFSR FSK<br>0110b - Pattern Register O-QPSK<br>0111b - LFSR O-QPSK<br>1000b - LFSR 802.15.4 Symbols<br>1001b - PLL Modulation from RAM<br>1010b - PLL Coarse Tune BIST<br>1011b - PLL Frequency Synthesizer BIST<br>1100b - High Port DAC BIST<br>1101b - VCO Frequency Meter<br>1110b - Reserved<br>1111b - Reserved |

## A.2.1.3 TX Data Padding (DATA_PADDING)

### A.2.1.3.1 Offset

| Register | Offset |
|---|---|
| DATA_PADDING | 4h |

### A.2.1.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LRM | DFT_LFSR_OUT | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_PADDING_PAT_1 | | | | | | | | DATA_PADDING_PAT_0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

#### A.2.1.3.3 Fields

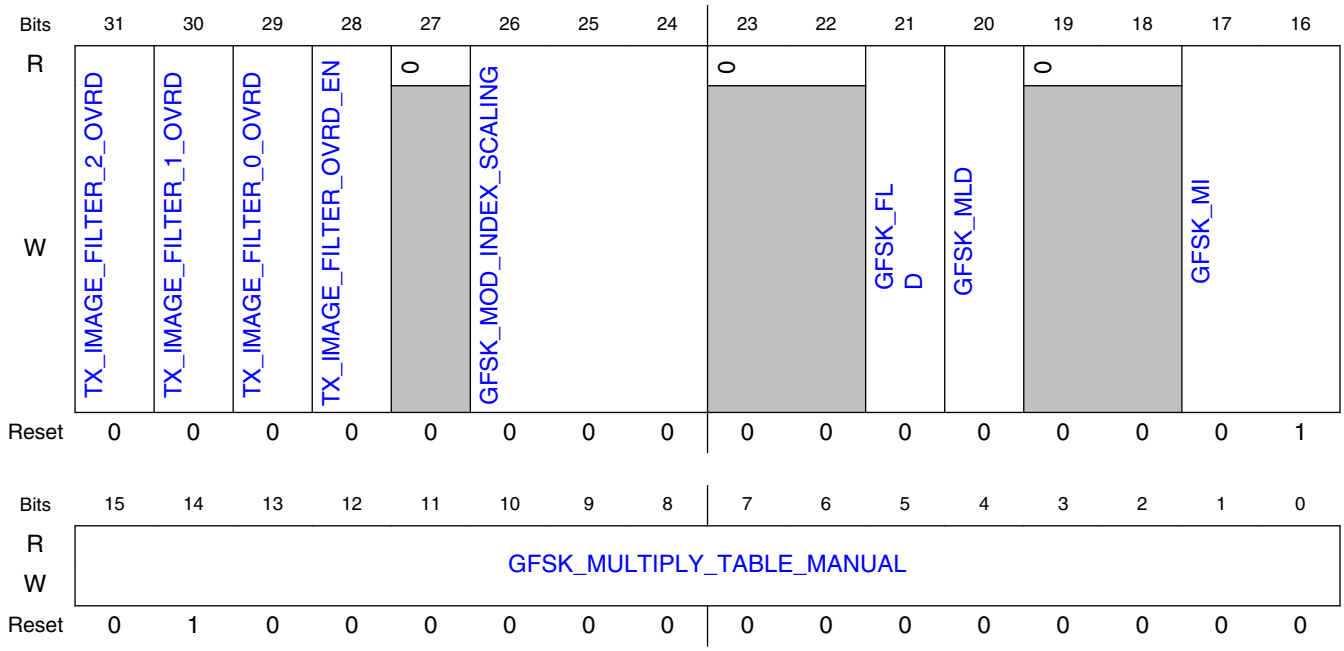
| Field | Function |
|---|---|
| 31<br>LRM | LFSR Reset Mask<br>When this bit is set the DFT LFSR will not be reset when LFSR_EN is cleared and will instead continue to repeat its sequence as defined by the DFT_LFSR_LEN bits when LFSR_EN is next set. When this bit is cleared the DFT LFSR will reset every time LFSR_EN is cleared. |
| 30-16<br>DFT_LFSR_OUT | LFSR Output<br>This register can be read to observe the current value of the DFT LFSR, only bits [14:0] are available. |
| 15-8<br>DATA_PADDING_PAT_1 | Data Padding Pattern 1<br>These bits are used for Data Padding when the first bit of the Preamble is 1; the LSB is the first bit shifted out as padding. |
| 7-0<br>DATA_PADDING_PAT_0 | Data Padding Pattern 0<br>These bits are used for Data Padding when the first bit of the Preamble is 0; the LSB is the first bit shifted out as padding. |

### A.2.1.4 TX GFSK Modulator Control (GFSK_CTRL)

#### A.2.1.4.1 Offset

| Register | Offset |
|---|---|
| GFSK_CTRL | 8h |

## A.2.1.4.2 Diagram




## A.2.1.4.3 Fields

| Field | Function |
|---|---|
| 31<br>TX_IMAGE_FILTER_2_OVRD | TX Image Filter 2 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 2 |
| 30<br>TX_IMAGE_FILTER_1_OVRD | TX Image Filter 1 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 1 |
| 29<br>TX_IMAGE_FILTER_0_OVRD | TX Image Filter 0 Override Control<br>If the TX_IMAGE_FILTER_OVRD_EN bit is set, then this bit turns on (=1) and off (=0) the Transmit Image Filter 0 |
| 28<br>TX_IMAGE_FILTER_OVRD_EN | TX Image Filter Override Enable<br>This bit enables the TX Image Filter Override Control bits. |
| 27<br>— | Reserved |
| 26-24<br>GFSK_MOD_INDEX_SCALING | GFSK Modulation Index Scaling Factor<br>This register selects the amount to scale the Transmitter Modulation Word in GFSK Protocols.<br>Transmit modulation scaling adds the capability to scale the GFSK modulated output up or down by a factor of 1/32, 1/16 or 1/8. This is equivalent to having some additional programmability of the modulation index.<br>000b - 1<br>001b - 1 + 1/32 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 010b - 1 + 1/16<br>011b - 1 + 1/8<br>100b - 1 - 1/32<br>101b - 1 - 1/16<br>110b - 1 - 1/8<br>111b - Reserved |
| 23-22<br>— | Reserved |
| 21<br>GFSK_FLD | Disable GFSK Filter Lookup Table<br>If this bit is set, the internal GFSK filter coefficients that are normally derived from a lookup table based on the reference clock frequency, are disabled, and the coefficients are instead derived from the GFSK_FILTER_COEFF_MANUAL1 and GFSK_FILTER_COEFF_MANUAL2 registers. |
| 20<br>GFSK_MLD | Disable GFSK Multiply Lookup Table<br>If this bit is set, the GFSK Multiply Lookup table is disabled and GFSK_MULTIPLY_TABLE_MANUAL is used instead. |
| 19-18<br>— | Reserved |
| 17-16<br>GFSK_MI | GFSK Modulation Index<br>This register selects the GFSK Modulation Index which, together with the GFSK Symbol Rate, determines the Peak Modulation frequency.<br>The formula used for the Peak Modulation is { Symbol Rate / (2 x 1/Modulation Index) }<br>00b - 0.32<br>01b - 0.50<br>10b - 0.70<br>11b - 1.00 |
| 15-0<br>GFSK_MULTIPLY_TABLE_MANUAL | Manual GFSK Multiply Lookup Table Value<br>The GFSK Modulator Multiplier uses a lookup table to select the multiplicand representing the { Frequency Deviation divided by the Low Port Sigma Delta LSB resolution in Hz } for the Modulation requested based on the Modulation Index, the Symbol Rate, and the Reference Clock Frequency.<br>The lookup table value is overridden by this register if GFSK_MLD is set, and these bits should then contain a number that represents { FDev/SD_LSB integer[11:0] + FDev/SD_LSB fraction[3:0] } |

## A.2.1.5 TX GFSK Filter Coefficients 2 (GFSK_COEFF2)

### A.2.1.5.1 Offset

| Register | Offset |
|---|---|
| GFSK_COEFF2 | Ch |

### A.2.1.5.2 Function

The two registers TX_GFSK_COEFF1 and TX_GFSK_COEFF2 form a 64-bit little endian register that is memory mapped internally as shown to override the GFSK Filter Coefficients.

In 64-bit they combine as {TX_GFSK_COEFF2[31:0],TX_GFSK_COEFF1[31:0]}

The resulting 64-bit value is the GFSK Manual Filter Coefficient [63:0]

### A.2.1.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL2 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### A.2.1.5.4 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-0<br><br>GFSK_FILTER_COEFF_MANUAL2</td><td>GFSK Manual Filter Coefficients[63:32]<br><br>If GFSK_FLD is set, the GFSK Filter Coefficients are overridden using the bits as described below, and since the filter is symmetrical each coefficient is used twice.<br><br>
<table>
<tr><th>Bits</th><th>Filter Coefficients</th></tr>
<tr><td>[36:32]</td><td>Filter coeff 0 and 15</td></tr>
<tr><td>[45:40]</td><td>Filter coeff 1 and 14</td></tr>
<tr><td>[55:48]</td><td>Filter coeff 4 and 11</td></tr>
<tr><td>[63:56]</td><td>Filter coeff 5 and 10</td></tr>
</table>
</td></tr>
</table>

## A.2.1.6 TX GFSK Filter Coefficients 1 (GFSK_COEFF1)

### A.2.1.6.1 Offset

| Register | Offset |
|---|---|
| GFSK_COEFF1 | 10h |

### A.2.1.6.2 Function

The two registers TX_GFSK_COEFF1 and TX_GFSK_COEFF2 form a 64-bit little endian register that is memory mapped internally as shown to override the GFSK Filter Coefficients.

In 64-bit they combine as {TX_GFSK_COEFF2[31:0],TX_GFSK_COEFF1[31:0]}

The resulting 64-bit value is the GFSK Manual Filter Coefficient [63:0]

### A.2.1.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL1 | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GFSK_FILTER_COEFF_MANUAL1 | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

### A.2.1.6.4 Fields

| Field | Function |
|---|---|
| 31-0<br>GFSK_FILTER_COEFF_MANUAL1 | GFSK Manual Filter Coefficient [31:0]<br>If GFSK_FLD is set, the GFSK Filter Coefficients are overridden using the bits as described below, and since the filter is symmetrical each coefficient is used twice. |

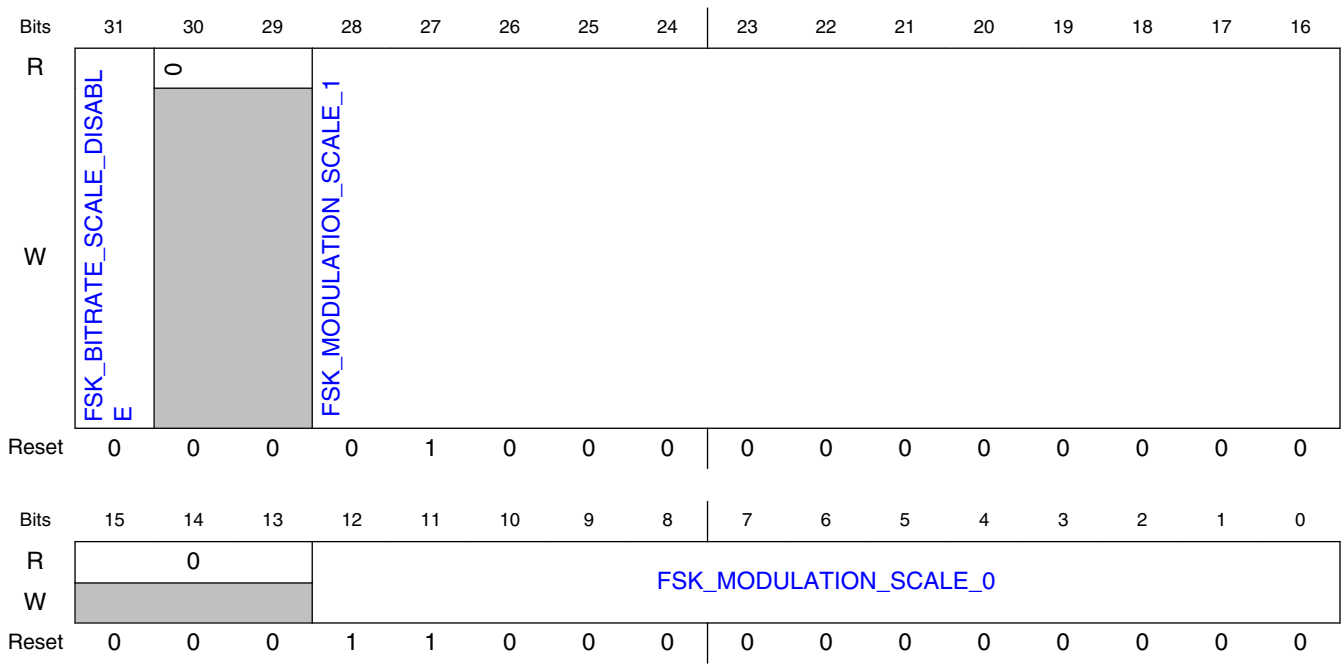
| Bits | 000 Filter Coefficients |
|---|---|
| [6:0] | Filter coeff 2 and 13 |
| [15:7] | Filter coeff 6 and 9 |
| [22:16] | Filter coeff 3 and 12 |
| [31:23] | Filter coeff 7 and 8 |

# A.2.1.7 TX FSK Modulation Levels (FSK_SCALE)

## A.2.1.7.1 Offset

| Register | Offset |
|---|---|
| FSK_SCALE | 14h |

## A.2.1.7.2 Diagram




## A.2.1.7.3 Fields

| Field | Function |
|---|---|
| 31<br>FSK_BITRATE_SCALE_DISABLE | FSK Bitrate Scaling Disable<br>If this bit is set, the FSK Modulation values will not be scaled according to the bitrate setting. |
| 30-29<br>— | Reserved |
| 28-16<br>FSK_MODULATION_SCALE_1 | FSK Modulation Scale for a data 1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | This register is used to provide the modulation level for a data 1 in FSK Protocols. The signed 12 bit value that is represented by this register is presented to the PLL as the Baseband Frequency Word. The frequency result for this Modulation Scale will be the value of this register times the PLL Low Port Modulator Frequency Resolution; a typical value for the Frequency Resolution is 244.14 Hz.<br>The range of signed values that this register supports is -4096 to 4095 |
| 15-13<br>— | Reserved |
| 12-0<br>FSK_MODULATION_SCALE_0 | FSK Modulation Scale for a data 0<br>This register is used to provide the modulation level for a data 0 in FSK Protocols. The signed 12 bit value that is represented by this register is presented to the PLL as the Baseband Frequency Word. The frequency result for this Modulation Scale will be the value of this register times the PLL Low Port Modulator Frequency Resolution; a typical value for the Frequency Resolution is 244.14 Hz.<br>The range of signed values that this register supports is -4096 to 4095 |

# A.2.1.8 TX DFT Modulation Pattern (DFT_PATTERN)

## A.2.1.8.1 Offset

| Register | Offset |
|---|---|
| DFT_PATTERN | 18h |

## A.2.1.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFT_MOD_PATTERN | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFT_MOD_PATTERN | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.1.8.3 Fields

| Field | Function |
|---|---|
| 31-0 | DFT Modulation Pattern |

| Field | Function |
|---|---|
| DFT_MOD_PATTERN | In TX DFT Pattern Register modes, if TX_DFT_EN is set, the bits in this register will be shifted out as the DFT Modulation Data in a repeating loop starting with bit [0]. |

## A.2.2 XCVR_PLL_DIG register descriptions

### A.2.2.1 XCVR_PLL_DIG_ADDR Memory map

XCVR_PLL_DIG base address: 4103_0224h

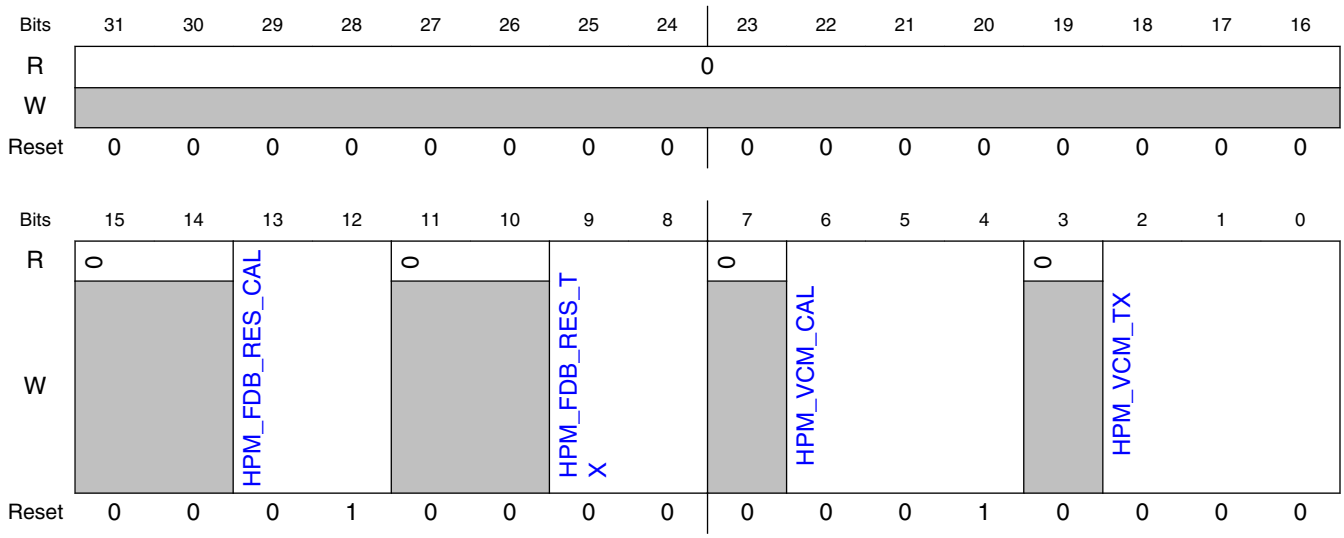
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | PLL HPM Analog Bump Control (HPM_BUMP) | 32 | RW | 0000_1010h |
| 4h | PLL Modulation Control (MOD_CTRL) | 32 | RW | 0000_0000h |
| 8h | PLL Channel Mapping (CHAN_MAP) | 32 | RW | 0000_0200h |
| Ch | PLL Lock Detect Control (LOCK_DETECT) | 32 | RW | 0060_6800h |
| 10h | PLL High Port Modulator Control (HPM_CTRL) | 32 | RW | 9084_0000h |
| 20h | PLL High Port Sigma Delta Results (HPM_SDM_RES) | 32 | RW | 0100_0000h |
| 24h | PLL Low Port Modulator Control (LPM_CTRL) | 32 | RW | 0202_0000h |
| 28h | PLL Low Port Sigma Delta Control 1 (LPM_SDM_CTRL1) | 32 | RW | 0026_0026h |
| 2Ch | PLL Low Port Sigma Delta Control 2 (LPM_SDM_CTRL2) | 32 | RW | 0008_0000h |
| 30h | PLL Low Port Sigma Delta Control 3 (LPM_SDM_CTRL3) | 32 | RW | 0010_0000h |
| 34h | PLL Low Port Sigma Delta Result 1 (LPM_SDM_RES1) | 32 | RO | 0000_0000h |
| 38h | PLL Low Port Sigma Delta Result 2 (LPM_SDM_RES2) | 32 | RO | 0400_0000h |
| 3Ch | PLL Delay Matching (DELAY_MATCH) | 32 | RW | 0000_0004h |
| 40h | PLL Coarse Tune Control (CTUNE_CTRL) | 32 | RW | 0000_0000h |
| 54h | PLL Coarse Tune Results (CTUNE_RES) | 32 | RO | 0962_0040h |

### A.2.2.2 PLL HPM Analog Bump Control (HPM_BUMP)

#### A.2.2.2.1 Offset

| Register | Offset |
|---|---|
| HPM_BUMP | 0h |

### A.2.2.2.2 Diagram




### A.2.2.2.3 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-12<br>HPM_FDB_RES_CAL | rfctrl_tx_dac_bump_fdb_res[1:0] during Calibration<br>This is the value applied to the rfctrl_tx_dac_bump_fdb_res[1:0] port during High Port Calibration.<br>This register sets the HPM DAC feedback resistor to increase the modulation gain during calibration.<br>00b - 29 kohms<br>01b - 58 kohms(gain of 2)<br>10b - 13 kohms<br>11b - 23.7 kohms |
| 11-10<br>— | Reserved |
| 9-8<br>HPM_FDB_RES_TX | rfctrl_tx_dac_bump_fdb_res[1:0] during Transmission<br>This is the value applied to the rfctrl_tx_dac_bump_fdb_res[1:0] port during Radio Transmissions.<br>This register sets the HPM DAC feedback resistor during transmissions.<br>00b - 29 kohms<br>01b - 58 kohms(gain of 2)<br>10b - 13 kohms<br>11b - 23.7 kohms |
| 7<br>— | Reserved |
| 6-4 | rfctrl_tx_dac_bump_vcm[2:0] during Calibration<br>This is the value applied to the rfctrl_tx_dac_bump_vcm[2:0] port during High Port Calibration. |

*Table continues on the next page...*

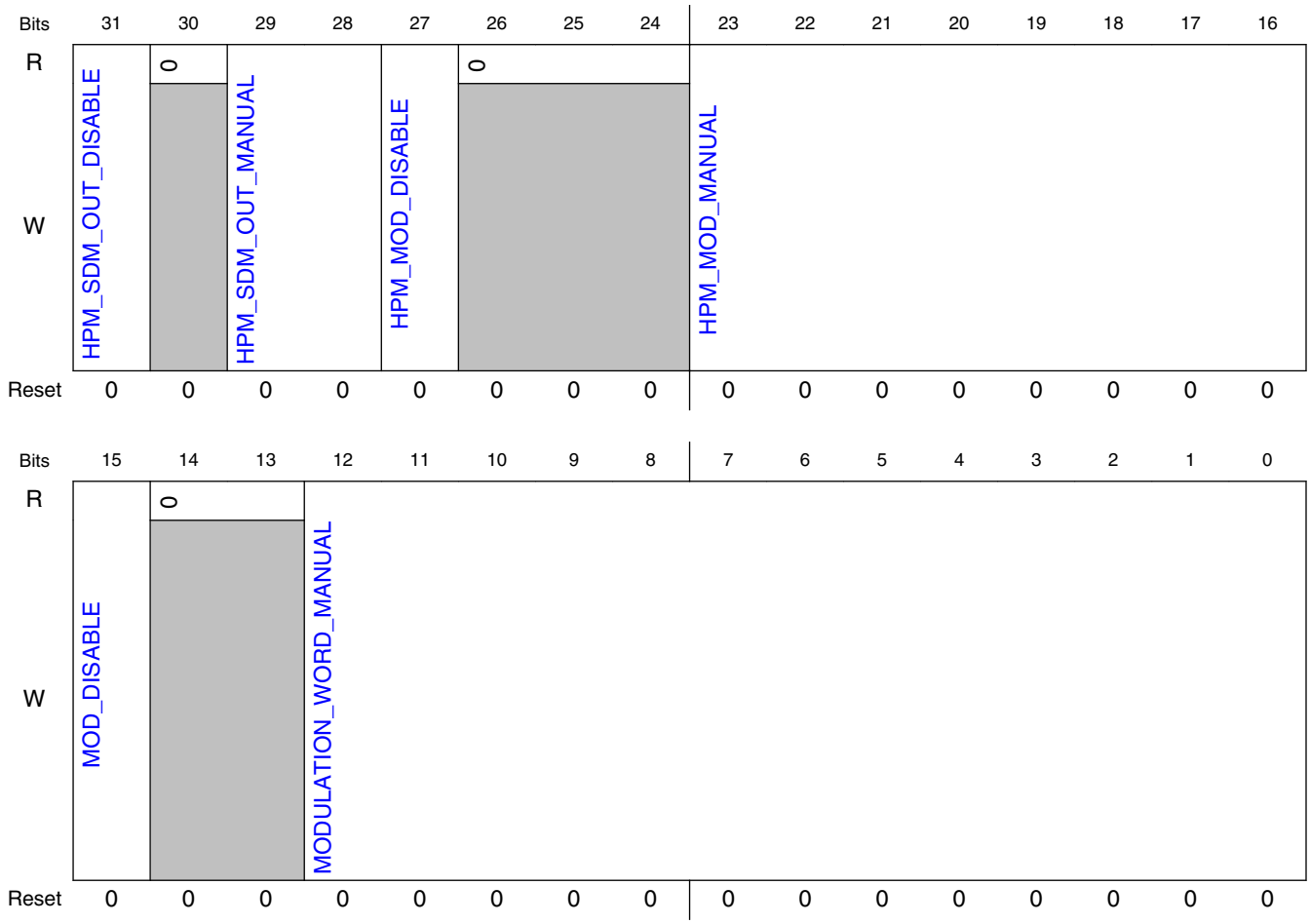
| Field | Function |
|---|---|
| HPM_VCM_CAL | This register sets the HPM DAC Op-Amp reference voltage during calibration.<br>000b - 432 mV<br>001b - 328 mV<br>010b - 456 mV<br>011b - 473 mV<br>100b - 488 mV<br>101b - 408 mV<br>110b - 392 mV<br>111b - 376 mV |
| 3<br>— | Reserved |
| 2-0<br>HPM_VCM_TX | rfctrl_tx_dac_bump_vcm[2:0] during Transmission<br>This is the value applied to the rfctrl_tx_dac_bump_vcm[2:0] port during Radio Transmissions.<br>This register sets the HPM DAC Op-Amp reference voltage during transmissions.<br>000b - 432 mV<br>001b - 328 mV<br>010b - 456 mV<br>011b - 473 mV<br>100b - 488 mV<br>101b - 408 mV<br>110b - 392 mV<br>111b - 376 mV |

## A.2.2.3 PLL Modulation Control (MOD_CTRL)

### A.2.2.3.1 Offset

| Register | Offset |
|---|---|
| MOD_CTRL | 4h |

## A.2.2.3.2 Diagram




## A.2.2.3.3 Fields

| Field | Function |
|---|---|
| 31<br>HPM_SDM_OUT_DISABLE | Disable HPM SDM out<br>If this bit is set, the High Port Sigma Delta Modulator output is disabled, and the High Port Fractional value applied to the VCO comes from the HPM_SDM_OUT_MANUAL register. |
| 30<br>— | Reserved |
| 29-28<br>HPM_SDM_OUT_MANUAL | Manual HPM SDM out<br>If HPM_SDM_OUT_DISABLE is set, this register is the Fractional value that is applied to the VCO High Port. |
| 27<br>HPM_MOD_DISABLE | Disable HPM Modulation<br>If this bit is set, the High Port Modulation is disabled, and the High Port Modulation value applied to the VCO comes from the HPM_MOD_MANUAL register. |
| 26-24<br>— | Reserved |

*Table continues on the next page...*

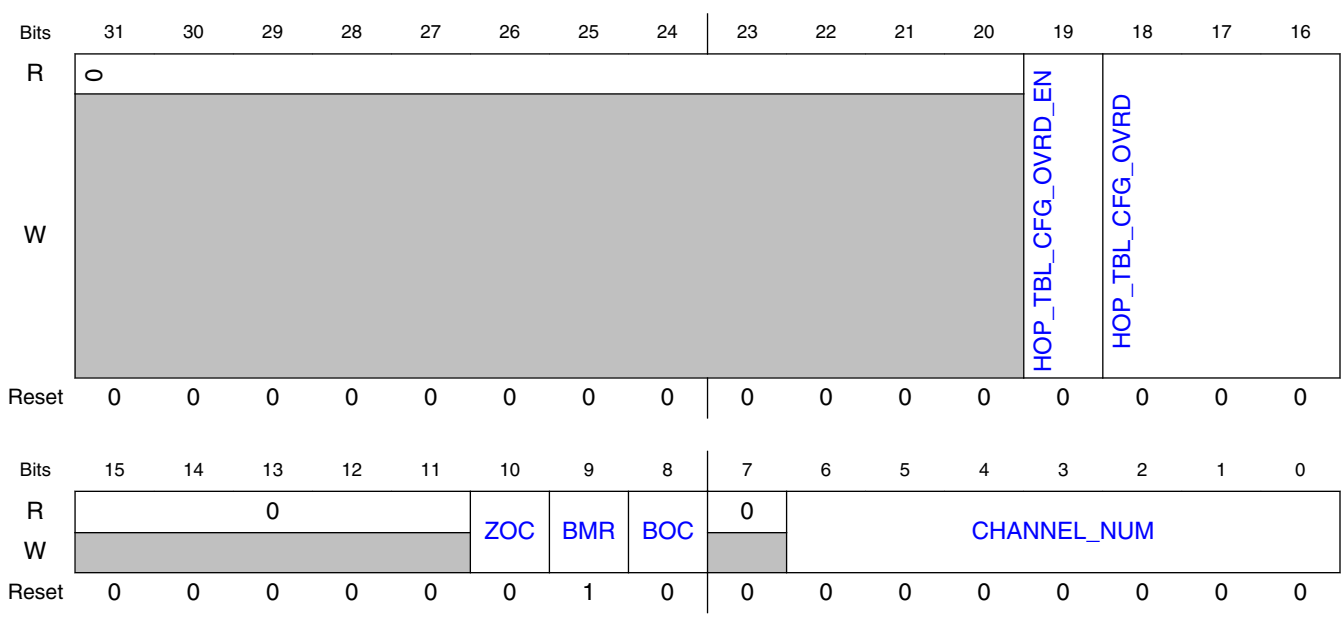
| Field | Function |
|---|---|
| 23-16<br>HPM_MOD_MANUAL | Manual HPM Modulation<br>If HPM_MOD_DISABLE is set, this register is the modulation value that is applied to the VCO High Port. |
| 15<br>MOD_DISABLE | Disable Modulation Word<br>If this bit is set, any modulation from the TX Digital is disabled and the source of the Baseband Frequency Word is the MODULATION_WORD_MANUAL register. |
| 14-13<br>— | Reserved |
| 12-0<br>MODULATION_WORD_MANUAL | Manual Modulation Word<br>If MOD_DISABLE is set, the signed 12 bit value that is represented by this register is the Baseband Frequency Word. |

## A.2.2.4 PLL Channel Mapping (CHAN_MAP)

### A.2.2.4.1 Offset

| Register | Offset |
|---|---|
| CHAN_MAP | 8h |

### A.2.2.4.2 Diagram



### A.2.2.4.3 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-20<br>—</td><td>Reserved</td></tr>
<tr><td>19<br>HOP_TBL_CFG_OVRD_EN</td><td>Hop Table Configuration Override Enable<br>If this bit is set then the HOP_TBL_CFG_OVRD and DFT_PATTERN Registers will be used to control the Coarse Tune, Channel Number, and Hop Table for the PLL Carrier Frequency Selection.</td></tr>
<tr><td>18-16<br>HOP_TBL_CFG_OVRD</td><td>Hop Table Configuration Override<br>If the HOP_TBL_CFG_OVRD_EN bit is set, then the DFT_PATTERN Register will be used to control the Hop Table for the PLL Carrier Frequency Selection.<br>010b - DFT_PATTERN[15:7] is signed offset to DFT_PATTERN[6:0] mapped channel number<br>011b - DFT_PATTERN[15:1] is signed Numerator, DFT_PATTERN[0] is integer selection</td></tr>
<tr><td>15-11<br>—</td><td>Reserved</td></tr>
<tr><td>10<br>ZOC</td><td>802.15.4 Channel Number Override<br>This bit controls the source of the 802.15.4 channel selection.<br>0b - 802.15.4 channel number comes from the 802.15.4 Link Layer.<br>1b - 802.15.4 channel number comes from the CHANNEL_NUM register (802.15.4 protocols 4 and 5)</td></tr>
<tr><td>9<br>BMR</td><td>BLE MBAN Channel Remap<br>This bit controls the mapping of BLE channel 39 in Radio Protocol 2, BLE overlap MBAN mode.<br>0b - BLE channel 39 is mapped to BLE channel 39, 2.480 GHz<br>1b - BLE channel 39 is mapped to MBAN channel 39, 2.399 GHz</td></tr>
<tr><td>8<br>BOC</td><td>BLE Channel Number Override<br>This bit controls the source of the BLE channel selection.<br>0b - BLE channel number comes from the BLE Link Layer<br>1b - BLE channel number comes from the CHANNEL_NUM register (BLE protocols 0 and 2)</td></tr>
<tr><td>7<br>—</td><td>Reserved</td></tr>
<tr><td>6-0<br>CHANNEL_NUM</td><td>Protocol specific Channel Number for PLL Frequency Mapping<br>When this register is active, it can be used to directly select a Protocol specific Channel Number, which is mapped internally to the correct Radio Carrier Frequency for PLL tuning. The internal mapping is detailed in the table below.<br>This register is active when BOC or ZOC are set along with their corresponding Radio Protocols, and this register is also active in protocols 1, 6 and 7.<br>The Radio Channel Frequency can also be selected by setting the SDM_MAP_DISABLE bit in the PLL_LPM_SDM_CTRL1 register along with the LPM_INTG, LPM_NUM, and LPM_DENOM registers to get a frequency that equals ((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM))<br>This table shows the internal mapping by Protocol of the Channel Numbers to the Radio Carrier Frequency (MHz).
<table>
<tr><th colspan="5">Radio Protocols Supported, Channel Frequency in MHz</th></tr>
<tr><th>BLE</th><th>MBAN</th><th>BLE Overlap MBAN</th><th>802.15.4</th><th>802.15.4j MBAN</th></tr>
</table>
</td></tr>
</table>

| Field | Function | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | 1 | | 2 | | 3 | | 4 | |
| | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz | Chan | MHz |
| | 0 | 2,402 | 0 | 2,360 | 0 | 2,402 | 11 | 2,405 | 0 | 2,363 |
| | 1 | 2,404 | 1 | 2,361 | 1 | 2,404 | 12 | 2,410 | 1 | 2,368 |
| | 2 | 2,406 | 2 | 2,362 | 2 | 2,406 | 13 | 2,415 | 2 | 2,373 |
| | 3 | 2,408 | 3 | 2,363 | 3 | 2,408 | 14 | 2,420 | 3 | 2,378 |
| | 4 | 2,410 | 4 | 2,364 | 4 | 2,410 | 15 | 2,425 | 4 | 2,383 |
| | 5 | 2,412 | 5 | 2,365 | 5 | 2,412 | 16 | 2,430 | 5 | 2,388 |
| | 6 | 2,414 | 6 | 2,366 | 6 | 2,414 | 17 | 2,435 | 6 | 2,393 |
| | 7 | 2,416 | 7 | 2,367 | 7 | 2,416 | 18 | 2,440 | 7 | 2,367 |
| | 8 | 2,418 | 8 | 2,368 | 8 | 2,418 | 19 | 2,445 | 8 | 2,372 |
| | 9 | 2,420 | 9 | 2,369 | 9 | 2,420 | 20 | 2,450 | 9 | 2,377 |
| | 10 | 2,422 | 10 | 2,370 | 10 | 2,422 | 21 | 2,455 | 10 | 2,382 |
| | 11 | 2,424 | 11 | 2,371 | 11 | 2,424 | 22 | 2,460 | 11 | 2,387 |
| | 12 | 2,426 | 12 | 2,372 | 12 | 2,426 | 23 | 2,465 | 12 | 2,392 |
| | 13 | 2,428 | 13 | 2,373 | 13 | 2,428 | 24 | 2,470 | 13 | 2,397 |
| | 14 | 2,430 | 14 | 2,374 | 14 | 2,430 | 25 | 2,475 | 14 | 2,395 |
| | 15 | 2,432 | 15 | 2,375 | 15 | 2,432 | 26 | 2,480 | | |
| | 16 | 2,434 | 16 | 2,376 | 16 | 2,434 | | | | |
| | 17 | 2,436 | 17 | 2,377 | 17 | 2,436 | | | | |
| | 18 | 2,438 | 18 | 2,378 | 18 | 2,438 | | | | |
| | 19 | 2,440 | 19 | 2,379 | 19 | 2,440 | | | | |
| | 20 | 2,442 | 20 | 2,380 | 20 | 2,442 | | | | |
| | 21 | 2,444 | 21 | 2,381 | 21 | 2,444 | | | | |
| | 22 | 2,446 | 22 | 2,382 | 22 | 2,446 | | | | |
| | 23 | 2,448 | 23 | 2,383 | 23 | 2,448 | | | | |
| | 24 | 2,450 | 24 | 2,384 | 24 | 2,450 | | | | |
| | 25 | 2,452 | 25 | 2,385 | 25 | 2,452 | | | | |
| | 26 | 2,454 | 26 | 2,386 | 26 | 2,454 | | | | |
| | 27 | 2,456 | 27 | 2,387 | 27 | 2,456 | | | | |
| | 28 | 2,458 | 28 | 2,388 | 28 | 2,458 | | | | |
| | 29 | 2,460 | 29 | 2,389 | 29 | 2,460 | | | | |
| | 30 | 2,462 | 30 | 2,390 | 30 | 2,390 | | | | |
| | 31 | 2,464 | 31 | 2,391 | 31 | 2,391 | | | | |
| | 32 | 2,466 | 32 | 2,392 | 32 | 2,392 | | | | |
| | 33 | 2,468 | 33 | 2,393 | 33 | 2,393 | | | | |
| | 34 | 2,470 | 34 | 2,394 | 34 | 2,394 | | | | |
| | 35 | 2,472 | 35 | 2,395 | 35 | 2,395 | | | | |
| | 36 | 2,474 | 36 | 2,396 | 36 | 2,396 | | | | |
| | 37 | 2,476 | 37 | 2,397 | 37 | 2,397 | | | | |
| | 38 | 2,478 | 38 | 2,398 | 38 | 2,398 | | | | |

| Field | Function |
|---|---|

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 39 | 2,480 | 39 | 2,399 | 39* | 2,480 | | | | |

* The BLE MBAN Channel Remap bit, BMR, controls the frequency mapping in this case.

| Radio Protocols Supported, Channel Frequency in MHz | | | | | |
|---|---|---|---|---|---|
| DFT, Generic | | | | | |
| 6,7,8,9 | | | | | |
| Channel | MHz | Channel | MHz | Channel | MHz |
| 0 | 2,360 | 43 | 2,403 | 86 | 2,446 |
| 1 | 2,361 | 44 | 2,404 | 87 | 2,447 |
| 2 | 2,362 | 45 | 2,405 | 88 | 2,448 |
| 3 | 2,363 | 46 | 2,406 | 89 | 2,449 |
| 4 | 2,364 | 47 | 2,407 | 90 | 2,450 |
| 5 | 2,365 | 48 | 2,408 | 91 | 2,451 |
| 6 | 2,366 | 49 | 2,409 | 92 | 2,452 |
| 7 | 2,367 | 50 | 2,410 | 93 | 2,453 |
| 8 | 2,368 | 51 | 2,411 | 94 | 2,454 |
| 9 | 2,369 | 52 | 2,412 | 95 | 2,455 |
| 10 | 2,370 | 53 | 2,413 | 96 | 2,456 |
| 11 | 2,371 | 54 | 2,414 | 97 | 2,457 |
| 12 | 2,372 | 55 | 2,415 | 98 | 2,458 |
| 13 | 2,373 | 56 | 2,416 | 99 | 2,459 |
| 14 | 2,374 | 57 | 2,417 | 100 | 2,460 |
| 15 | 2,375 | 58 | 2,418 | 101 | 2,461 |
| 16 | 2,376 | 59 | 2,419 | 102 | 2,462 |
| 17 | 2,377 | 60 | 2,420 | 103 | 2,463 |
| 18 | 2,378 | 61 | 2,421 | 104 | 2,464 |
| 19 | 2,379 | 62 | 2,422 | 105 | 2,465 |
| 20 | 2,380 | 63 | 2,423 | 106 | 2,466 |
| 21 | 2,381 | 64 | 2,424 | 107 | 2,467 |
| 22 | 2,382 | 65 | 2,425 | 108 | 2,468 |
| 23 | 2,383 | 66 | 2,426 | 109 | 2,469 |
| 24 | 2,384 | 67 | 2,427 | 110 | 2,470 |
| 25 | 2,385 | 68 | 2,428 | 111 | 2,471 |
| 26 | 2,386 | 69 | 2,429 | 112 | 2,472 |
| 27 | 2,387 | 70 | 2,430 | 113 | 2,473 |
| 28 | 2,388 | 71 | 2,431 | 114 | 2,474 |
| 29 | 2,389 | 72 | 2,432 | 115 | 2,475 |
| 30 | 2,390 | 73 | 2,433 | 116 | 2,476 |
| 31 | 2,391 | 74 | 2,434 | 117 | 2,477 |

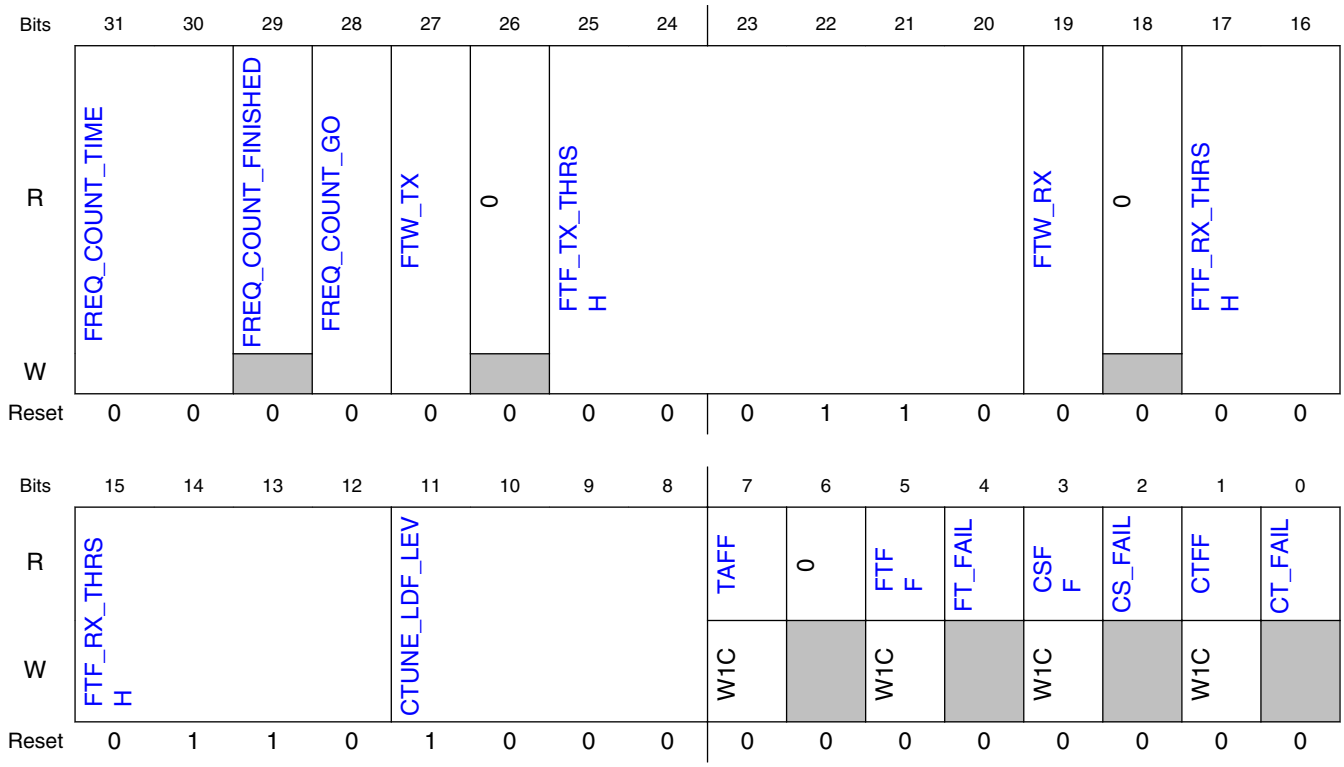
| Field | Function | | | | | |
|---|---|---|---|---|---|---|
| | 32 | 2,392 | 75 | 2,435 | 118 | 2,478 |
| | 33 | 2,393 | 76 | 2,436 | 119 | 2,479 |
| | 34 | 2,394 | 77 | 2,437 | 120 | 2,480 |
| | 35 | 2,395 | 78 | 2,438 | 121 | 2,481 |
| | 36 | 2,396 | 79 | 2,439 | 122 | 2,482 |
| | 37 | 2,397 | 80 | 2,440 | 123 | 2,483 |
| | 38 | 2,398 | 81 | 2,441 | 124 | 2,484 |
| | 39 | 2,399 | 82 | 2,442 | 125 | 2,485 |
| | 40 | 2,400 | 83 | 2,443 | 126 | 2,486 |
| | 41 | 2,401 | 84 | 2,444 | 127 | 2,487 |
| | 42 | 2,402 | 85 | 2,445 | | |

## A.2.2.5 PLL Lock Detect Control (LOCK_DETECT)

### A.2.2.5.1 Offset

| Register | Offset |
|---|---|
| LOCK_DETECT | Ch |

## A.2.2.5.2 Diagram




## A.2.2.5.3 Fields

| Field | Function |
|---|---|
| 31-30<br>FREQ_COUNT_TIME | Frequency Meter Count Time<br>This is the length of time that the Frequency Meter will count VCO clocks.<br>00b - 800 us<br>01b - 25 us<br>10b - 50 us<br>11b - 100 us |
| 29<br>FREQ_COUNT_FINISHED | Frequency Meter has finished the Count Time<br>This bit is set when the FREQ_COUNT_TIME value is reached, it is cleared when FREQ_COUNT_GO is cleared. |
| 28<br>FREQ_COUNT_GO | Start the Frequency Meter<br>The Frequency Meter starts when this bit is set and runs until the FREQ_COUNT_TIME value is reached. The bit should not be cleared until after the counting is finished.<br>After the Counting Time finishes, the FREQ_COUNT_FINISHED bit will read as a one. Then the measured frequency can be calculated by reading the frequency counts in the DFT_FREQ_COUNTER register (in the XCVR_ANALOG register space) and dividing by the FREQ_COUNT_TIME.<br>Note that the counting is done at the VCO frequency which is 2X the Carrier Frequency |
| 27<br>FTW_TX | TX Frequency Target Window time select<br>In Radio Transmit Mode, this bit selects the length of time to count for the estimation of PLL lock frequency, which is compared with the Frequency Target Window, set by FTF_TX. |

*Table continues on the next page...*

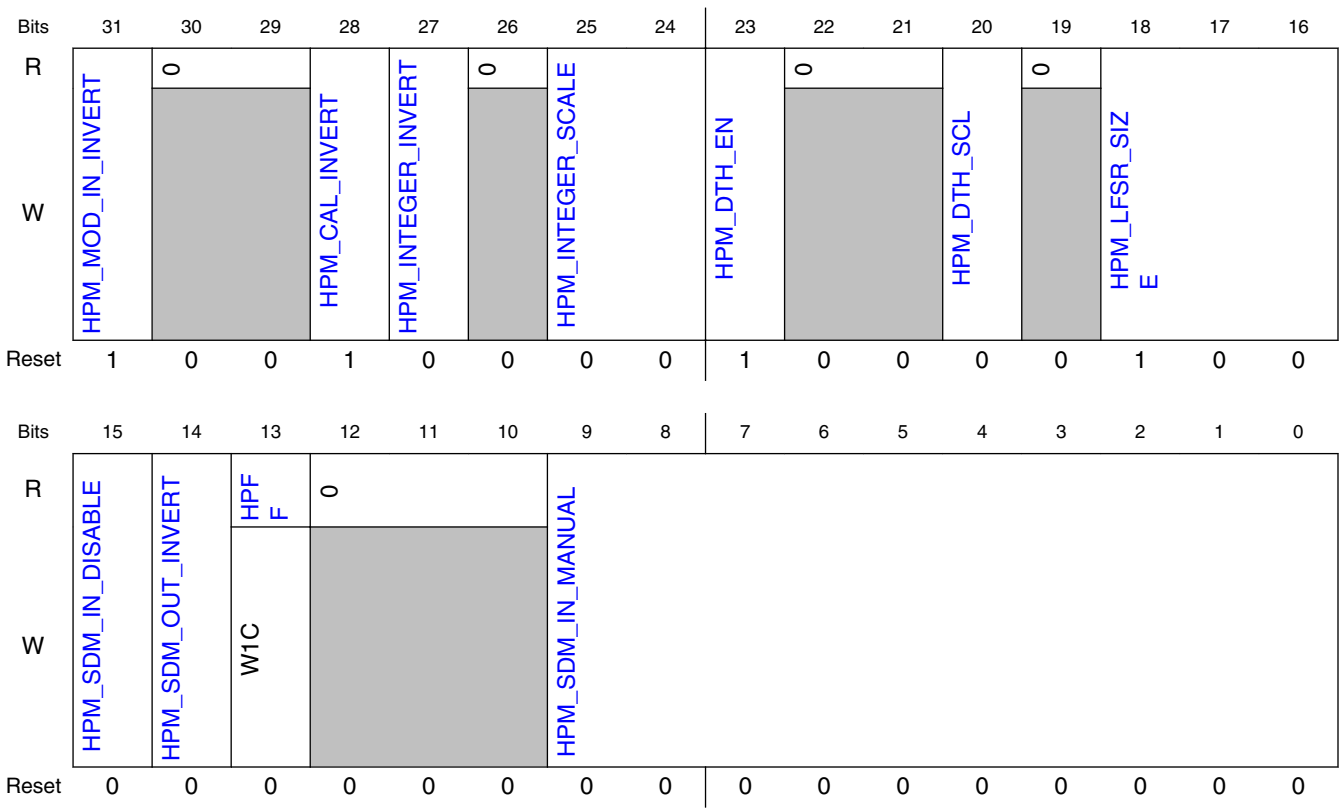
| Field | Function |
|---|---|
| | 0b - 4 us<br>1b - 8 us |
| 26<br>— | Reserved |
| 25-20<br>FTF_TX_THRSH | TX Frequency Target Fail Threshold<br>In Radio Transmit Mode, the FT_FAIL and FTFF bits will be set after the Frequency Target Count completes if the absolute value of the count difference from the frequency target count is greater than this register value. |
| 19<br>FTW_RX | RX Frequency Target Window time select<br>In Radio Receive Mode, this bit selects the length of time to count for the estimation of PLL lock frequency, which is compared with the Frequency Target Window, set by FTF_RX.<br>0b - 4 us<br>1b - 8 us |
| 18<br>— | Reserved |
| 17-12<br>FTF_RX_THRSH | RX Frequency Target Fail Threshold<br>In Radio Receive Mode, the FT_FAIL and FTFF bits will be set after the Frequency Target Count completes if the absolute value of the count difference from the frequency target count is greater than this register value. |
| 11-8<br>CTUNE_LDF_LEV | CTUNE Lock Detect Fail Level<br>The CT_FAIL and CTFF bits will be set after Coarse Tune Calibration completes if the absolute value of the Coarse Tune best count difference (CTUNE_BEST_DIFF in the PLL_CTUNE_RESULTS register) is greater than this register value. |
| 7<br>TAFF | TSM Abort Failure Flag<br>This bit is set if the TSM Sequence Aborts, and this bit is cleared by writing a 1 to it. |
| 6<br>— | Reserved |
| 5<br>FTFF | Frequency Target Failure Flag<br>This bit is set when FT_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 4<br>FT_FAIL | Real time status of Frequency Target Failure<br>If the Frequency Target Count has completed and the count was out of the range selected by FTW_TX or FTW_RX, then this bit will be set. |
| 3<br>CSFF | Cycle Slip Failure Flag, held until cleared<br>This bit is set when CS_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 2<br>CS_FAIL | Real time status of Cycle Slip circuit<br>This bit shows the real-time status of the Cycle Slip State Machine Interrupt which is configured using the control bits in the PLL_HPM_CAL1 and PLL_HPM_CAL2 registers. |
| 1<br>CTFF | CTUNE Failure Flag, held until cleared<br>This bit is set when CT_FAIL is first set, and this bit is cleared by writing a 1 to it. |
| 0<br>CT_FAIL | Real time status of Coarse Tune Fail signal<br>If the Coarse Tune Calibration has completed and the best count difference is out of the range selected by CTUNE_LDF_LEV, then this bit will be set. |

# A.2.2.6 PLL High Port Modulator Control (HPM_CTRL)

## A.2.2.6.1 Offset

| Register | Offset |
|---|---|
| HPM_CTRL | 10h |

## A.2.2.6.2 Diagram




## A.2.2.6.3 Fields

| Field | Function |
|---|---|
| 31<br>HPM_MOD_IN_INVERT | Invert High Port Modulation<br>If this bit is set then the High Port Modulation Word is inverted before it is multiplied and split into Integer and Fractional values. |
| 30-29 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 28<br>HPM_CAL_INVERT | Invert High Port Modulator Calibration<br>If this bit is set then the order of the High Port Calibration is reversed in order to get a positive count difference between the Maximum and Minimum settings of the HPM DAC. |
| 27<br>HPM_INTEGER_INVERT | Invert High Port Modulation Integer<br>If this bit is set then the High Port Modulation Integer value, after the multiply and split into Integer and Fractional values, will be inverted before it is applied to the VCO High Port DAC Array. |
| 26<br>— | Reserved |
| 25-24<br>HPM_INTEGER_SCALE | High Port Modulation Integer Scale<br>This register controls the scaling of the High Port Modulation Integer Value applied to the VCO High Port DAC Array.<br>00b - No Scaling<br>01b - Multiply by 2<br>10b - Divide by 2<br>11b - Reserved |
| 23<br>HPM_DTH_EN | Dither Enable for HPM LFSR<br>If this bit is set, the High Port Fraction will be Dithered by the High Port LFSR before it is applied to the High Port SDM. |
| 22-21<br>— | Reserved |
| 20<br>HPM_DTH_SCL | HPM Dither Scale<br>If this bit is set, the LFSR dithering of the High Port Fraction will be multiplied by 2. |
| 19<br>— | Reserved |
| 18-16<br>HPM_LFSR_SIZE | HPM LFSR Length<br>This register selects the length of the HPM LFSR and the associated LFSR Tap Mask<br>000b - LFSR 9, tap mask 100010000<br>001b - LFSR 10, tap mask 1001000000<br>010b - LFSR 11, tap mask 11101000000<br>011b - LFSR 13, tap mask 1101100000000<br>100b - LFSR 15, tap mask 111010000000000<br>101b - LFSR 17, tap mask 11110000000000000<br>110b - Reserved<br>111b - Reserved |
| 15<br>HPM_SDM_IN_DISABLE | Disable HPM SDM Input<br>If this bit is set, the Fractional portion of the High Port Modulation to the High Port SDM is disabled, and the High Port SDM input comes from the HPM_SDM_IN_MANUAL register. |
| 14<br>HPM_SDM_OUT_INVERT | Invert HPM SDM Output<br>If this bit is set the High Port SDM result will be Inverted before it is applied to the VCO High Port DAC Array. |
| 13<br>HPFF | HPM SDM Invalid Flag<br>This bit is set if the High Port Sigma Delta Modulator output is invalid due to an error in the fraction applied, and this bit is cleared by writing a 1 to it. |
| 12-10 | Reserved |

*Table continues on the next page...*

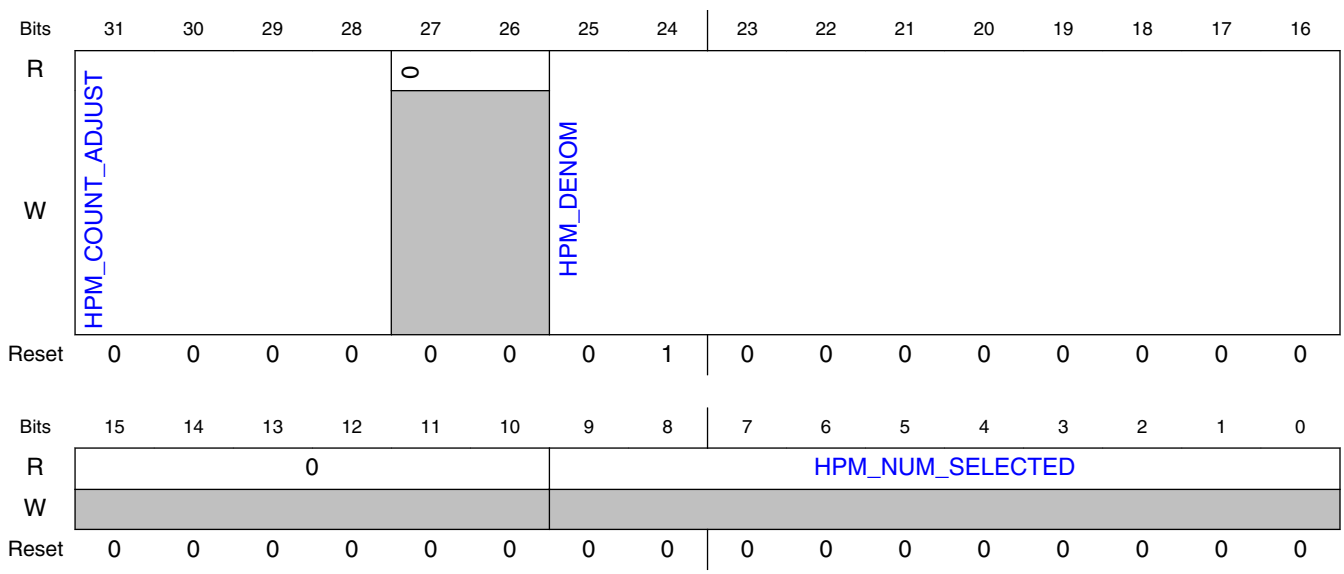
| Field | Function |
|---|---|
| — | |
| 9-0<br>HPM_SDM_IN_MANUAL | Manual High Port SDM Fractional value<br>If HPM_SDM_IN_DISABLE is set, this register is the value that is applied as the Fractional value to the input of the High Port SDM. |

## A.2.2.7 PLL High Port Sigma Delta Results (HPM_SDM_RES)

### A.2.2.7.1 Offset

| Register | Offset |
|---|---|
| HPM_SDM_RES | 20h |

### A.2.2.7.2 Diagram




### A.2.2.7.3 Fields

| Field | Function |
|---|---|
| 31-28<br>HPM_COUNT_ADJUST | HPM_COUNT_ADJUST<br>This register represents a signed three bit value that is used to adjust the High Port Calibration Frequency Count Difference.<br>The range of adjustment is -8 to +7, applied to the difference between Count 1 and Count 2. |

*Table continues on the next page...*

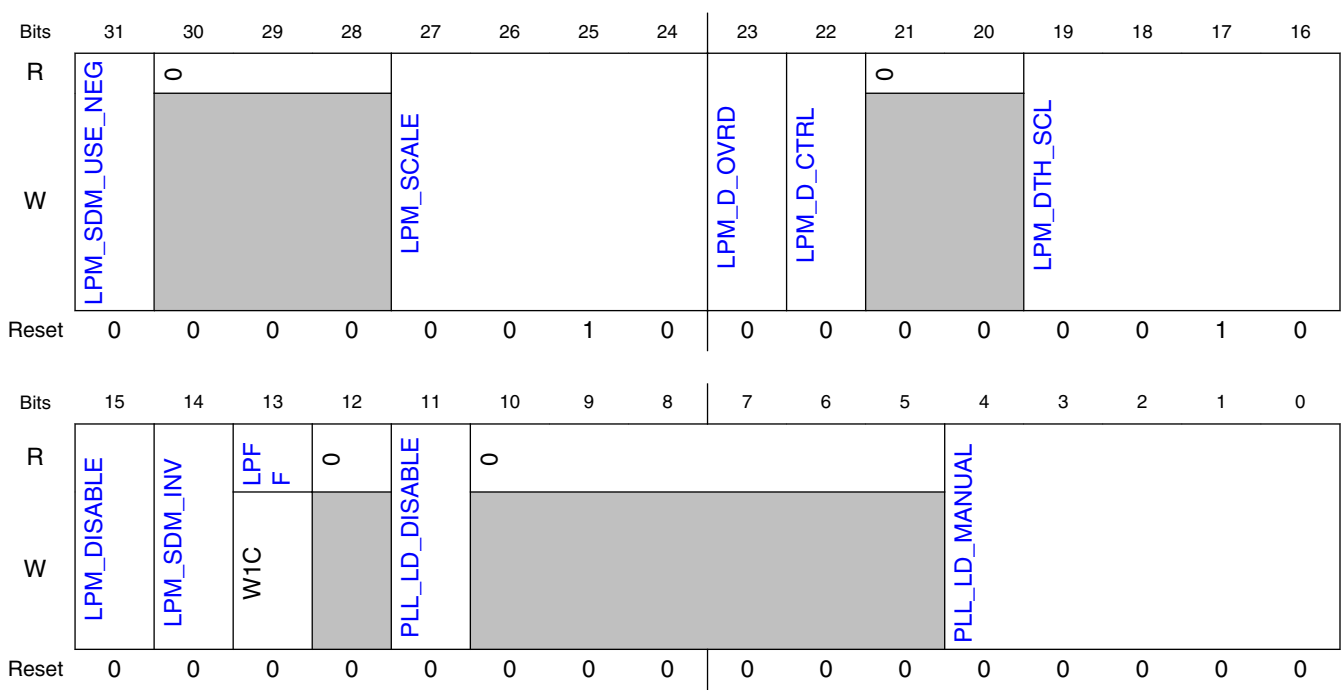
| Field | Function |
|---|---|
| 27-26<br>— | Reserved |
| 25-16<br>HPM_DENOM | High Port Modulator SDM Denominator<br>This is the denominator that is currently being applied to the High Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |
| 15-10<br>— | Reserved |
| 9-0<br>HPM_NUM_SELECTED | High Port Modulator SDM Numerator<br>This is the numerator that is currently being applied to the High Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## A.2.2.8 PLL Low Port Modulator Control (LPM_CTRL)

### A.2.2.8.1 Offset

| Register | Offset |
|---|---|
| LPM_CTRL | 24h |

### A.2.2.8.2 Diagram



### A.2.2.8.3 Fields

| Field | Function |
|---|---|
| 31<br>LPM_SDM_USE_NEG | Use the Negedge of the Sigma Delta clock<br>If this bit is set then the negative edge of the Sigma Delta clock is used to launch the Low Port Modulation word to the VCO Loop Divider. |
| 30-28<br>— | Reserved |
| 27-24<br>LPM_SCALE | LPM Scale Factor<br>This register controls the scaling of the Baseband Frequency Word and is used to match the Modulation Frequency Deviation required to the Low Port Sigma Delta Modulator LSB size in Hz.<br>0000b - No Scaling<br>0001b - Multiply by 2<br>0010b - Multiply by 4<br>0011b - Multiply by 8<br>0100b - Multiply by 16<br>0101b - Multiply by 32<br>0110b - Multiply by 64<br>0111b - Multiply by 128<br>1000b - Multiply by 256, this is the intended setting for normal operation.<br>1001b - Multiply by 512<br>1010b - Multiply by 1024<br>1011b - Multiply by 2048<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| 23<br>LPM_D_OVRD | LPM Dither Override Mode Select<br>When this bit is set, the Scaled Baseband Frequency Word applied to the Low Port Sigma Delta Modulator will be dithered if LPM_D_CTRL is set, and not dithered if LPM_D_CTRL is cleared.<br>If this bit is cleared, then the LPM Numerator will be dithered in Radio Receive mode, and also in Radio Transmit mode when the LPM Numerator approaches an Integer value in order to preserve the validity of the Sigma Delta Modulator output. |
| 22<br>LPM_D_CTRL | LPM Dither Control in Override Mode<br>If LPM_D_OVRD is set, this bit turns LPM Dithering on and off. |
| 21-20<br>— | Reserved |
| 19-16<br>LPM_DTH_SCL | LPM Dither Scale<br>This register controls the scale of the Dithering added to the Scaled Baseband Frequency Word before it is applied to the Low Port Sigma Delta Modulator as the LPM Numerator.<br>The unit for the ranges shown below is the LP SDM LSB in Hz.<br>0000b - Reserved<br>0001b - Reserved<br>0010b - Reserved<br>0011b - Reserved<br>0100b - Reserved<br>0101b - -128 to 96<br>0110b - -256 to 192 |

*Table continues on the next page...*

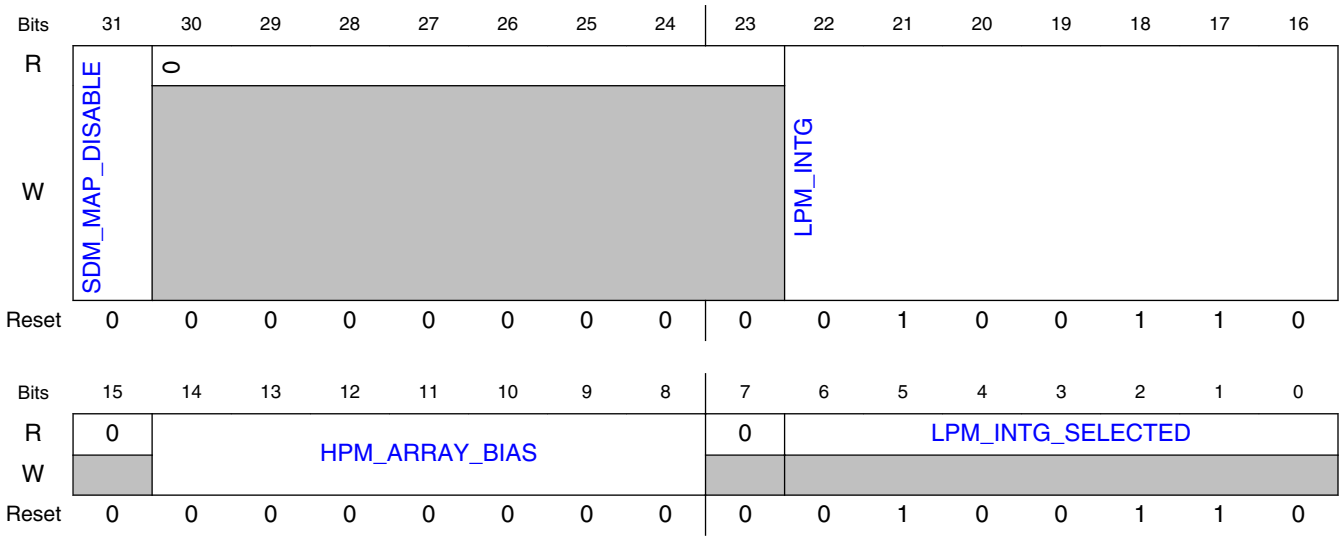
| Field | Function |
|---|---|
| | 0111b - -512 to 384<br>1000b - -1024 to 768, this is the intended setting for normal operation.<br>1001b - -2048 to 1536<br>1010b - -4096 to 3072<br>1011b - -8192 to 6144<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| 15<br>LPM_DISABLE | Disable LPM SDM<br>This bit controls the Modulation of the Low Port Sigma Delta.<br>If this bit is set, the Low Port Sigma Delta Modulator will be active and control the PLL to maintain a steady frequency based on the current Integer, Numerator, and Denominator values that are being applied.<br>No Modulation or Dithering will be added to the steady frequency result. |
| 14<br>LPM_SDM_INV | Invert LPM SDM<br>If this bit is set the Scaled Baseband Frequency Word, including any Dithering, will be Inverted before it is applied to the Low Port Sigma Delta Modulator. |
| 13<br>LPFF | LPM SDM Invalid Flag<br>This bit is set if the Low Port Sigma Delta Modulator output is invalid due to an error in the fraction applied, and this bit is cleared by writing a 1 to it. |
| 12<br>— | Reserved |
| 11<br>PLL_LD_DISABLE | Disable PLL Loop Divider<br>If this bit is set, the Low Port Sigma Delta Modulator output is disabled, and the PLL Loop Divider value applied to the PLL comes from the PLL_LD_MANUAL register. |
| 10-5<br>— | Reserved |
| 4-0<br>PLL_LD_MANUAL | Manual PLL Loop Divider value<br>If PLL_LD_DISABLE is set, this register is the value that is applied to the PLL Loop Divider. |

## A.2.2.9 PLL Low Port Sigma Delta Control 1 (LPM_SDM_CTRL1)

### A.2.2.9.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_CTRL1 | 28h |

## A.2.2.9.2 Diagram




## A.2.2.9.3 Fields

| Field | Function |
|---|---|
| 31<br>SDM_MAP_DISABLE | Disable SDM Mapping<br>If this bit is set, the Low Port Sigma Delta Modulator internal frequency mapping based on Protocol specific channel numbers is disabled, and the Radio Channel Frequency is selected by setting the LPM_INTG, LPM_NUM, and LPM_DENOM registers to get a frequency that equals :<br>((Reference Clock Frequency x 2) x (LPM_INTG + (LPM_NUM / LPM_DENOM)) |
| 30-23<br>— | Reserved |
| 22-16<br>LPM_INTG | Manual Low Port Modulation Integer Value<br>If SDM_MAP_DISABLE is set, this register is the value that is applied to the Low Port Sigma Delta Modulator for the Integer, the nominal range is 36 to 39 in decimal. |
| 15<br>— | Reserved |
| 14-8<br>HPM_ARRAY_BIAS | Bias value for High Port DAC Array Midpoint<br>The value of this register represents a signed six bit value that can be used to adjust the midpoint of the High Port Modulator DAC.<br>The range of adjustment is -64 to +63, and the adjustment is made to the High Port Modulation Word before the multiply, and before the split into Integer and Fractional values.<br>The range of adjustment in Hz is approximately +/- 62.5 kHz |
| 7<br>— | Reserved |
| 6-0<br>LPM_INTG_SELECTED | Low Port Modulation Integer Value Selected<br>This shows the Integer value that is currently being applied to the Low Port Sigma Delta Modulator. |

## A.2.2.10 PLL Low Port Sigma Delta Control 2 (LPM_SDM_CTRL2)

### A.2.2.10.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_CTRL2 | 2Ch |

### A.2.2.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_NUM | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_NUM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.2.10.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_NUM | Low Port Modulation Numerator<br>If SDM_MAP_DISABLE is set, this register is the signed 27 bit value that is applied to the Low Port Sigma Delta Modulator for the Numerator. For valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## A.2.2.11 PLL Low Port Sigma Delta Control 3 (LPM_SDM_CTRL3)

### A.2.2.11.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_CTRL3 | 30h |

### A.2.2.11.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_DENOM | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_DENOM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.2.11.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_DENOM | Low Port Modulation Denominator<br>If SDM_MAP_DISABLE is set, this register is the signed 27 bit value that is applied to the Low Port Sigma Delta Modulator for the Denominator. For valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## A.2.2.12 PLL Low Port Sigma Delta Result 1 (LPM_SDM_RES1)

### A.2.2.12.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_RES1 | 34h |

### A.2.2.12.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_NUM_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_NUM_SELECTED | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.2.12.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_NUM_SELECTED | Low Port Modulation Numerator Applied<br>This is the value that is currently being applied to the Low Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## A.2.2.13 PLL Low Port Sigma Delta Result 2 (LPM_SDM_RES2)

### A.2.2.13.1 Offset

| Register | Offset |
|---|---|
| LPM_SDM_RES2 | 38h |

### A.2.2.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LPM_DENOM_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LPM_DENOM_SELECTED | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.2.13.3 Fields

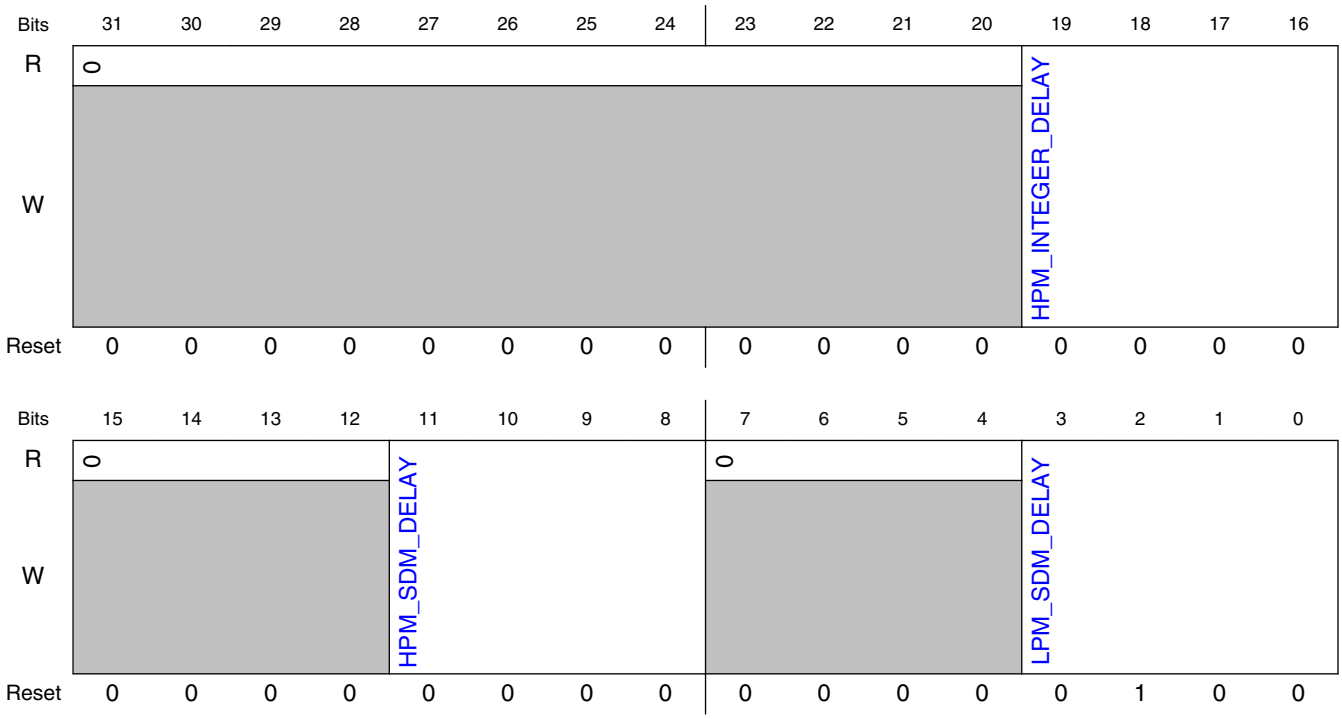
| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>LPM_DENOM_SELECTED | Low Port Modulation Denominator Selected<br>This is the value that is currently being applied to the Low Port Sigma Delta Modulator, for valid Sigma Delta operation the resulting fraction NUM/DENOM must be in the range -0.6 to +0.6 |

## A.2.2.14 PLL Delay Matching (DELAY_MATCH)

### A.2.2.14.1 Offset

| Register | Offset |
|---|---|
| DELAY_MATCH | 3Ch |

## A.2.2.14.2 Diagram




## A.2.2.14.3 Fields

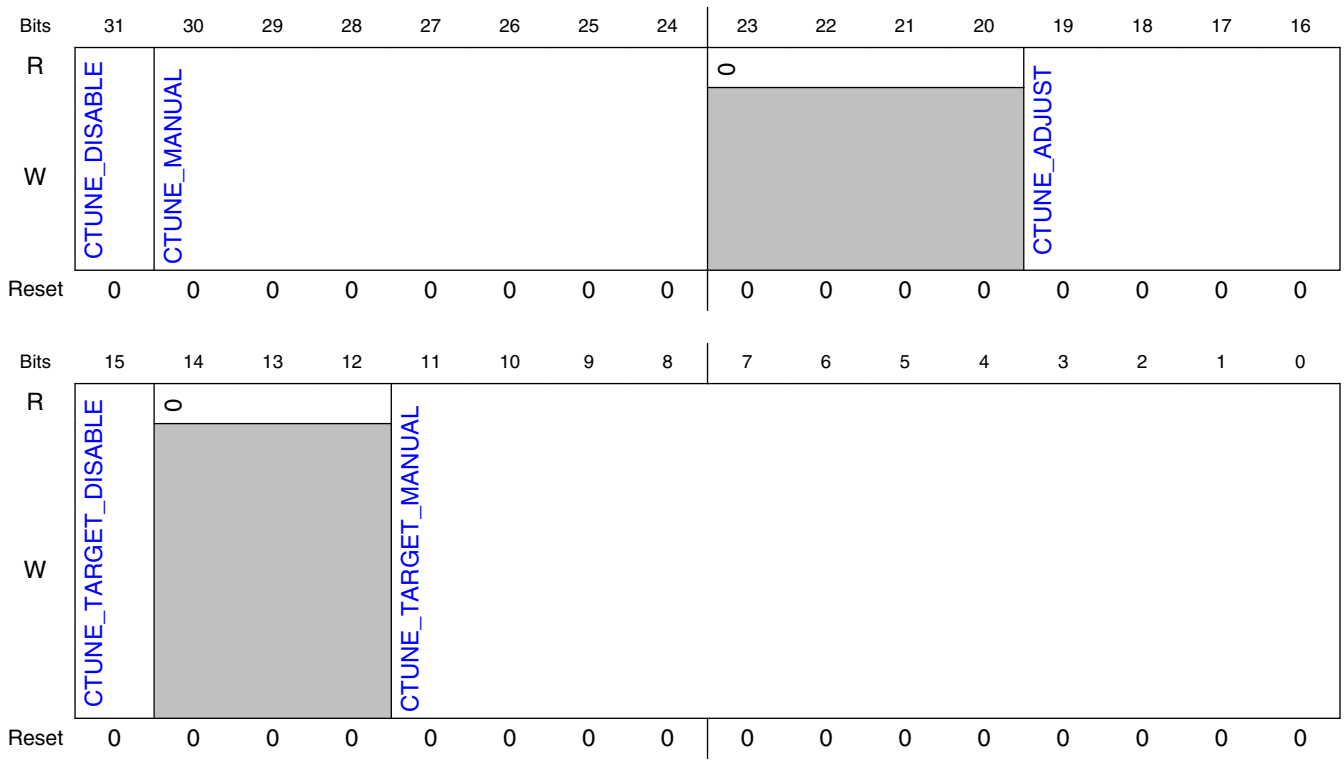
| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>HPM_INTEGER_DELAY | High Port Integer Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock divided by 2 to delay the High Port Integer modulation of the VCO High Port DAC Array. |
| 15-12<br>— | Reserved |
| 11-8<br>HPM_SDM_DELAY | High Port SDM Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock divided by 2 to delay the High Port Sigma Delta modulation of the VCO High Port Fraction. Note that the High Port SDM is clocked by the PLL Sigma Delta Clock but the modulation is based on a divide by 2 version of this same clock. |
| 7-4<br>— | Reserved |
| 3-0<br>LPM_SDM_DELAY | Low Port SDM Delay Matching<br>This register selects the number of clock cycles of the PLL Sigma Delta Clock to delay the Low Port Sigma Delta modulation of the PLL Loop Divider. |

# A.2.2.15 PLL Coarse Tune Control (CTUNE_CTRL)

## A.2.2.15.1 Offset

| Register | Offset |
|---|---|
| CTUNE_CTRL | 40h |

## A.2.2.15.2 Diagram




## A.2.2.15.3 Fields

| Field | Function |
|---|---|
| 31<br>CTUNE_DISABLE | Coarse Tune Disable<br>If this bit is set, the Coarse Tune Setting applied to the VCO comes from the CTUNE_MANUAL register. |
| 30-24<br>CTUNE_MANUAL | Manual Coarse Tune Setting<br>If CTUNE_DISABLE is set, this register is the value that is applied to the VCO as the Coarse Tune Setting. |
| 23-20<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19-16<br>CTUNE_ADJUST | Coarse Tune Count Adjustment<br>This register is a signed three bit value that adjusts the PLL Frequency Meter count used in the Coarse Tune Calibration.<br>The range of adjustment is -8 to +7, and the adjustment is only made to the PLL Frequency count used by the Coarse Tune Calibration sequence. |
| 15<br>CTUNE_TARGET_DISABLE | Disable Coarse Tune Target<br>If this bit is set, the Frequency Target presented to the Coarse Tune Calibrator comes from the CTUNE_TARGET_MANUAL register. |
| 14-12<br>— | Reserved |
| 11-0<br>CTUNE_TARGET_MANUAL | Manual Coarse Tune Target<br>If CTUNE_TD is set, this register is the value that is presented to the Coarse Tune Calibrator as the Frequency Target in MHz. The nominal range of this target is from 2360 to 2487 in decimal. |

# A.2.2.16 PLL Coarse Tune Results (CTUNE_RES)

## A.2.2.16.1 Offset

| Register | Offset |
|---|---|
| CTUNE_RES | 54h |

## A.2.2.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CTUNE_FREQ_SELECTED | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CTUNE_BEST_DIFF | | | | | | | | 0 | CTUNE_SELECTED | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.2.16.3 Fields

| Field | Function |
|---|---|
| 31-28 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27-16<br>CTUNE_FREQ_SELECTED | Coarse Tune Frequency Selected<br>This is the Frequency Target in MHz that is currently being presented to the Coarse Tune Calibrator. |
| 15-8<br>CTUNE_BEST_DIFF | Coarse Tune Absolute Best Difference<br>This is the absolute value of the best difference found during Coarse Tune between the targeted frequency count and the actual frequency count. |
| 7<br>— | Reserved |
| 6-0<br>CTUNE_SELECTED | Coarse Tune Setting to VCO<br>This is the current VCO Coarse Tune setting, it is the result of the Coarse Tune Calibration, unless overridden using CTUNE_DISABLE. |

## A.2.3 XCVR_RX_DIG register descriptions

### A.2.3.1 XCVR_RX_DIG_ADDR Memory map

XCVR_RX_DIG base address: 4103_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | RX Digital Control (RX_DIG_CTRL) | 32 | RW | 0000_0000h |
| 4h | AGC Control 0 (AGC_CTRL_0) | 32 | RW | 0000_0000h |
| 8h | AGC Control 1 (AGC_CTRL_1) | 32 | RW | 0000_0000h |
| Ch | AGC Control 2 (AGC_CTRL_2) | 32 | RW | 00A6_9000h |
| 10h | AGC Control 3 (AGC_CTRL_3) | 32 | RW | 0000_0000h |
| 14h | AGC Status (AGC_STAT) | 32 | RO | 0000_0000h |
| 18h | RSSI Control 0 (RSSI_CTRL_0) | 32 | RW | 0030_0000h |
| 1Ch | RSSI Control 1 (RSSI_CTRL_1) | 32 | RW | 0000_0000h |
| 24h | DCOC Control 0 (DCOC_CTRL_0) | 32 | RW | 0000_0000h |
| 28h | DCOC Control 1 (DCOC_CTRL_1) | 32 | RW | 0000_0000h |
| 2Ch | DCOC DAC Initialization (DCOC_DAC_INIT) | 32 | RW | 8080_2020h |
| 30h | DCOC Digital Correction Manual Override (DCOC_DIG_MAN) | 32 | RW | 0000_0000h |
| 34h | DCOC Calibration Gain (DCOC_CAL_GAIN) | 32 | RW | 0000_0000h |
| 38h | DCOC Status (DCOC_STAT) | 32 | RO | 8080_2020h |
| 3Ch | DCOC DC Estimate (DCOC_DC_EST) | 32 | RO | 0000_0000h |
| 40h | DCOC Calibration Reciprocals (DCOC_CAL_RCP) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 44h | DCOC Control 2 (DCOC_CTRL_2) | 32 | RW | 0000_0000h |
| 48h | IQMC Control (IQMC_CTRL) | 32 | RW | 0400_8000h |
| 4Ch | IQMC Calibration (IQMC_CAL) | 32 | RW | 0000_0400h |
| 50h | LNA_GAIN Step Values 3..0 (LNA_GAIN_VAL_3_0) | 32 | RW | 3809_321Dh |
| 54h | LNA_GAIN Step Values 7..4 (LNA_GAIN_VAL_7_4) | 32 | RW | 8B74_5D4Fh |
| 58h | LNA_GAIN Step Values 8 (LNA_GAIN_VAL_8) | 32 | RW | 0000_B6A1h |
| 5Ch | BBA Resistor Tune Values 7..0 (BBA_RES_TUNE_VAL_7_0) | 32 | RW | 0000_0000h |
| 60h | BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_VAL_10_8) | 32 | RW | 0000_0000h |
| 64h | LNA Linear Gain Values 2..0 (LNA_GAIN_LIN_VAL_2_0) | 32 | RW | 0000_0000h |
| 68h | LNA Linear Gain Values 5..3 (LNA_GAIN_LIN_VAL_5_3) | 32 | RW | 0000_0000h |
| 6Ch | LNA Linear Gain Values 8..6 (LNA_GAIN_LIN_VAL_8_6) | 32 | RW | 0000_0000h |
| 70h | LNA Linear Gain Values 9 (LNA_GAIN_LIN_VAL_9) | 32 | RW | 0000_0000h |
| 74h | BBA Resistor Tune Values 3..0 (BBA_RES_TUNE_LIN_VAL_3_0) | 32 | RW | 0000_0000h |
| 78h | BBA Resistor Tune Values 7..4 (BBA_RES_TUNE_LIN_VAL_7_4) | 32 | RW | 0000_0000h |
| 7Ch | BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_LIN_VAL_10_8) | 32 | RW | 0000_0000h |
| 80h | AGC Gain Tables Step 03..00 (AGC_GAIN_TBL_03_00) | 32 | RW | 0000_0000h |
| 84h | AGC Gain Tables Step 07..04 (AGC_GAIN_TBL_07_04) | 32 | RW | 0000_0000h |
| 88h | AGC Gain Tables Step 11..08 (AGC_GAIN_TBL_11_08) | 32 | RW | 0000_0000h |
| 8Ch | AGC Gain Tables Step 15..12 (AGC_GAIN_TBL_15_12) | 32 | RW | 0000_0000h |
| 90h | AGC Gain Tables Step 19..16 (AGC_GAIN_TBL_19_16) | 32 | RW | 0000_0000h |
| 94h | AGC Gain Tables Step 23..20 (AGC_GAIN_TBL_23_20) | 32 | RW | 0000_0000h |
| 98h | AGC Gain Tables Step 26..24 (AGC_GAIN_TBL_26_24) | 32 | RW | 0000_0000h |
| A0h - 108h | DCOC Offset (DCOC_OFFSET_0 - DCOC_OFFSET_26) | 32 | RW | 0000_0000h |
| 10Ch | DCOC BBA DAC Step (DCOC_BBA_STEP) | 32 | RW | 0000_0000h |
| 110h | DCOC TZA DAC Step 0 (DCOC_TZA_STEP_0) | 32 | RW | 0000_0000h |
| 114h | DCOC TZA DAC Step 1 (DCOC_TZA_STEP_1) | 32 | RW | 0000_0000h |
| 118h | DCOC TZA DAC Step 2 (DCOC_TZA_STEP_2) | 32 | RW | 0000_0000h |
| 11Ch | DCOC TZA DAC Step 3 (DCOC_TZA_STEP_3) | 32 | RW | 0000_0000h |
| 120h | DCOC TZA DAC Step 4 (DCOC_TZA_STEP_4) | 32 | RW | 0000_0000h |
| 124h | DCOC TZA DAC Step 5 (DCOC_TZA_STEP_5) | 32 | RW | 0000_0000h |
| 128h | DCOC TZA DAC Step 6 (DCOC_TZA_STEP_6) | 32 | RW | 0000_0000h |
| 12Ch | DCOC TZA DAC Step 7 (DCOC_TZA_STEP_7) | 32 | RW | 0000_0000h |
| 130h | DCOC TZA DAC Step 5 (DCOC_TZA_STEP_8) | 32 | RW | 0000_0000h |
| 134h | DCOC TZA DAC Step 9 (DCOC_TZA_STEP_9) | 32 | RW | 0000_0000h |
| 138h | DCOC TZA DAC Step 10 (DCOC_TZA_STEP_10) | 32 | RW | 0000_0000h |
| 160h | DCOC Calibration Fail Thresholds (DCOC_CAL_FAIL_TH) | 32 | RW | 0000_0000h |
| 164h | DCOC Calibration Pass Thresholds (DCOC_CAL_PASS_TH) | 32 | RW | 0000_0000h |
| 168h | DCOC Calibration Alpha (DCOC_CAL_ALPHA) | 32 | RO | 0000_0000h |
| 16Ch | DCOC Calibration Beta Q (DCOC_CAL_BETA_Q) | 32 | RO | 0000_0000h |
| 170h | DCOC Calibration Beta I (DCOC_CAL_BETA_I) | 32 | RO | 0000_0000h |

*Table continues on the next page...*

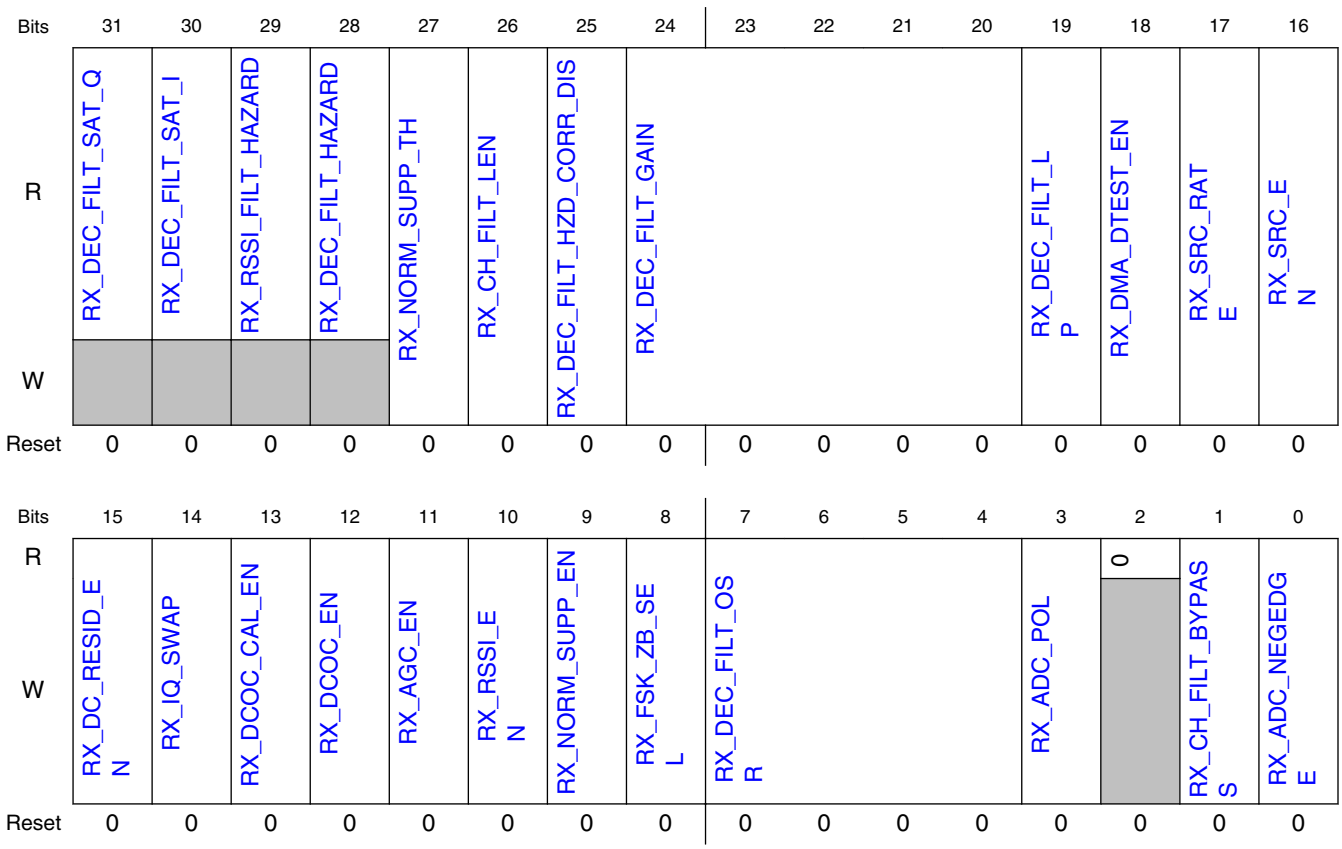
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 174h | DCOC Calibration Gamma (DCOC_CAL_GAMMA) | 32 | RO | 0000_0000h |
| 178h | DCOC Calibration IIR (DCOC_CAL_IIR) | 32 | RW | 0000_0000h |
| 180h - 188h | DCOC Calibration Result (DCOC_CAL1 - DCOC_CAL3) | 32 | RO | 0000_0000h |
| 190h | RX_DIG CCA ED LQI Control Register 0 (CCA_ED_LQI_CTRL_0) | 32 | RW | 0000_0000h |
| 194h | RX_DIG CCA ED LQI Control Register 1 (CCA_ED_LQI_CTRL_1) | 32 | RW | 0000_0000h |
| 198h | RX_DIG CCA ED LQI Status Register 0 (CCA_ED_LQI_STAT_0) | 32 | RO | 0000_0000h |
| 1A0h | Receive Channel Filter Coefficient 0 (RX_CHF_COEF_0) | 32 | RW | 0000_0000h |
| 1A4h | Receive Channel Filter Coefficient 1 (RX_CHF_COEF_1) | 32 | RW | 0000_0000h |
| 1A8h | Receive Channel Filter Coefficient 2 (RX_CHF_COEF_2) | 32 | RW | 0000_0000h |
| 1ACh | Receive Channel Filter Coefficient 3 (RX_CHF_COEF_3) | 32 | RW | 0000_0000h |
| 1B0h | Receive Channel Filter Coefficient 4 (RX_CHF_COEF_4) | 32 | RW | 0000_0000h |
| 1B4h | Receive Channel Filter Coefficient 5 (RX_CHF_COEF_5) | 32 | RW | 0000_0000h |
| 1B8h | Receive Channel Filter Coefficient 6 (RX_CHF_COEF_6) | 32 | RW | 0000_0000h |
| 1BCh | Receive Channel Filter Coefficient 7 (RX_CHF_COEF_7) | 32 | RW | 0000_0000h |
| 1C0h | Receive Channel Filter Coefficient 8 (RX_CHF_COEF_8) | 32 | RW | 0000_0000h |
| 1C4h | Receive Channel Filter Coefficient 9 (RX_CHF_COEF_9) | 32 | RW | 0000_0000h |
| 1C8h | Receive Channel Filter Coefficient 10 (RX_CHF_COEF_10) | 32 | RW | 0000_0000h |
| 1CCh | Receive Channel Filter Coefficient 11 (RX_CHF_COEF_11) | 32 | RW | 0000_0000h |
| 1D0h | AGC Manual AGC Index (AGC_MAN_AGC_IDX) | 32 | RW | 0000_0000h |
| 1D4h | DC Residual Control (DC_RESID_CTRL) | 32 | RW | 0000_0000h |
| 1D8h | DC Residual Estimate (DC_RESID_EST) | 32 | RO | 0000_0000h |
| 1DCh | RX RC Calibration Control0 (RX_RCCAL_CTRL0) | 32 | RW | 0000_0000h |
| 1E0h | RX RC Calibration Control1 (RX_RCCAL_CTRL1) | 32 | RW | 0000_0000h |
| 1E4h | RX RC Calibration Status (RX_RCCAL_STAT) | 32 | RO | 0210_4210h |
| 1E8h | Aux PLL Frequency Calibration Control (AUXPLL_FCAL_CTRL) | 32 | RW | 0040_0000h |
| 1ECh | Aux PLL Frequency Calibration Count 6 (AUXPLL_FCAL_CNT6) | 32 | RO | 0000_0000h |
| 1F0h | Aux PLL Frequency Calibration Count 5 and 4 (AUXPLL_FCAL_CNT5_4) | 32 | RO | 0000_0000h |
| 1F4h | Aux PLL Frequency Calibration Count 3 and 2 (AUXPLL_FCAL_CNT3_2) | 32 | RO | 0000_0000h |
| 1F8h | Aux PLL Frequency Calibration Count 1 and 0 (AUXPLL_FCAL_CNT1_0) | 32 | RO | 0000_0000h |

## A.2.3.2 RX Digital Control (RX_DIG_CTRL)

### A.2.3.2.1 Offset

| Register | Offset |
|---|---|
| RX_DIG_CTRL | 0h |

### A.2.3.2.2 Diagram




### A.2.3.2.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_DEC_FILT_SAT_Q | Decimator output, saturation detected for Q channel<br>This bit will be set if a saturation condition is detected in the decimator filter Q channel. This bit is cleared by tsm_rx_dcoc_init and rx_init.<br>0b - A saturation condition has not occurred.<br>1b - A saturation condition has occurred. |
| 30<br>RX_DEC_FILT_SAT_I | Decimator output, saturation detected for I channel<br>This bit will be set if a saturation condition is detected in the decimator filter I channel. This bit is cleared by tsm_rx_dcoc_init and rx_init.<br>0b - A saturation condition has not occurred. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 1b - A saturation condition has occurred. |
| 29<br>RX_RSSI_FILT_HAZARD | Decimator output for RSSI, hazard condition detected<br>This bit will be set if a hazard condition is detected in either the I or Q decimator filter related to the wideband RSSI measurement. This does not indicate that a data corruption has occurred, only that the conditions associated with data corruption were detected. This bit is cleared by rx_init.<br>0b - A hazard condition has not been detected<br>1b - A hazard condition has been detected |
| 28<br>RX_DEC_FILT_HAZARD | Decimator output, hazard condition detected<br>This bit will be set if a hazard condition is detected in either the I or Q decimator filter. This does not indicate that a data corruption has occurred, only that the conditions associated with data corruption were detected. This bit is cleared by rx_init.<br>0b - A hazard condition has not been detected<br>1b - A hazard condition has been detected |
| 27<br>RX_NORM_SUPP_TH | Normalizer Suppression Threshold<br>Configures the threshold for suppressing the normalizer's phase output when RX_NORM_SUPP_EN is set.<br>0b - Normalizer suppression threshold is 12'd7.<br>1b - Normalizer suppression threshold is 12'd15. |
| 26<br>RX_CH_FILT_LEN | RX Channel Filter Length<br>0b - Channel filter length is 24.<br>1b - Channel filter length is 16. Only RX_CHF_COEF_4 - RX_CHF_COEF_11 are used in this mode. |
| 25<br>RX_DEC_FILT_HZD_CORR_DIS | Decimator filter hazard correction disable<br>The decimator uses an asynchronous FIFO between the data coming out of the ADC (in the ADC clock domain) and the decimator output (in the ref osc clock domain, with the rest of the radio digital logic). The hazard detection logic checks for a hazard condition when the ADC clock domain logic may be about to write into the same FIFO location where the ref osc clock domain logic is about to read. If hazard correction is not disabled, the FIFO pointers will be adjusted on detection of a hazard condition.<br>0b - Hazard correction is enabled<br>1b - Hazard correction is disabled |
| 24-20<br>RX_DEC_FILT_GAIN | Decimation Filter Fractional Gain<br>Defines the fractional gain which is applied at the decimator output. The format is u0.5. The gain applied is 1 + RX_DEC_FILT_GAIN/32, so e.g. if RX_DEC_FILT_GAIN is 5'b10110=5'd22, the gain is 1+22/32=1.6875. The nominal gain through the ADC is 2048codes/1.7V; the combined gain through the ADC and decimator would therefore be 2048codes/1.7e3mV*1.6875=2.03codes/mV for the example RX_DEC_FILT_GAIN=5'b10110. |
| 19<br>RX_DEC_FILT_LP | RX Decimator Low Power<br>This bit can be set to reduce the power in the decimator.<br>0b - Decimator operates in normal mode.<br>1b - Decimator operates in low power mode. |
| 18<br>RX_DMA_DTEST_EN | RX DMA and DTEST enable<br>This bit should be set to ensure that all of the rx_dig outputs related to DMA or DTEST are enabled. In mission mode this bit is intended to be cleared to reduce switching power. |
| 17<br>RX_SRC_RATE | RX Sample Rate Converter Rate Selections<br>0b - SRC is configured for a First Order Hold rate of 8/13.<br>1b - SRC is configured for a Zero Order Hold rate of 12/13. |
| 16<br>RX_SRC_EN | RX Sample Rate Converter Enable<br>0b - SRC is disabled.<br>1b - SRC is enabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15<br>RX_DC_RESID_EN | DC Residual Enable<br>Enables DC Residual block<br>0b - DC Residual block is disabled.<br>1b - DC Residual block is enabled. |
| 14<br>RX_IQ_SWAP | RX IQ Swap<br>Enable swap of I/Q channels (does not affect ADC raw mode).<br>0b - IQ swap is disabled.<br>1b - IQ swap is enabled. |
| 13<br>RX_DCOC_CAL_EN | DCOC Calibration Enable<br>Enable DCOC warm-up calibration in receiver.<br>0b - DCOC calibration is disabled.<br>1b - DCOC calibration is enabled. |
| 12<br>RX_DCOC_EN | DCOC Enable<br>Enables DCO calculation and application of corrections.<br>0b - DCOC is disabled.<br>1b - DCOC is enabled. |
| 11<br>RX_AGC_EN | AGC Global Enable<br>Does NOT affect user gains (user gain programming has priority).<br>0b - AGC is disabled.<br>1b - AGC is enabled. |
| 10<br>RX_RSSI_EN | RSSI Measurement Enable<br>0b - RSSI measurement is disabled.<br>1b - RSSI measurement is enabled. |
| 9<br>RX_NORM_SUPP_EN | Normalizer Suppression Enable<br>0b - Normalizer suppression is disabled.<br>1b - Normalizer suppression is enabled. |
| 8<br>RX_FSK_ZB_SEL | FSK / 802.15.4 demodulator select<br>Select between FSK and 802.15.4 demodulator. This is used in the RSSI and AGC to select trigger source input signals.<br>0b - FSK demodulator.<br>1b - 802.15.4 demodulator. |
| 7-4<br>RX_DEC_FILT_OSR | Decimation Filter Oversampling<br>**NOTE:** All undocumented values are Reserved.<br>0000b - OSR 4<br>0001b - OSR 8<br>0010b - OSR 16<br>0011b - OSR 6<br>0100b - OSR 32<br>0101b - OSR 12<br>0110b - OSR 24<br>0111b - OSR 48<br>1000b - OSR 64 |
| 3<br>RX_ADC_POL | Receive ADC Polarity<br>Selects polarity of the ADC data<br>0b - ADC output of 1'b0 maps to -1, 1'b1 maps to +1 (default)<br>1b - ADC output of 1'b0 maps to +1, 1'b1 maps to -1 |
| 2<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 1<br>RX_CH_FILT_BYPASS | Receive Channel Filter Bypass<br>Selects whether to disable and bypass channel filter.<br>0b - Channel filter is enabled.<br>1b - Disable and bypass channel filter. |
| 0<br>RX_ADC_NEGEDGE | Receive ADC Negative Edge Selection<br>Selects which edge of the clock the ADC data is registered.<br>0b - Register ADC data on positive edge of clock<br>1b - Register ADC data on negative edge of clock |

## A.2.3.3 AGC Control 0 (AGC_CTRL_0)

### A.2.3.3.1 Offset

| Register | Offset |
|---|---|
| AGC_CTRL_0 | 4h |

### A.2.3.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AGC_DOWN_RSSI_THRESH | | | | | | | | AGC_UP_RSSI_THRESH | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AGC_DOWN_LNA_STEP_SZ | | | | AGC_DOWN_BBA_STEP_SZ | | | | AGC_UP_SRC | AGC_UP_EN | AGC_FREEZE_SRC | | AGC_FREEZE_EN | SLOW_AGC_SRC | | SLOW_AGC_EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.3.3 Fields

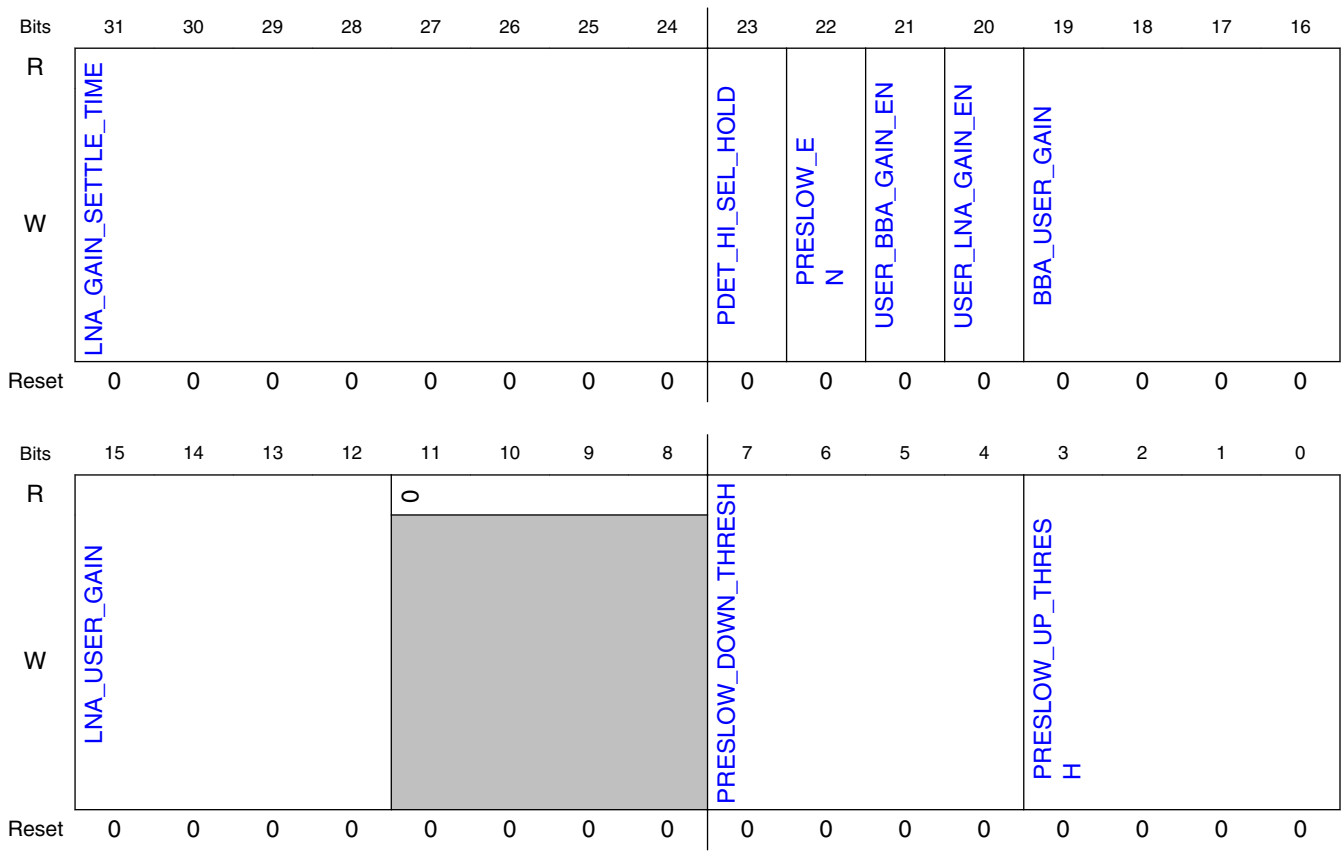
| Field | Function |
|---|---|
| 31-24<br>AGC_DOWN_RSSI_THRESH | AGC DOWN RSSI Threshold<br>ADC RSSI threshold to take downward step (AGC slow). If the ADC RSSI measurment is higher than this threshold, a downward gain step can be taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 23-16<br>AGC_UP_RSSI_THRESH | AGC UP RSSI Threshold<br>ADC RSSI threshold to take upward step (AGC slow). If the ADC RSSI measurment is below this threshold, an upward gain step can be taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 15-12<br>AGC_DOWN_LNA_STEP_SZ | AGC_DOWN_LNA_STEP_SZ<br>Number of table steps for downward step (LNA) in AGC fast. |
| 11-8<br>AGC_DOWN_BBA_STEP_SZ | AGC_DOWN_BBA_STEP_SZ<br>Number of table steps for downward step (BBA) in AGC fast. |
| 7<br>AGC_UP_SRC | AGC Up Source<br>Criterion to use for upward AGC steps in SLOW state. For pdet_lo, a timing window is observed for no assertions of the LO peak detectors and then an upward step is made. If RSSI is selected, the current RSSI measurement is compared with the UP threshold to determine if an upward step is due.<br>0b - PDET LO<br>1b - RSSI |
| 6<br>AGC_UP_EN | AGC Up Enable<br>Allow AGC to take upward steps in the gain table in slow mode. |
| 5-4<br>AGC_FREEZE_SRC | AGC Freeze Source Selection<br>Select trigger source for entering freeze AGC (HOLD state).<br>00b - Access Address match (for active protocol)<br>01b - Preamble Detect (for active protocol)<br>10b - PD confirmation / Access Address match (for active protocol) |
| 3<br>AGC_FREEZE_EN | AGC Freeze Enable<br>Allow AGC to freeze (ie enter HOLD state). AGC can still go to hold mode if timer expires (same as fast expire) from slow mode. |
| 2-1<br>SLOW_AGC_SRC | Slow AGC Source Selection<br>Select trigger source for entering slow AGC. For address match and preamble detect, the trigger is generated by the PHY. If the Fast AGC expire timer is selected, when the timer expires, the AGC state machine will transition into SLOW.<br>00b - Access Address match (for active protocol)<br>01b - Preamble Detect (for active protocol)<br>10b - Fast AGC expire timer<br>11b - Reserved |
| 0<br>SLOW_AGC_EN | Slow AGC Enable<br>Allow AGC to enter into slow mode. |

# A.2.3.4 AGC Control 1 (AGC_CTRL_1)

## A.2.3.4.1 Offset

| Register | Offset |
|---|---|
| AGC_CTRL_1 | 8h |

## A.2.3.4.2 Diagram




## A.2.3.4.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LNA_GAIN_SETTLE_TIME | LNA_GAIN_SETTLE_TIME<br>At LNA gain change, number of clocks to assert TZA peak detector reset (for automatic control). Should be programmed greater than zero. |
| 23 | AGC HOLD hysteresis |

*Table continues on the next page...*

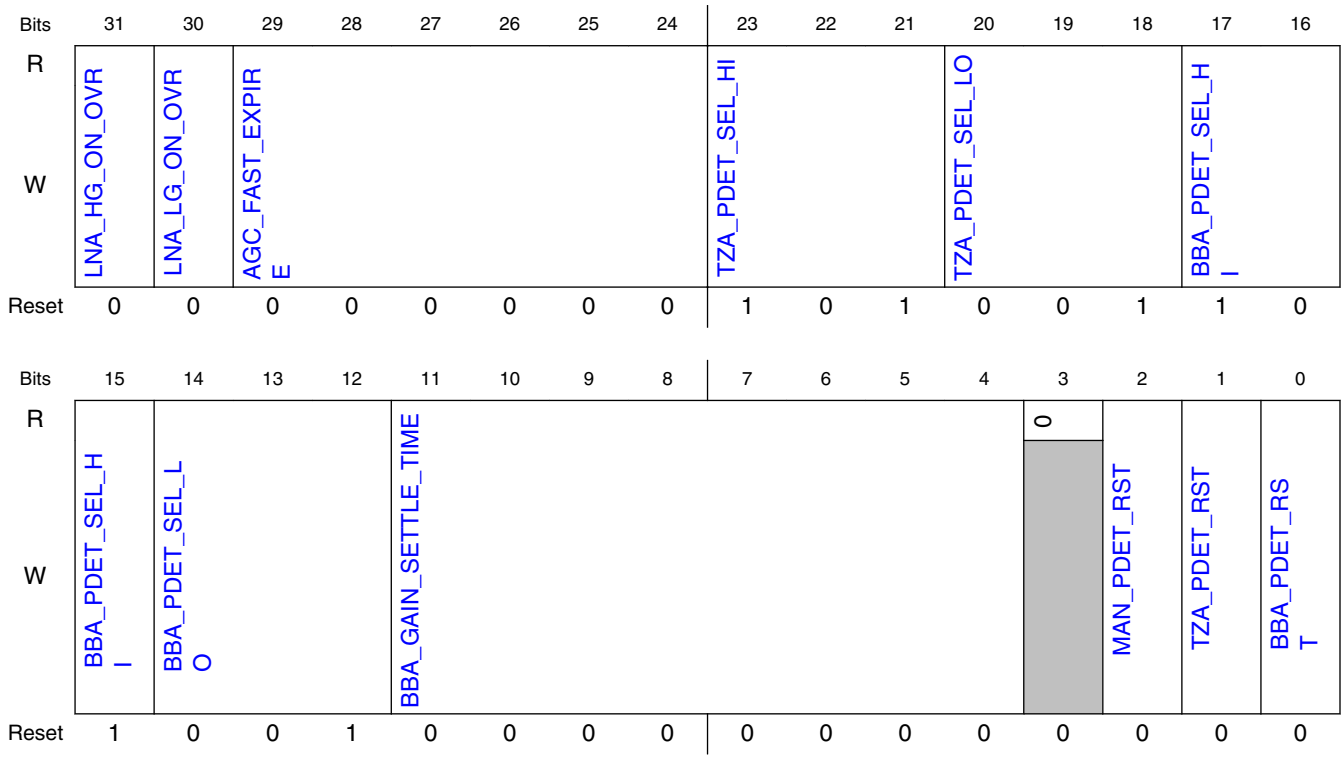
| Field | Function |
|---|---|
| PDET_HI_SEL_HOLD | When enabled, and the AGC state machine is in HOLD, the BBA clip detector threshold is raised by 1 count.<br>0b - Disabled.<br>1b - Enabled. |
| 22<br>PRESLOW_EN | Pre-slow Enable<br>Enable the pre-slow state where signal headroom is checked before entering SLOW.<br>0b - Pre-slow is disabled.<br>1b - Pre-slow is enabled. |
| 21<br>USER_BBA_GAIN_EN | User BBA Gain Enable<br>Enable user defined BBA gain (no AGC). |
| 20<br>USER_LNA_GAIN_EN | User LNA Gain Enable<br>Enable user defined LNA gain (no AGC). |
| 19-16<br>BBA_USER_GAIN | BBA_USER_GAIN<br>User defined BBA gain index if user_bba_gain_en =1 |
| 15-12<br>LNA_USER_GAIN | LNA_USER_GAIN<br>User defined LNA gain index if user_lna_gain_en =1. |
| 11-8<br>— | Reserved. |
| 7-4<br>PRESLOW_DOWN_THRESH | PRESLOW_DOWN_THRESH<br>ADC RSSI threshold to take downward step in (AGC PRESLOW state). The actual threshold is computed as the programmed value minus 10, so the threshold assumes values in the range [-10,5]. If the ADC RSSI measurment is above this threshold, a downward gain step of 1 is taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |
| 3-0<br>PRESLOW_UP_THRESH | PRESLOW_UP_THRESH<br>ADC RSSI threshold to take upward step in (AGC PRESLOW state). The actual threshold is computed as the programmed value minus 15, so the threshold assumes values in the range [-15,0]. If the ADC RSSI measurment is below this threshold, an upward gain step of 1 is taken. The ADC RSSI is measured directly from the ADC input and does not have any adjustment applied for gain settings, etc. In other words, it is a measurement of how saturated the digital bus is, similar to a dBFS measurement. |

## A.2.3.5 AGC Control 2 (AGC_CTRL_2)

### A.2.3.5.1 Offset

| Register | Offset |
|---|---|
| AGC_CTRL_2 | Ch |

## A.2.3.5.2 Diagram




## A.2.3.5.3 Fields

| Field | Function |
|---|---|
| 31<br>LNA_HG_ON_OVR | LNA_HG_ON override<br>If set, the lna high gain "on" signal is always asserted. Otherwise, this signal is asserted automatically based on the LNA_GAIN code. |
| 30<br>LNA_LG_ON_OVR | LNA_LG_ON override<br>If set, the lna low gain "on" signal is always asserted. Otherwise, this signal is asserted automatically based on the LNA_GAIN code. |
| 29-24<br>AGC_FAST_EXPIRE | AGC Fast Expire<br>Expire time (uS) for fast AGC (1-63uS). |
| 23-21<br>TZA_PDET_SEL_HI | TZA PDET Threshold High<br>TZA peak detect HI threshold.<br>000b - 0.600V<br>001b - 0.645V<br>010b - 0.705V<br>011b - 0.750V<br>100b - 0.795V<br>101b - 0.855V<br>110b - 0.900V<br>111b - 0.945V |
| 20-18 | TZA PDET Threshold Low |

*Table continues on the next page...*

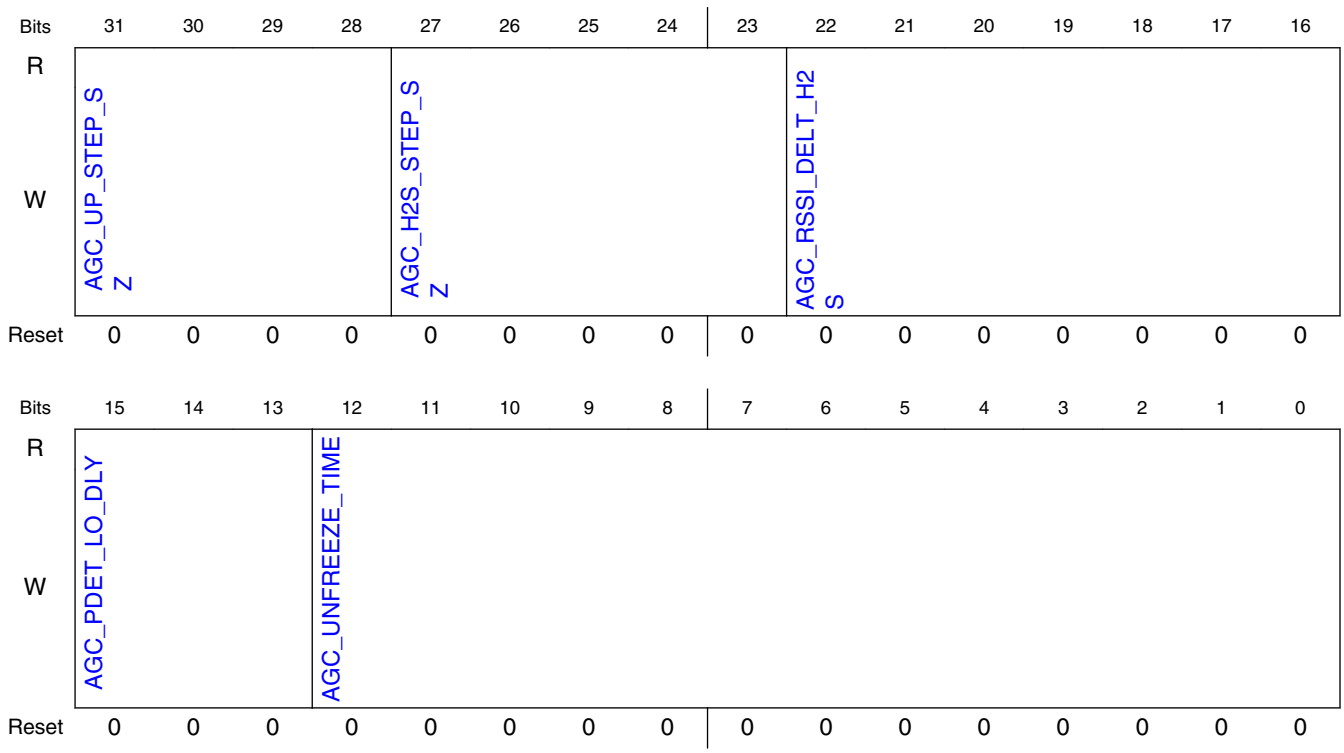
| Field | Function |
|---|---|
| TZA_PDET_SEL_LO | TZA peak detect LO threshold.<br>000b - 0.600V<br>001b - 0.615V<br>010b - 0.630V<br>011b - 0.645V<br>100b - 0.660V<br>101b - 0.675V<br>110b - 0.690V<br>111b - 0.705V |
| 17-15<br>BBA_PDET_SEL_HI | BBA PDET Threshold High<br>BBA peak detect HI threshold.<br>000b - 0.600V<br>001b - 0.795V<br>010b - 0.900V<br>011b - 0.945V<br>100b - 1.005V<br>101b - 1.050V<br>110b - 1.095V<br>111b - 1.155V |
| 14-12<br>BBA_PDET_SEL_LO | BBA PDET Threshold Low<br>BBA peak detect LO threshold.<br>000b - 0.600V<br>001b - 0.615V<br>010b - 0.630V<br>011b - 0.645V<br>100b - 0.660V<br>101b - 0.675V<br>110b - 0.690V<br>111b - 0.705V |
| 11-4<br>BBA_GAIN_SETTLE_TIME | BBA Gain Settle Time<br>Number of clocks to assert BBA peak detector reset (for automatic control). Should be programmed greater than zero. |
| 3<br>— | Reserved. |
| 2<br>MAN_PDET_RST | MAN PDET Reset<br>0b - The peak detector reset signals are controlled automatically by the AGC.<br>1b - The BBA_PDET_RST and TZA_PDET_RST are used to manually control the peak detector reset signals. |
| 1<br>TZA_PDET_RST | TZA PDET Reset<br>TZA peak detector reset, manual control. |
| 0<br>BBA_PDET_RST | BBA PDET Reset<br>BBA peak detector reset, manual control. |

## A.2.3.6 AGC Control 3 (AGC_CTRL_3)

## A.2.3.6.1 Offset

| Register | Offset |
|---|---|
| AGC_CTRL_3 | 10h |

## A.2.3.6.2 Diagram




## A.2.3.6.3 Fields

| Field | Function |
|---|---|
| 31-28<br>AGC_UP_STEP_SZ | AGC Up Step Size<br>Number of AGC gain table steps for upward step |
| 27-23<br>AGC_H2S_STEP_SZ | AGC_H2S_STEP_SZ<br>AGC gain table step size for hold to slow jump. |
| 22-16<br>AGC_RSSI_DELT_H2S | AGC_RSSI_DELT_H2S<br>RSSI delta that causes hold to slow transition. This delta is observed from a previous RSSI measurement to a current measurement. |
| 15-13 | AGC Peak Detect Low Delay<br>Time (uS) to wait for pdet low to assert (1-7uS). |

*Table continues on the next page...*

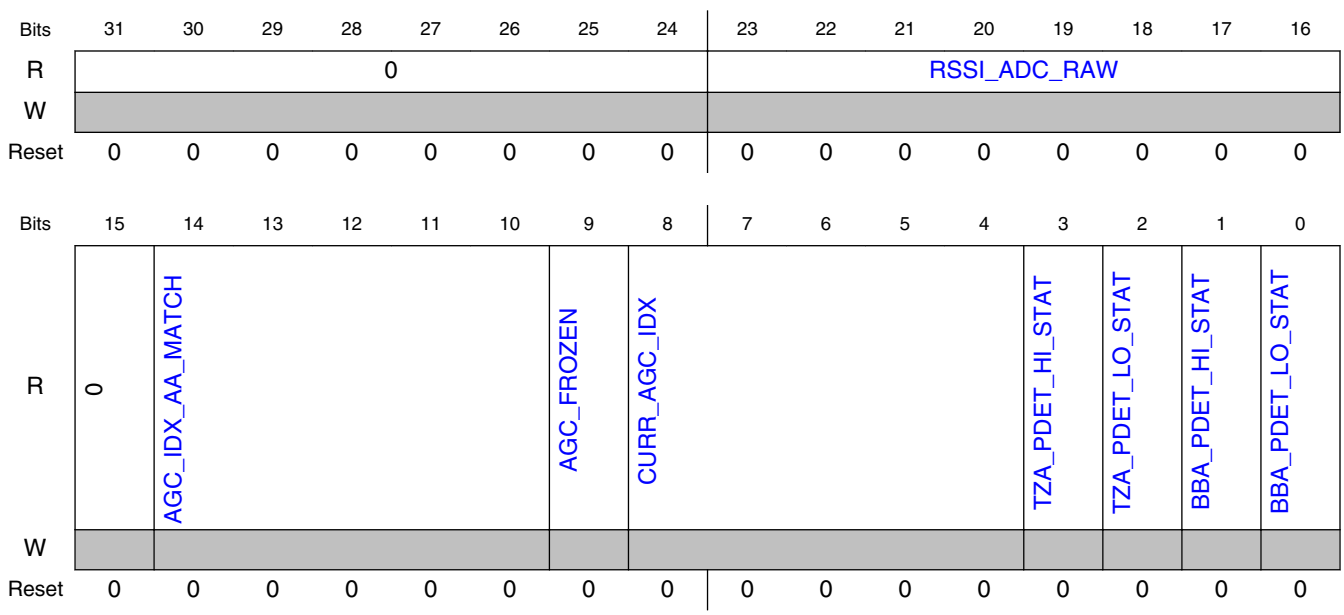
| Field | Function |
|---|---|
| AGC_PDET_LO_DLY | |
| 12-0<br>AGC_UNFREEZE_TIME | AGC Unfreeze Time<br>Time (uS) for AGC to unfreeze (1-8191uS) from HOLD and re-enter SLOW state. |

## A.2.3.7 AGC Status (AGC_STAT)

### A.2.3.7.1 Offset

| Register | Offset |
|---|---|
| AGC_STAT | 14h |

### A.2.3.7.2 Diagram




### A.2.3.7.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |

*Table continues on the next page...*

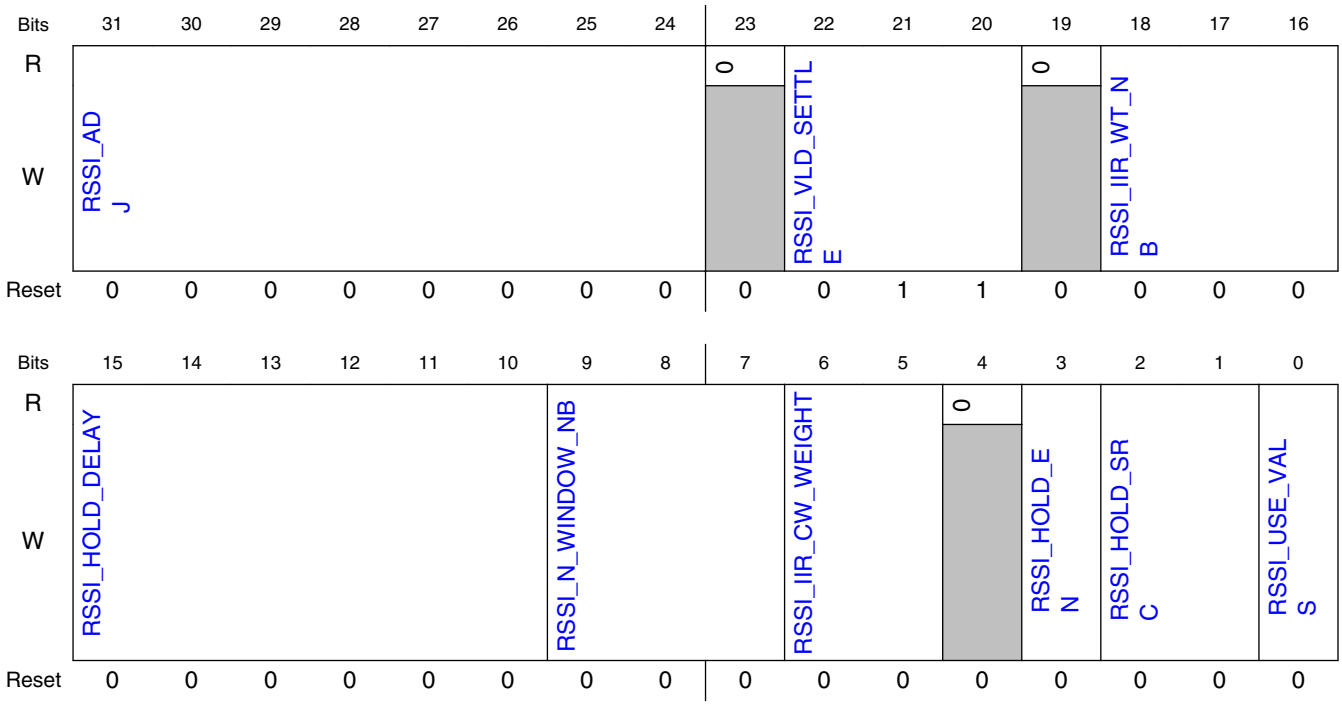
| Field | Function |
|---|---|
| 23-16<br>RSSI_ADC_RAW | ADC RAW RSSI Reading<br>Reading of ADC rssi (before adjustments) |
| 15<br>— | Reserved. |
| 14-10<br>AGC_IDX_AA_MATCH | AGC Gain Index at AA Match<br>AGC Gain Index captured at AA Match. This value remains static so it can be read by software after the end of the RX burst. The value is cleared to 0 at TSM rx_init and radio reset conditions. |
| 9<br>AGC_FROZEN | AGC Frozen Status<br>Status of AGC freeze.<br>0b - AGC is not frozen.<br>1b - AGC is frozen. |
| 8-4<br>CURR_AGC_IDX | Current AGC Gain Index<br>Current AGC gain table index |
| 3<br>TZA_PDET_HI_STAT | TZA Peak Detector High Status<br>Status of TZA peak detector HI flag (1=set) |
| 2<br>TZA_PDET_LO_STAT | TZA Peak Detector Low Status<br>Status of TZA peak detector LO flag (1=set) |
| 1<br>BBA_PDET_HI_STAT | BBA Peak Detector High Status<br>Status of BBA peak detector HI flag (1=set) |
| 0<br>BBA_PDET_LO_STAT | BBA Peak Detector Low Status<br>Status of BBA peak detector LO flag (1=set) |

## A.2.3.8 RSSI Control 0 (RSSI_CTRL_0)

### A.2.3.8.1 Offset

| Register | Offset |
|---|---|
| RSSI_CTRL_0 | 18h |

## A.2.3.8.2 Diagram




## A.2.3.8.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RSSI_ADJ | RSSI Adjustment<br>RSSI calculation adjustment (8-bit signed 1/4 dB). |
| 23<br>— | Reserved. |
| 22-20<br>RSSI_VLD_SETTLE | RSSI Valid Settle<br>Sets number of us (times 8) that RSSI will be considered invalid after certain events. |
| 19<br>— | Reserved. |
| 18-16<br>RSSI_IIR_WT_NB | RSSI IIR Weighting Narrowband<br>IIR filter weight for RSSI filtering for Narrowband mode.<br>**NOTE:** All undocumented values are Reserved.<br>000b - Bypass<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32 |
| 15-10 | RSSI Hold Delay |

*Table continues on the next page...*

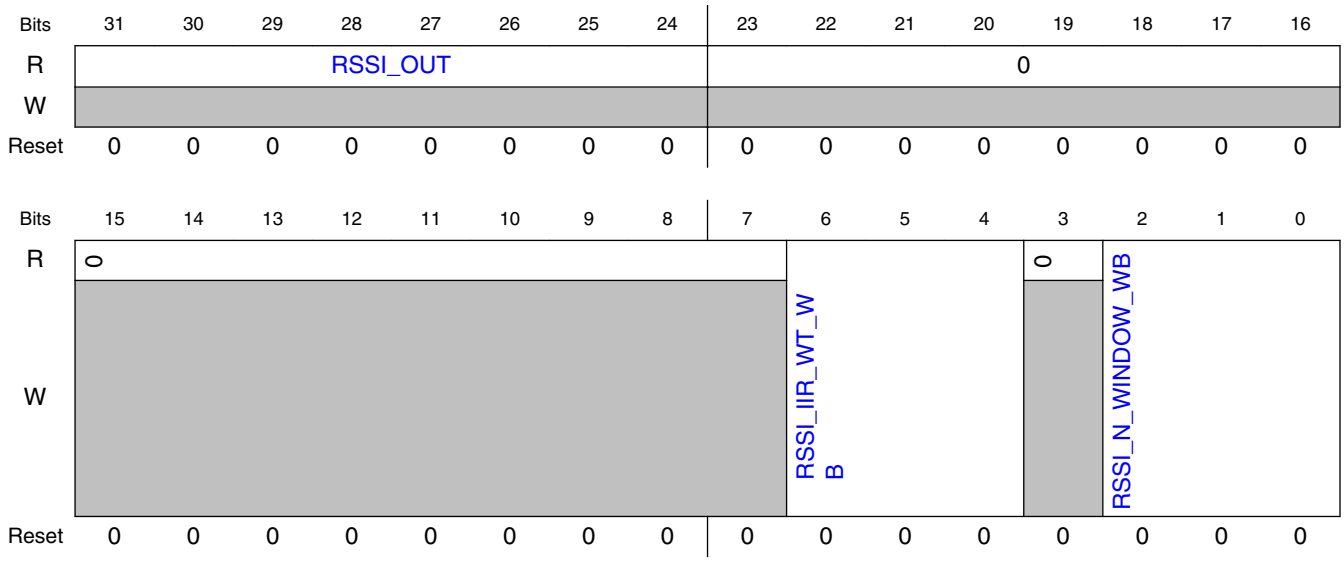
| Field | Function |
|---|---|
| RSSI_HOLD_DELAY | Sets number of us (times 8) that RSSI will run after a hold event before the value is frozen. |
| 9-7<br>RSSI_N_WINDOW_NB | RSSI N Window Average Narrowband<br>Selects Averaging window length for RSSI in Narrowband mode.<br>000b - No averaging<br>001b - Averaging window length is 2 samples<br>010b - Averaging window length is 4 samples<br>011b - Averaging window length is 8 samples<br>100b - Averaging window length is 16 samples<br>101b - Averaging window length is 32 samples |
| 6-5<br>RSSI_IIR_CW_WEIGHT | RSSI IIR CW Weighting<br>IIR filter weight for RSSI filtering of a CW input.<br>00b - Bypass<br>01b - 1/8<br>10b - 1/16<br>11b - 1/32 |
| 4<br>— | Reserved. |
| 3<br>RSSI_HOLD_EN | RSSI Hold Enable<br>Enable RSSI to freeze after hold criterion met. RSSI will still be briefly held when a gain change occurs. |
| 2-1<br>RSSI_HOLD_SRC | RSSI Hold Source Selection<br>Select trigger source for freezing RSSI measurement.<br>00b - Access Address match<br>01b - Preamble Detect<br>10b - Reserved<br>11b - 802.15.4 LQI done (1=freeze, 0=run AGC) |
| 0<br>RSSI_USE_VALS | RSSI Values Selection<br>Enable use of LNA and BBA gain values programmed in registers for calculation. |

## A.2.3.9 RSSI Control 1 (RSSI_CTRL_1)

### A.2.3.9.1 Offset

| Register | Offset |
|---|---|
| RSSI_CTRL_1 | 1Ch |

## A.2.3.9.2 Diagram




## A.2.3.9.3 Fields

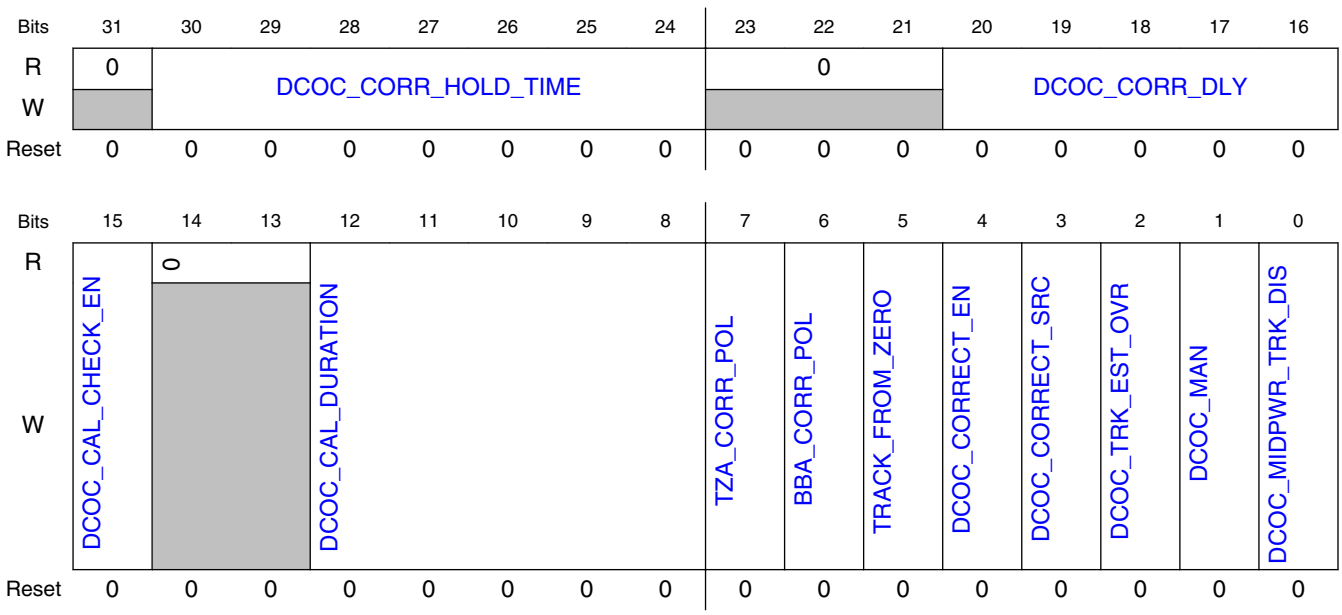
| Field | Function |
|---|---|
| 31-24<br>RSSI_OUT | RSSI Reading<br>RSSI output (8-bit signed). |
| 23-7<br>— | Reserved. |
| 6-4<br>RSSI_IIR_WT_WB | RSSI IIR Weighting Wideband<br>IIR filter weight for RSSI filtering in Wideband mode.<br>**NOTE:** All undocumented values are Reserved.<br>000b - Bypass<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32 |
| 3<br>— | Reserved. |
| 2-0<br>RSSI_N_WINDOW_WB | RSSI N Window Average Wideband<br>Selects Averaging window length for RSSI in Wideband mode.<br>000b - No averaging<br>001b - Averaging window length is 2 samples<br>010b - Averaging window length is 4 samples<br>011b - Averaging window length is 8 samples<br>100b - Averaging window length is 16 samples<br>101b - Averaging window length is 32 samples |

# A.2.3.10 DCOC Control 0 (DCOC_CTRL_0)

## A.2.3.10.1 Offset

| Register | Offset |
|---|---|
| DCOC_CTRL_0 | 24h |

## A.2.3.10.2 Diagram




## A.2.3.10.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>DCOC_CORR_HOLD_TIME | DCOC Correction Hold Time<br>Delay from last gain change to freezing the DC correction.<br>0000000b - Reserved<br>0000001-1111110b - For a 32MHz reference clock, this is the delay in microseconds; for other reference clock frequencies, the delay is scaled accordingly.<br>1111111b - The DC correction is not frozen. |
| 23-21<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 20-16<br>DCOC_CORR_DLY | DCOC Correction Delay<br>Wait time between corrections.<br>00000b - Reserved<br>00001-11111b - For a 32MHz reference clock, this is the wait time in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 15<br>DCOC_CAL_CHECK_EN | DCOC Calibration Check Enable<br>0b - Calibration checking disabled. The DCOC_OFFSET_n registers are always updated during calibration.<br>1b - Calibration checking enabled. The DCOC_OFFSET_n registers are updated conditionally depending on the outcome of the pass/fail threshold checks performed on the alpha-hat and beta-hat estimates during calibration. |
| 14-13<br>— | Reserved. |
| 12-8<br>DCOC_CAL_DURATION | DCOC Calibration Duration<br>00000b - Reserved<br>00001-11111b - For a 32MHz reference clock, this is the calibration duration in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 7<br>TZA_CORR_POL | TZA Correction Polarity<br>Selects polarity of TZA corrections.<br>0b - Normal polarity.<br>1b - Negative polarity. This should be set if the ADC output is inverted, or if the TZA DACs were implemented with negative polarity. |
| 6<br>BBA_CORR_POL | BBA Correction Polarity<br>Selects polarity of BBA corrections.<br>0b - Normal polarity.<br>1b - Negative polarity. This should be set if the ADC output is inverted, or if the BBA DACs were implemented with negative polarity. |
| 5<br>TRACK_FROM_ZERO | Track from Zero<br>Selects whether the tracking estimator resets its DC estimate on every AGC gain change to zero or uses the current I/Q sample.<br>0b - Track from current I/Q sample.<br>1b - Track from zero. |
| 4<br>DCOC_CORRECT_EN | DCOC Correction Enable<br>0b - Correction disabled. The DCOC will not correct the DC offset.<br>1b - Correction enabled. The DCOC will use the TZA and BBA DACs, and apply digital corrections (if DCOC_CORRECT_SRC=1) to correct the DC offset. |
| 3<br>DCOC_CORRECT_SRC | DCOC Corrector Source<br>If not set, the corrector uses only the DCOC calibration table to apply corrections to the DCOC DACs.<br>0b - If correction is enabled, the DCOC will use only the DCOC calibration table to correct the DC offset.<br>1b - If correction is enabled, the DCOC will use the DCOC calibration table and then the tracking estimator to correct the DC offset. |
| 2<br>DCOC_TRK_EST_OVR | Override for the DCOC tracking estimator<br>0b - The tracking estimator is enabled only as needed by the corrector<br>1b - The tracking estimator remains enabled whenever the DCOC is active |
| 1<br>DCOC_MAN | DCOC Manual Override<br>If the manual override bit is set, it forces the DCOC to use the DAC and digital correction values from registers DCOC_DAC_INIT and DCOC_DIG_MAN, respectively. |
| 0 | DCOC Mid Power Tracking Disable |

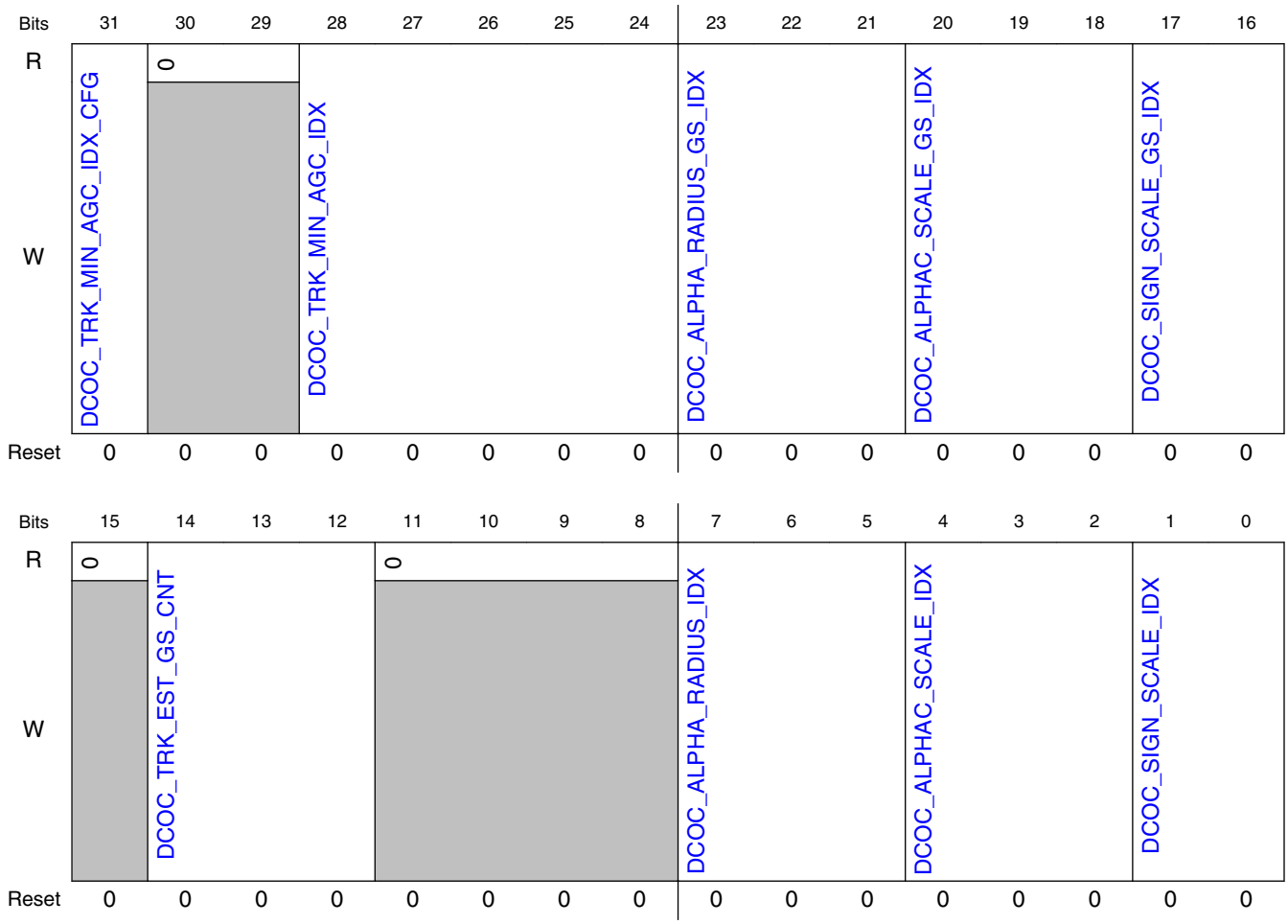
| Field | Function |
|---|---|
| DCOC_MIDPWR_TRK_DIS | Disables tracking correction at mid power levels, as indicated by the TZA and BBA lo peak detectors. This is implemented by reseting the counters associated with DCOC_CORR_DLY and DCOC_CORR_HOLD_TIME, so if the lo peak detectors do not continue to assert, tracking corrections would resume operation. The tracking estimator is not disabled and the counter associated with DCOC_TRK_EST_GS_CNT is not reset when this condition occurs.<br>0b - Tracking corrections are enabled as determined by DCOC_CORRECT_SRC and DCOC_TRK_MIN_AGC_IDX.<br>1b - Tracking corrections are disabled when either the TZA or BBA lo peak detector asserts. |

## A.2.3.11 DCOC Control 1 (DCOC_CTRL_1)

### A.2.3.11.1 Offset

| Register | Offset |
|---|---|
| DCOC_CTRL_1 | 28h |

## A.2.3.11.2 Diagram




## A.2.3.11.3 Fields

| Field | Function |
|---|---|
| 31<br>DCOC_TRK_MIN_AGC_IDX_CFG | DCOC_TRK_MIN_AGC_IDX Configuration<br>0b - Tracking is disabled when the AGC index is less than DCOC_TRK_MIN_AGC_IDX<br>1b - Tracking is enabled when AGC index is less than DCOC_TRK_MIN_AGC_IDX, but DCOC_CORR_DLY_ALT and DCOC_CORR_HOLD_TIME_ALT are used instead of DCOC_CORR_DLY and DCOC_CORR_HOLD_TIME |
| 30-29<br>— | Reserved. |
| 28-24<br>DCOC_TRK_MIN_AGC_IDX | DCOC Tracking Minimum AGC Table Index<br>Specifies the minimum AGC table index value at which tracking is enabled. E.g., if this is 5'd0, then tracking is enabled for all AGC gain table indexes (assuming DCOC_CORRECT_SRC is set) if this is 5'd20, then tracking is enabled for AGC gain table indexes 20-26 (assuming DCOC_CORRECT_SRC is set). |
| 23-21 | Alpha-R Scaling for Gearshift |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DCOC_ALPHA_RADIUS_GS_IDX | DCOC Alpha-R Scaling for Gearshift. Radius stepsize used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>000b - 1<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32<br>110b - 1/64<br>111b - Reserved |
| 20-18<br>DCOC_ALPHAC_SCALE_GS_IDX | DCOC Alpha-C Scaling for Gearshift<br>DCOC Alpha-C Scaling for Gearshift. I/Q center stepsize used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>000b - 1/2<br>001b - 1/4<br>010b - 1/8<br>011b - 1/16<br>100b - 1/32<br>101b - 1/64<br>110b - Reserved<br>111b - Reserved |
| 17-16<br>DCOC_SIGN_SCALE_GS_IDX | DCOC Sign Scaling for Gearshift<br>DCOC Sign Scaling for Gearshift. Sign()-based scaling factor used in the DCOC tracking estimator after the gearshift counter has expired. Used when DCOC_CORRECT_SRC=1.<br>00b - 1/8<br>01b - 1/16<br>10b - 1/32<br>11b - 1/64 |
| 15<br>— | Reserved. |
| 14-12<br>DCOC_TRK_EST_GS_CNT | DCOC Tracking Estimator Gearshift Count<br>Specifies the number of corrections (periods of DCOC_CORR_DLY) before the tracking estimator switches from using the set of parameters {DCOC_ALPHA_RADIUS_IDX, DCOC_ALPHAC_SCALE_IDX, DCOC_SIGN_SCALE_IDX} to the set of gearshift parameters {DCOC_ALPHA_RADIUS_GS_IDX, DCOC_ALPHAC_SCALE_GS_IDX, DCOC_SIGN_SCALE_GS_IDX}. If the value is 0, the set of gearshift parameters are not used. |
| 11-8<br>— | Reserved. |
| 7-5<br>DCOC_ALPHA_RADIUS_IDX | Alpha-R Scaling<br>DCOC Alpha-R Scaling. Radius stepsize used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>000b - 1<br>001b - 1/2<br>010b - 1/4<br>011b - 1/8<br>100b - 1/16<br>101b - 1/32<br>110b - 1/64<br>111b - Reserved |
| 4-2 | DCOC Alpha-C Scaling |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DCOC_ALPHAC_SCALE_IDX | DCOC Alpha-C Scaling. I/Q center stepsize used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>000b - 1/2<br>001b - 1/4<br>010b - 1/8<br>011b - 1/16<br>100b - 1/32<br>101b - 1/64<br>110b - Reserved<br>111b - Reserved |
| 1-0<br>DCOC_SIGN_SCALE_IDX | DCOC Sign Scaling<br>DCOC Sign Scaling. Sign()-based scaling factor used in the DCOC tracking estimator. Used when DCOC_CORRECT_SRC=1.<br>00b - 1/8<br>01b - 1/16<br>10b - 1/32<br>11b - 1/64 |

## A.2.3.12 DCOC DAC Initialization (DCOC_DAC_INIT)

### A.2.3.12.1 Offset

| Register | Offset |
|---|---|
| DCOC_DAC_INIT | 2Ch |

### A.2.3.12.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TZA_DCOC_INIT_Q | | | | | | | | TZA_DCOC_INIT_I | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | BBA_DCOC_INIT_Q | | | | | | 0 | | BBA_DCOC_INIT_I | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.12.3 Fields

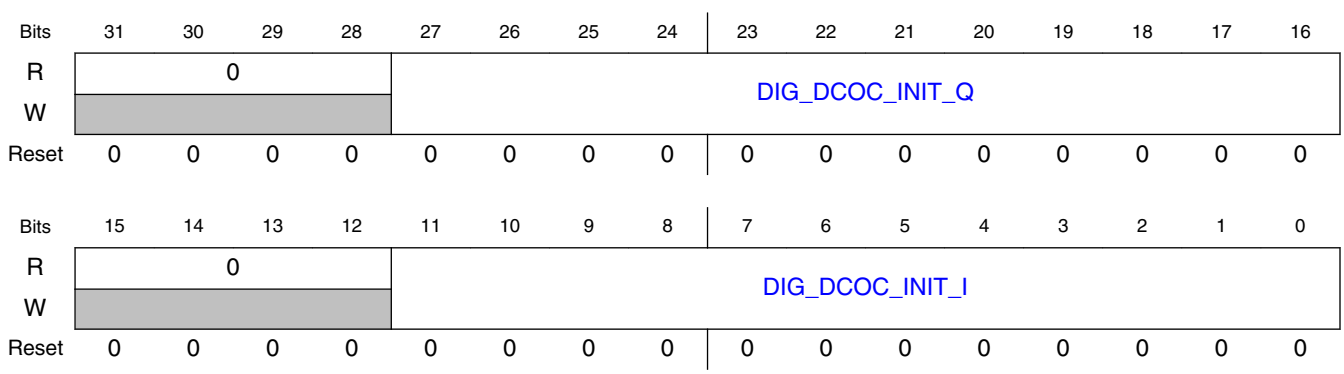
| Field | Function |
|---|---|
| 31-24<br>TZA_DCOC_INIT_Q | DCOC TZA Init Q<br>Value used for DCOC TZA Q channel DAC during calibration and for manual override. |
| 23-16<br>TZA_DCOC_INIT_I | DCOC TZA Init I<br>Value used for DCOC TZA I channel DAC during calibration and for manual override. |
| 15-14<br>— | Reserved. |
| 13-8<br>BBA_DCOC_INIT_Q | DCOC BBA Init Q<br>Value used for DCOC BBA Q channel DAC during calibration and for manual override. |
| 7-6<br>— | Reserved. |
| 5-0<br>BBA_DCOC_INIT_I | DCOC BBA Init I<br>Value used for DCOC BBA I channel DAC during calibration and for manual override. |

## A.2.3.13 DCOC Digital Correction Manual Override (DCOC_DIG_MAN)

### A.2.3.13.1 Offset

| Register | Offset |
|---|---|
| DCOC_DIG_MAN | 30h |

### A.2.3.13.2 Diagram



### A.2.3.13.3 Fields

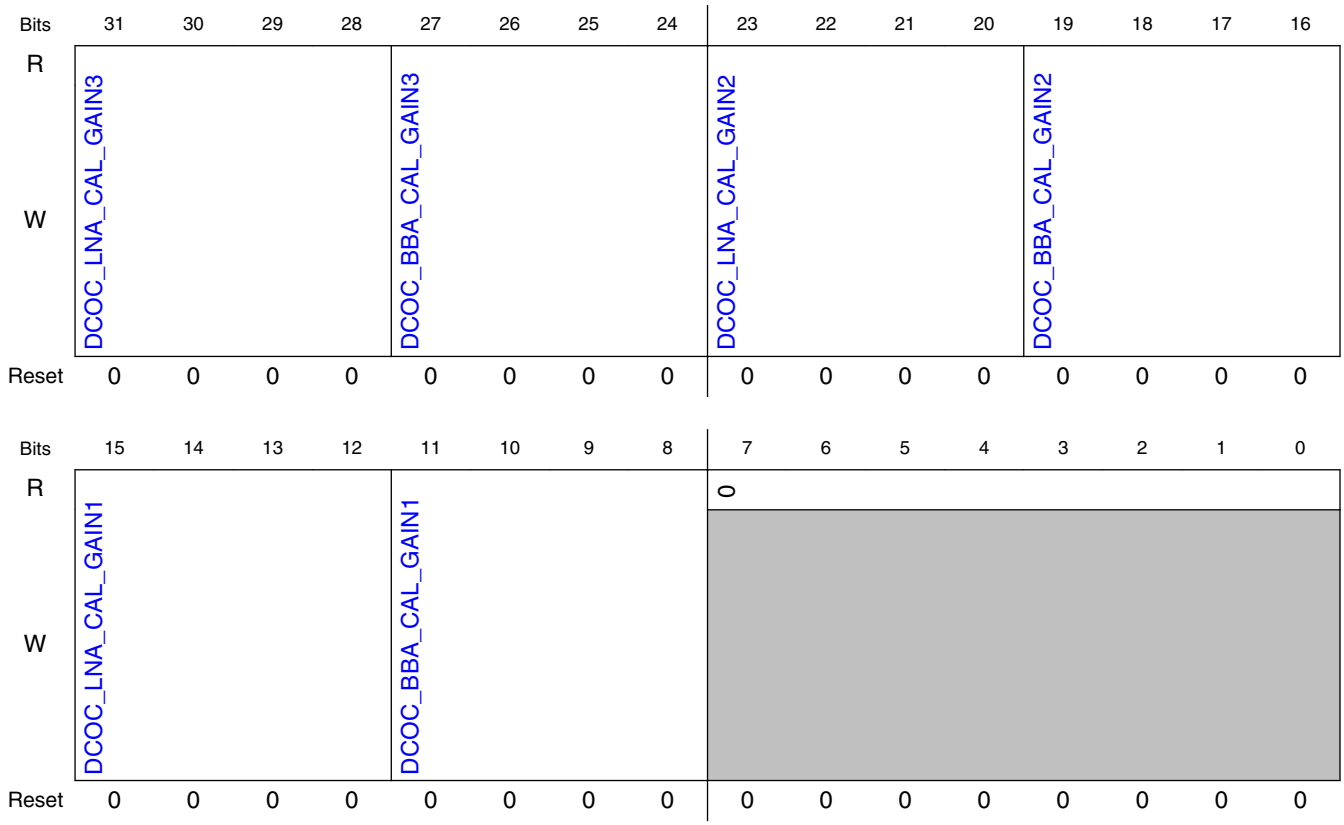
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DIG_DCOC_INIT_Q | DCOC DIG Init Q<br>Manual override for DCOC DIG Q channel correction. Value to be subtracted from downsampled Q channel. Used when DCOC_MAN=1. |
| 15-12<br>— | Reserved. |
| 11-0<br>DIG_DCOC_INIT_I | DCOC DIG Init I<br>Manual override for DCOC DIG I channel correction. Value to be subtracted from downsampled I channel. Used when DCOC_MAN=1. |

## A.2.3.14 DCOC Calibration Gain (DCOC_CAL_GAIN)

### A.2.3.14.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_GAIN | 34h |

## A.2.3.14.2 Diagram




## A.2.3.14.3 Fields

| Field | Function |
|---|---|
| 31-28<br>DCOC_LNA_CAL_GAIN3 | DCOC LNA Calibration Gain 3<br>The LNA gain index used for the 3rd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 27-24<br>DCOC_BBA_CAL_GAIN3 | DCOC BBA Calibration Gain 3<br>The BBA gain index used for the 3rd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 23-20<br>DCOC_LNA_CAL_GAIN2 | DCOC LNA Calibration Gain 2<br>The LNA gain index used for the 2nd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 19-16<br>DCOC_BBA_CAL_GAIN2 | DCOC BBA Calibration Gain 2<br>The BBA gain index used for the 2nd DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 15-12<br>DCOC_LNA_CAL_GAIN1 | DCOC LNA Calibration Gain 1<br>The LNA gain index used for the 1st DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 11-8 | DCOC BBA Calibration Gain 1 |

*Table continues on the next page...*

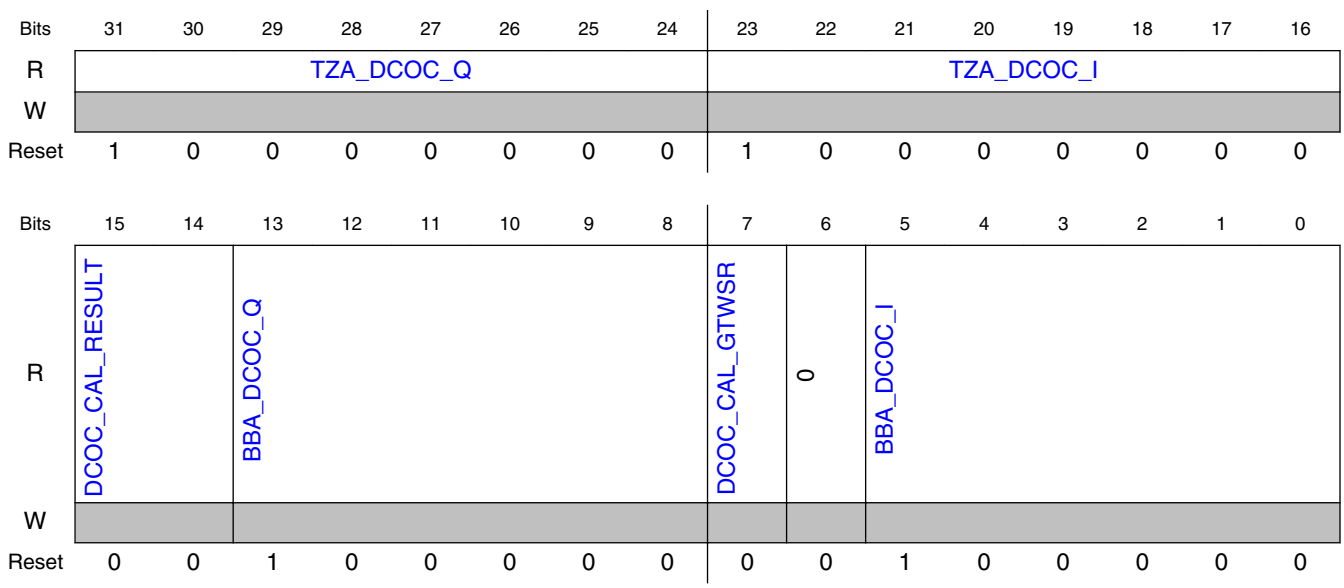
| Field | Function |
|---|---|
| DCOC_BBA_CAL_GAIN1 | The BBA gain index used for the 1st DCOC calibration point. Used when RX_DCOC_CAL_EN=1. |
| 7-0<br>— | Reserved. |

## A.2.3.15 DCOC Status (DCOC_STAT)

### A.2.3.15.1 Offset

| Register | Offset |
|---|---|
| DCOC_STAT | 38h |

### A.2.3.15.2 Diagram




### A.2.3.15.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TZA_DCOC_Q | DCOC TZA DAC Q<br>Current TZA DAC setting for Q channel. Note that the TZA DACs have a bias of 0x80; 0x0 represents the most negative DC offset, 0x80 represents a DC offset of 0, and 0xFF represents the most postiive DC offset. This is provided for debug and characterization purposes only. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>TZA_DCOC_I | DCOC TZA DAC I<br>Current TZA DAC setting for I channel. Note that the TZA DACs have a bias of 0x80; 0x0 represents the most negative DC offset, 0x80 represents a DC offset of 0, and 0xFF represents the most postiive DC offset. This is provided for debug and characterization purposes only. |
| 15-14<br>DCOC_CAL_RESULT | DCOC_CAL_RESULT<br>Indicates the calibration result. Only applicable when DCOC_CTRL_0[DCOC_CAL_CHECK_EN] is set.<br>00b - Calibration checks failed. DCOC_OFFSET_n tables not updated.<br>01b - Calibration checks neither passed nor failed, DCOC_OFFSET_n tables not updated.<br>10b - Calibration checks neither passed nor failed, DCOC_OFFSET_n tables updated since no previous Pass condition has occurred since the last radio reset.<br>11b - Calibration checks passed. DCOC_OFFSET_n tables updated |
| 13-8<br>BBA_DCOC_Q | DCOC BBA DAC Q<br>Current BBA DAC setting for Q channel. Note that the BBA DACs have a bias of 0x20; 0x0 represents the most negative DC offset, 0x20 represents a DC offset of 0, and 0x3F represents the most postiive DC offset. This is provided for debug and characterization purposes only. |
| 7<br>DCOC_CAL_GTWSR | DCOC calibration Good Table Written Since Reset<br>Indicates whether a Passing calibration check has occurred since the last reset. Only applicable when DCOC_CTRL_0[DCOC_CAL_CHECK_EN] is set.<br>0b - A Passing calibration result has not occurred since the last radio reset.<br>1b - A Passing calibration result has occurred since the last radio reset. |
| 6<br>— | Reserved. |
| 5-0<br>BBA_DCOC_I | DCOC BBA DAC I<br>Current BBA DAC setting for I channel. Note that the BBA DACs have a bias of 0x20; 0x0 represents the most negative DC offset, 0x20 represents a DC offset of 0, and 0x3F represents the most postiive DC offset. This is provided for debug and characterization purposes only. |

## A.2.3.16 DCOC DC Estimate (DCOC_DC_EST)

### A.2.3.16.1 Offset

| Register | Offset |
|---|---|
| DCOC_DC_EST | 3Ch |

### A.2.3.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DC_EST_Q | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DC_EST_I | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.16.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DC_EST_Q | DCOC DC Estimate Q<br>Reflects the current DCOC DC tracking estimate for Q channel. Format s11. Used when DCOC_CORRECT_SRC=1. This is provided for debug and characterization purposes only. |
| 15-12<br>— | Reserved. |
| 11-0<br>DC_EST_I | DCOC DC Estimate I<br>Reflects the current DCOC DC tracking estimate for I channel. Format s11. Used when DCOC_CORRECT_SRC=1. This is provided for debug and characterization purposes only. |

## A.2.3.17 DCOC Calibration Reciprocals (DCOC_CAL_RCP)

### A.2.3.17.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_RCP | 40h |

## A.2.3.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ALPHA_CALC_RECIP | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_TMP_CALC_RECIP | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.17.3 Fields

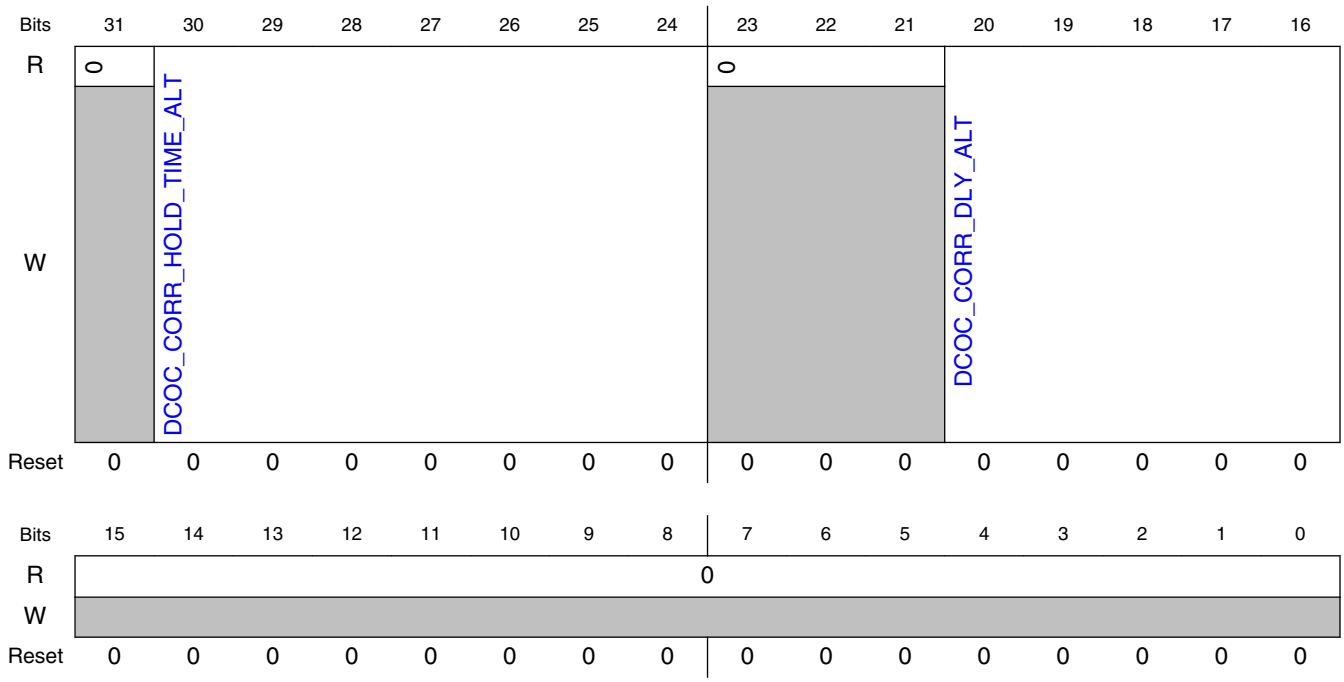
| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>ALPHA_CALC_RECIP | Alpha Calculation Reciprocal<br>DCOC Alpha calculation reciprocal (format: u.11). This is used in DCOC calibration calculation of the alpha DC component. It is defined as:<br>$1.0/((G_{L_HI} - G_{L_LO})*G_{B_LO})$<br>This is stored as with 11 fractional bits, so program the value<br>$round([1.0/((G_{L_HI} - G_{L_LO})*G_{B_LO})]*2^{11})$. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_TMP_CALC_RECIP | DCOC Calculation Reciprocal<br>DCOC_tmp calculation reciprocal (format: u1.10). This is used in DCDC calibration calculation. It is defined as<br>$1.0/(G_{B_HI} - G_{B_LO})$<br>This is stored with 10 fractional bits, so program the value $round([1.0/(G_{B_HI} - G_{B_LO})]*2^{10})$. |

# A.2.3.18 DCOC Control 2 (DCOC_CTRL_2)

## A.2.3.18.1 Offset

| Register | Offset |
|---|---|
| DCOC_CTRL_2 | 44h |

## A.2.3.18.2 Diagram




## A.2.3.18.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>DCOC_CORR_HOLD_TIME_ALT | DCOC Correction Hold Time Alternate<br>Delay from last gain change to freezing the DC correction. This is used instead of DCOC_CORR_HOLD_TIME depending on programming of DCOC_TRK_MIN_AGC_IDX_CFG and the current AGC index.<br>0000000b - Reserved<br>0000001-1111110b - For a 32MHz reference clock, this is the delay in microseconds; for other reference clock frequencies, the delay is scaled accordingly.<br>1111111b - The DC correction is not frozen. |
| 23-21<br>— | Reserved. |
| 20-16<br>DCOC_CORR_DLY_ALT | DCOC Correction Delay Alternate<br>Wait time between corrections. This is used instead of DCOC_CORR_DLY depending on programming of DCOC_TRK_MIN_AGC_IDX_CFG and the current AGC index.<br>00000b - Reserved<br>00001-11111b - For a 32MHz reference clock, this is the wait time in microseconds; for other reference clock frequencies, the delay is scaled accordingly. |
| 15-0<br>— | Reserved. |

## A.2.3.19 IQMC Control (IQMC_CTRL)

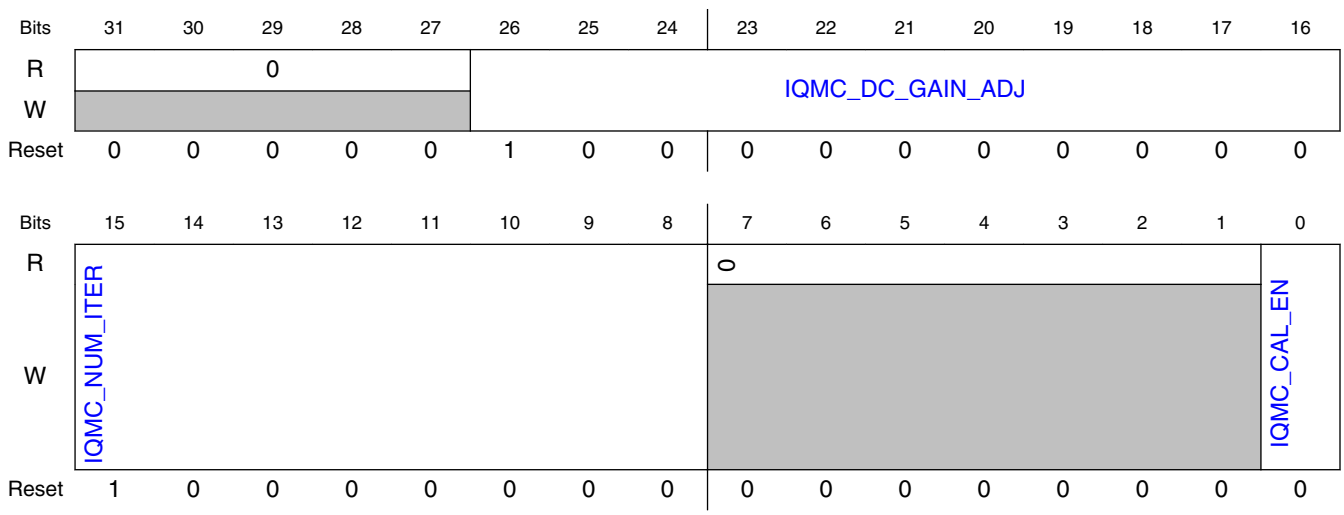
### A.2.3.19.1 Offset

| Register | Offset |
|---|---|
| IQMC_CTRL | 48h |

### A.2.3.19.2 Function

IQMC Control register. This register can only be accessed when the radio oscillator clock is active.

### A.2.3.19.3 Diagram




### A.2.3.19.4 Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>IQMC_DC_GAIN_ADJ | IQ Mismatch Correction DC Gain Coeff<br>I/Q mismatch correction DC gain coefficient. This gain value is used only during DCOC calibration. It is not calculated during the IQMC calibration sequence, so it must be written by software. The format is u1.10; the reset value of 0x400 corresponds to a DC gain coefficient of 1.0. |
| 15-8 | IQ Mismatch Cal Num Iter |

*Table continues on the next page...*

| Field | Function |
|---|---|
| IQMC_NUM_ITER | Number of iterations for IQ Mismatch Calibration. |
| 7-1<br>— | Reserved. |
| 0<br>IQMC_CAL_EN | IQ Mismatch Cal Enable<br>Enables IQ mismatch calibration. This bit is self-clearing; it will clear automatically at the end of the calibration sequence. |

## A.2.3.20 IQMC Calibration (IQMC_CAL)

### A.2.3.20.1 Offset

| Register | Offset |
|---|---|
| IQMC_CAL | 4Ch |

### A.2.3.20.2 Function

IQMC Calibration register. This register can only be accessed when the radio oscillator clock is active.

### A.2.3.20.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | IQMC_PHASE_ADJ | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | IQMC_GAIN_ADJ | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.20.4 Fields

| Field | Function |
|---|---|
| 31-28 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 27-16<br>IQMC_PHASE_ADJ | IQ Mismatch Correction Phase Coeff<br>I/Q mismatch correction phase coefficient. This value is updated automatically at the end of the IQMC calibration sequence. It can also be written by software. The format is signed, with the maximum positive value 0x7ff corresponding to a phase coefficient of 0.25, and the maximum negative value 0x800 corresponding to -0.25. The reset value of 0x000 corresponds to a phase coefficient of 0. |
| 15-11<br>— | Reserved. |
| 10-0<br>IQMC_GAIN_ADJ | IQ Mismatch Correction Gain Coeff<br>I/Q mismatch correction gain coefficient. This value is updated automatically at the end of the IQMC calibration sequence. It can also be written by software. The format is u1.10. The reset value of 0x400 corresponds to a gain coefficient of 1.0. |

# A.2.3.21 LNA_GAIN Step Values 3..0 (LNA_GAIN_VAL_3_0)

## A.2.3.21.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_3_0 | 50h |

## A.2.3.21.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_3 | | | | | | | | LNA_GAIN_VAL_2 | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_1 | | | | | | | | LNA_GAIN_VAL_0 | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

## A.2.3.21.3 Fields

| Field | Function |
|---|---|
| 31-24 | LNA_GAIN step 3<br>Gain for LNA gain Step 3, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LNA_GAIN_VAL_3 | |
| 23-16<br>LNA_GAIN_VAL_2 | LNA_GAIN step 2<br>Gain for LNA gain Step 2, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 15-8<br>LNA_GAIN_VAL_1 | LNA_GAIN step 1<br>Gain for LNA gain Step 1, used in RSSI calculation. Unsigned, 4*(measured_gain_dB+16). |
| 7-0<br>LNA_GAIN_VAL_0 | LNA_GAIN step 0<br>Gain for LNA gain Step 0, used in RSSI calculation. Unsigned, 4*(measured_gain_dB+16). |

## A.2.3.22 LNA_GAIN Step Values 7..4 (LNA_GAIN_VAL_7_4)

### A.2.3.22.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_7_4 | 54h |

### A.2.3.22.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_7 | | | | | | | | LNA_GAIN_VAL_6 | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_VAL_5 | | | | | | | | LNA_GAIN_VAL_4 | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

### A.2.3.22.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LNA_GAIN_VAL_7 | LNA_GAIN step 7<br>Gain for LNA gain Step 7, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>LNA_GAIN_VAL_6 | LNA_GAIN step 6<br>Gain for LNA gain Step 6, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 15-8<br>LNA_GAIN_VAL_5 | LNA_GAIN step 5<br>Gain for LNA gain Step 5, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |
| 7-0<br>LNA_GAIN_VAL_4 | LNA_GAIN step 4<br>Gain for LNA gain Step 4, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

## A.2.3.23 LNA_GAIN Step Values 8 (LNA_GAIN_VAL_8)

### A.2.3.23.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_VAL_8 | 58h |

### A.2.3.23.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LNA_GAIN_VAL_9 | | | | | | | | LNA_GAIN_VAL_8 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

### A.2.3.23.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8 | LNA_GAIN step 9<br>Gain for LNA gain Step 9, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

*Table continues on the next page...*

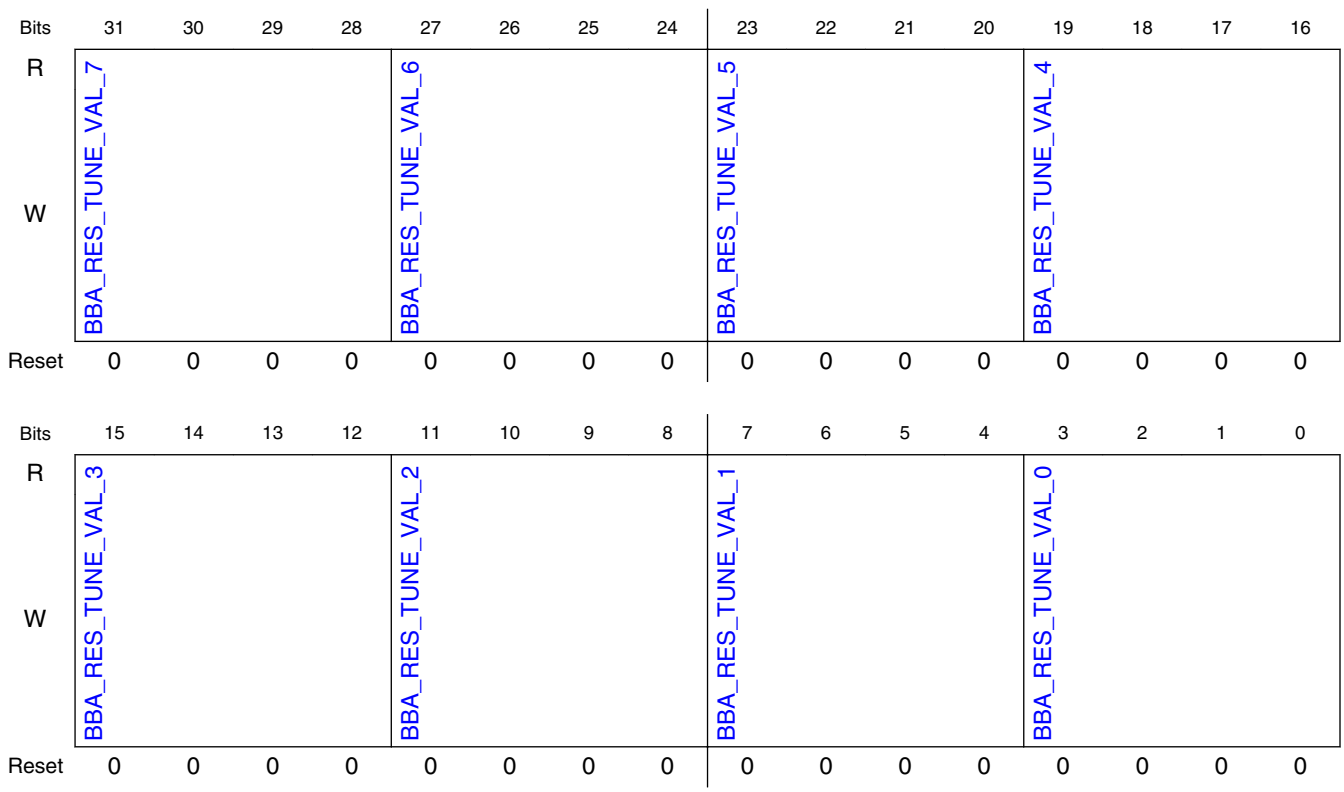
| Field | Function |
|---|---|
| LNA_GAIN_VAL_9 | |
| 7-0<br>LNA_GAIN_VAL_8 | LNA_GAIN step 8<br>Gain for LNA gain Step 8, used in RSSI calculation. Unsigned, 4*(measured_gain_dB). |

## A.2.3.24 BBA Resistor Tune Values 7..0 (BBA_RES_TUNE_VAL_7_0)

### A.2.3.24.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_VAL_7_0 | 5Ch |

### A.2.3.24.2 Diagram



### A.2.3.24.3 Fields

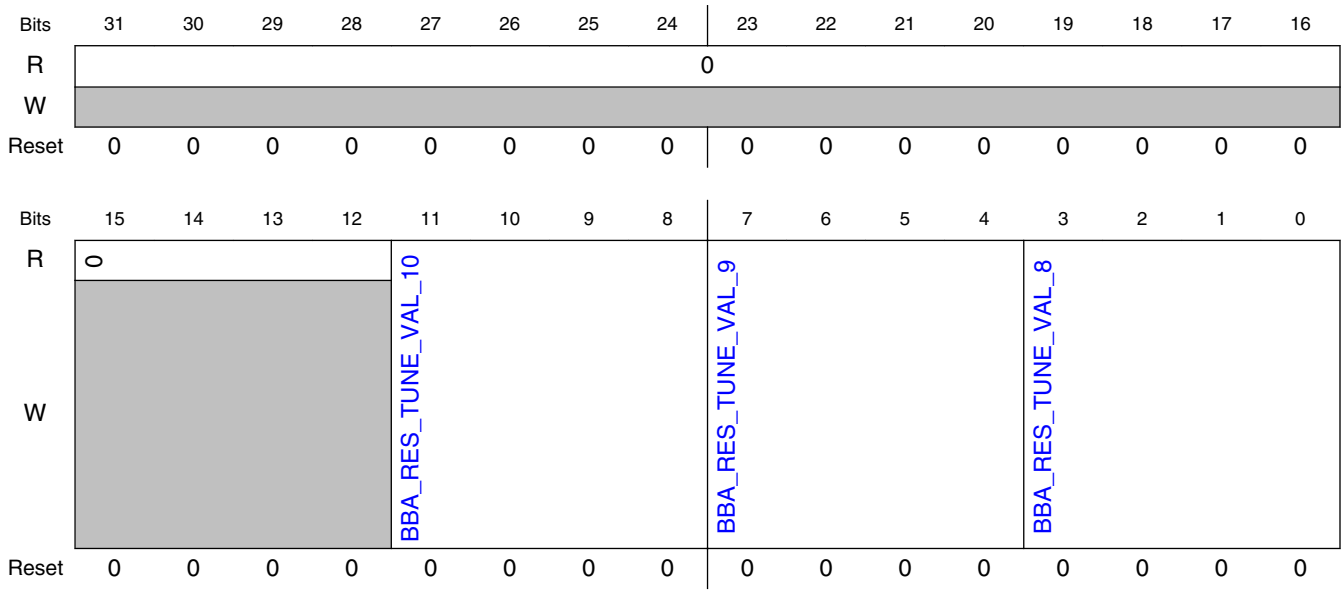
| Field | Function |
|---|---|
| 31-28<br>BBA_RES_TUNE_VAL_7 | BBA Resistor Tune Step 7<br>Gain offset for BBA gain Step 7, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 27-24<br>BBA_RES_TUNE_VAL_6 | BBA Resistor Tune Step 6<br>Gain offset for BBA gain Step 6, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 23-20<br>BBA_RES_TUNE_VAL_5 | BBA Resistor Tune Step 5<br>Gain offset for BBA gain Step 5, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 19-16<br>BBA_RES_TUNE_VAL_4 | BBA Resistor Tune Step 4<br>Gain offset for BBA gain Step 4, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 15-12<br>BBA_RES_TUNE_VAL_3 | BBA Resistor Tune Step 3<br>Gain offset for BBA gain Step 3, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 11-8<br>BBA_RES_TUNE_VAL_2 | BBA Resistor Tune Step 2<br>Gain offset for BBA gain Step 2, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 7-4<br>BBA_RES_TUNE_VAL_1 | BBA Resistor Tune Step 1<br>Gain offset for BBA gain Step 1, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 3-0<br>BBA_RES_TUNE_VAL_0 | BBA Resistor Tune Step 0<br>Gain offset for BBA gain Step 0, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |

## A.2.3.25 BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_VAL_10_8)

### A.2.3.25.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_VAL_10_8 | 60h |

### A.2.3.25.2 Diagram




### A.2.3.25.3 Fields

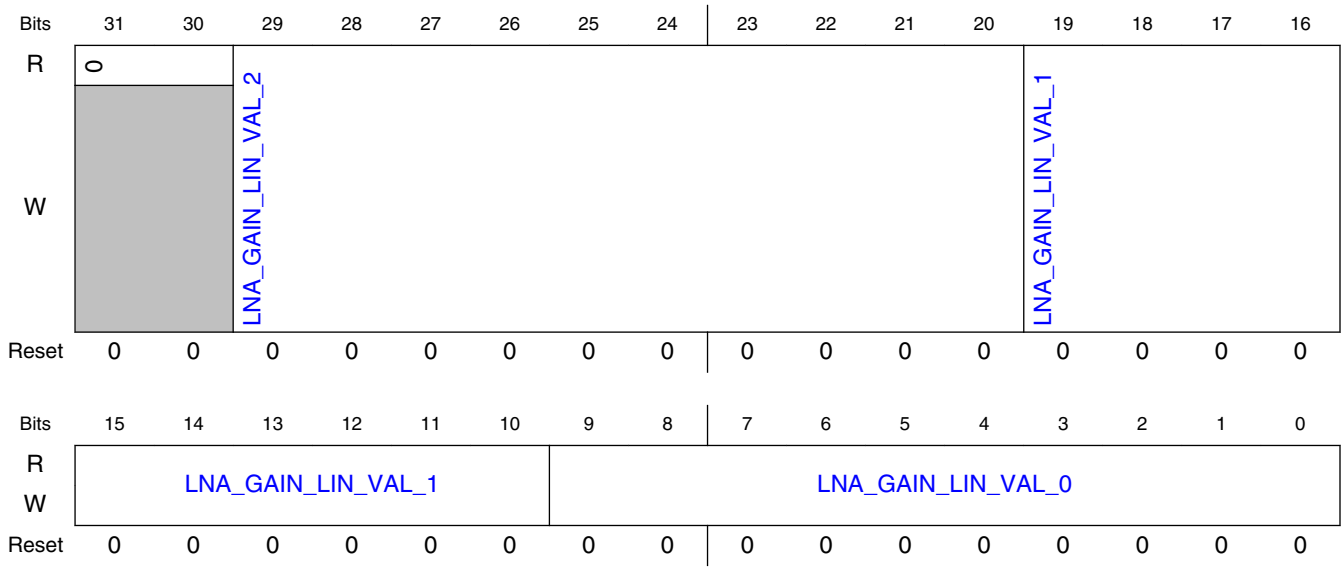
| Field | Function |
|---|---|
| 31-12<br>— | Reserved. |
| 11-8<br>BBA_RES_TUNE_VAL_10 | BBA Resistor Tune Step 10<br>Gain offset for BBA gain Step 10, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 7-4<br>BBA_RES_TUNE_VAL_9 | BBA Resistor Tune Step 9<br>Gain offset for BBA gain Step 9, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |
| 3-0<br>BBA_RES_TUNE_VAL_8 | BBA Resistor Tune Step 8<br>Gain offset for BBA gain Step 8, used in RSSI calculation. Signed, 2*(measured_gain_dB - spec_gain_dB). |

## A.2.3.26 LNA Linear Gain Values 2..0 (LNA_GAIN_LIN_VAL_2_0)

### A.2.3.26.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_2_0 | 64h |

## A.2.3.26.2 Diagram




## A.2.3.26.3 Fields

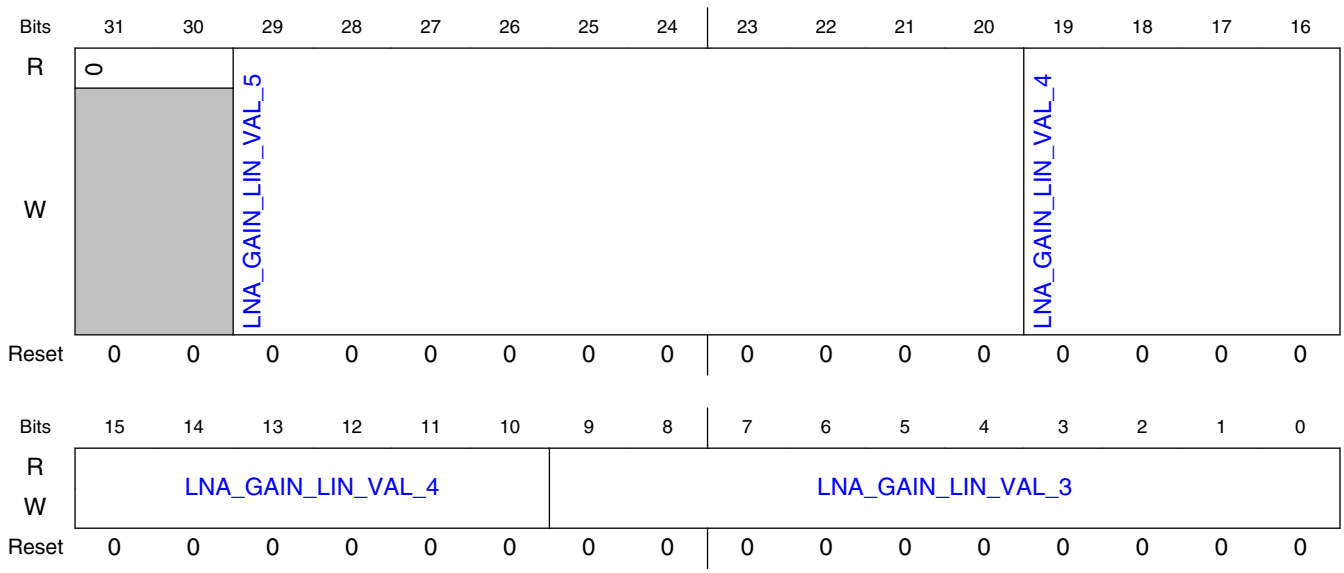
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_2 | LNA Linear Gain Step 2<br>LNA linear gain value for index 2, e.g. nominal value is 10^(2.2/20). Stored with 2 fractional bits, e.g. round([10^(2.2/20)]*2^2) = 5decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_1 | LNA Linear Gain Step 1<br>LNA linear gain value for index 1, e.g. nominal value is 10^(-3.5/20). Stored with 2 fractional bits, e.g. round([10^(-3.5/20)]*2^2) = 3decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_0 | LNA Linear Gain Step 0<br>LNA linear gain value for index 0, e.g. nominal value is 10^(-8.6/20). Stored with 2 fractional bits, e.g. round([10^(-8.6/20)]*2^2) = 1decimal. Format (8.2). |

# A.2.3.27 LNA Linear Gain Values 5..3 (LNA_GAIN_LIN_VAL_5_3)

## A.2.3.27.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_5_3 | 68h |

## A.2.3.27.2 Diagram




## A.2.3.27.3 Fields

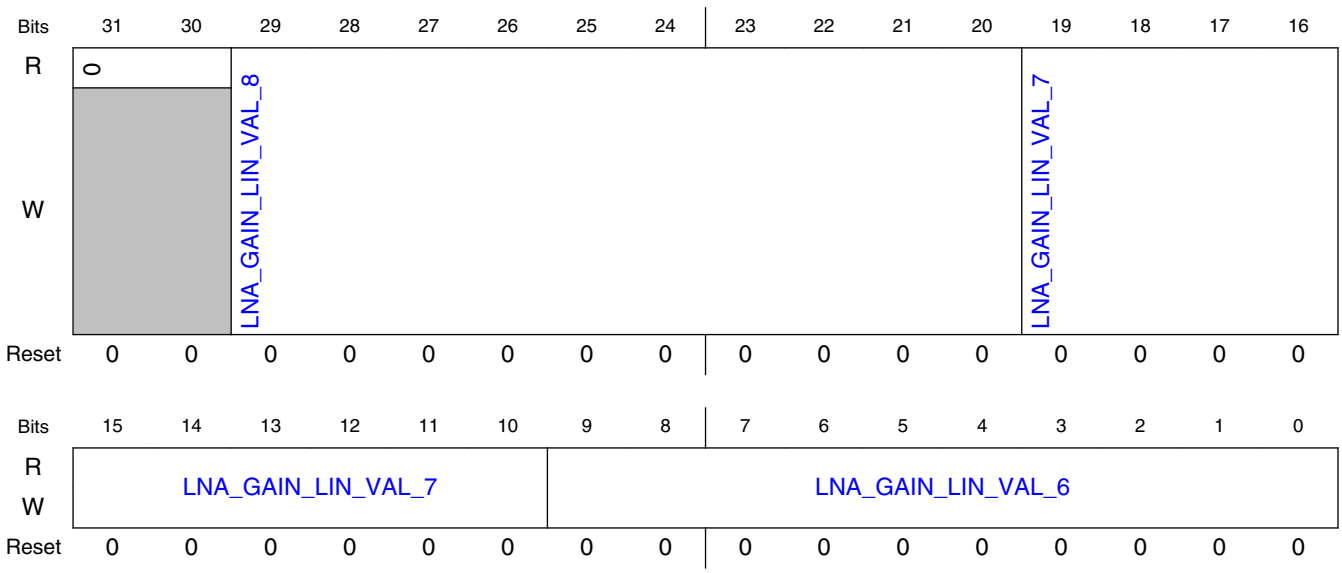
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_5 | LNA Linear Gain Step 5<br>LNA linear gain value for index 5, e.g. nominal value is 10^(22.7/20). Stored with 2 fractional bits, e.g. round([10^(22.7/20)]*2^2) = 55decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_4 | LNA Linear Gain Step 4<br>LNA linear gain value for index 4, e.g. nominal value is 10^(19.8/20). Stored with 2 fractional bits, e.g. round([10^(19.8/20)]*2^2) = 39decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_3 | LNA Linear Gain Step 3<br>LNA linear gain value for index 3, e.g. nominal value is 10^(13.9/20). Stored with 2 fractional bits, e.g. round([10^(13.9/20)]*2^2) = 20decimal. Format (8.2). |

# A.2.3.28 LNA Linear Gain Values 8..6 (LNA_GAIN_LIN_VAL_8_6)

## A.2.3.28.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_8_6 | 6Ch |

## A.2.3.28.2 Diagram




## A.2.3.28.3 Fields

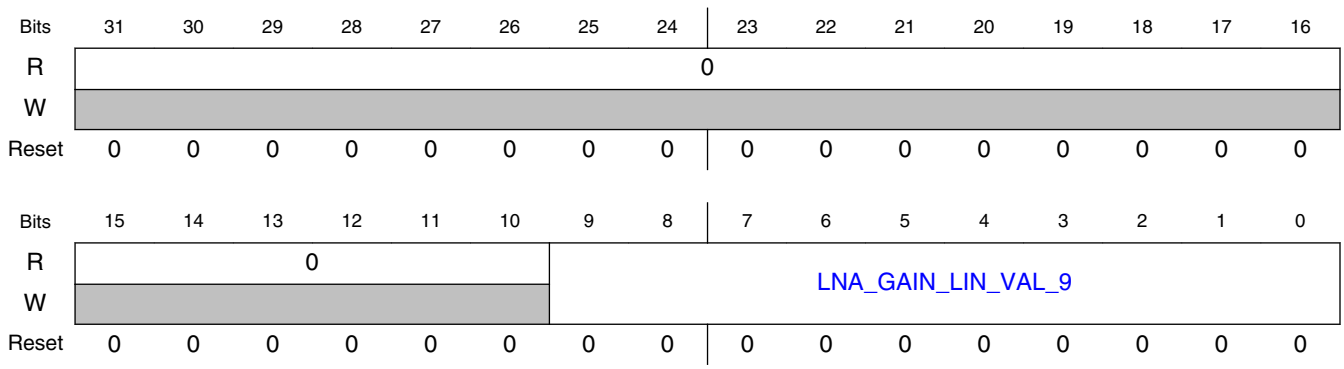
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>LNA_GAIN_LIN_VAL_8 | LNA Linear Gain Step 8<br>LNA linear gain value for index 8, e.g. nominal value is 10^(39.9/20). Stored with 2 fractional bits, e.g. round([10^(39.9/20)]*2^2) = 396decimal. Format (8.2). |
| 19-10<br>LNA_GAIN_LIN_VAL_7 | LNA Linear Gain Step 7<br>LNA linear gain value for index 7, e.g. nominal value is 10^(34.4/20). Stored with 2 fractional bits, e.g. round([10^(34.4/20)]*2^2) = 210decimal. Format (8.2). |
| 9-0<br>LNA_GAIN_LIN_VAL_6 | LNA Linear Gain Step 6<br>LNA linear gain value for index 6, e.g. nominal value is 10^(28.6/20). Stored with 2 fractional bits, e.g. round([10^(28.6/20)]*2^2) = 108decimal. Format (8.2). |

# A.2.3.29 LNA Linear Gain Values 9 (LNA_GAIN_LIN_VAL_9)

## A.2.3.29.1 Offset

| Register | Offset |
|---|---|
| LNA_GAIN_LIN_VAL_9 | 70h |

## A.2.3.29.2 Diagram




## A.2.3.29.3 Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>LNA_GAIN_LIN_VAL_9 | LNA Linear Gain Step 9<br>LNA linear gain value for index 9, e.g. nominal value is 10^(45.4/20). Stored with 2 fractional bits, e.g. round([10^(45.4/20)]*2^2) = 744decimal. Format (8.2). |

# A.2.3.30 BBA Resistor Tune Values 3..0 (BBA_RES_TUNE_LIN_VAL_3_0)

### A.2.3.30.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_3_0 | 74h |

### A.2.3.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_3 | | | | | | | | BBA_RES_TUNE_LIN_VAL_2 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_1 | | | | | | | | BBA_RES_TUNE_LIN_VAL_0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.30.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BBA_RES_TUNE_LIN_VAL_3 | BBA Resistor Tune Linear Gain Step 3<br>BBA linear gain value for index 3 (format: u5.3). Nominal value is 10^(9/20). Stored with 3 fractional bits, e.g. round([10^(9/20)]*2^3) = 23decimal |
| 23-16<br>BBA_RES_TUNE_LIN_VAL_2 | BBA Resistor Tune Linear Gain Step 2<br>BBA linear gain value for index 2 (format: u5.3). Nominal value is 10^(6/20). Stored with 3 fractional bits, e.g. round([10^(6/20)]*2^3) = 16decimal |
| 15-8<br>BBA_RES_TUNE_LIN_VAL_1 | BBA Resistor Tune Linear Gain Step 1<br>BBA linear gain value for index 1 (format: u5.3). Nominal value is 10^(3/20). Stored with 3 fractional bits, e.g. round([10^(3/20)]*2^3) = 11decimal |
| 7-0<br>BBA_RES_TUNE_LIN_VAL_0 | BBA Resistor Tune Linear Gain Step 0<br>BBA linear gain value for index 0 (format: u5.3). Nominal value is 10^(0/20). Stored with 3 fractional bits, e.g. round([10^(0/20)]*2^3) = 8decimal |

## A.2.3.31 BBA Resistor Tune Values 7..4 (BBA_RES_TUNE_LIN_VAL_7_4)

### A.2.3.31.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_7_4 | 78h |

### A.2.3.31.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_7 | | | | | | | | BBA_RES_TUNE_LIN_VAL_6 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BBA_RES_TUNE_LIN_VAL_5 | | | | | | | | BBA_RES_TUNE_LIN_VAL_4 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.31.3 Fields

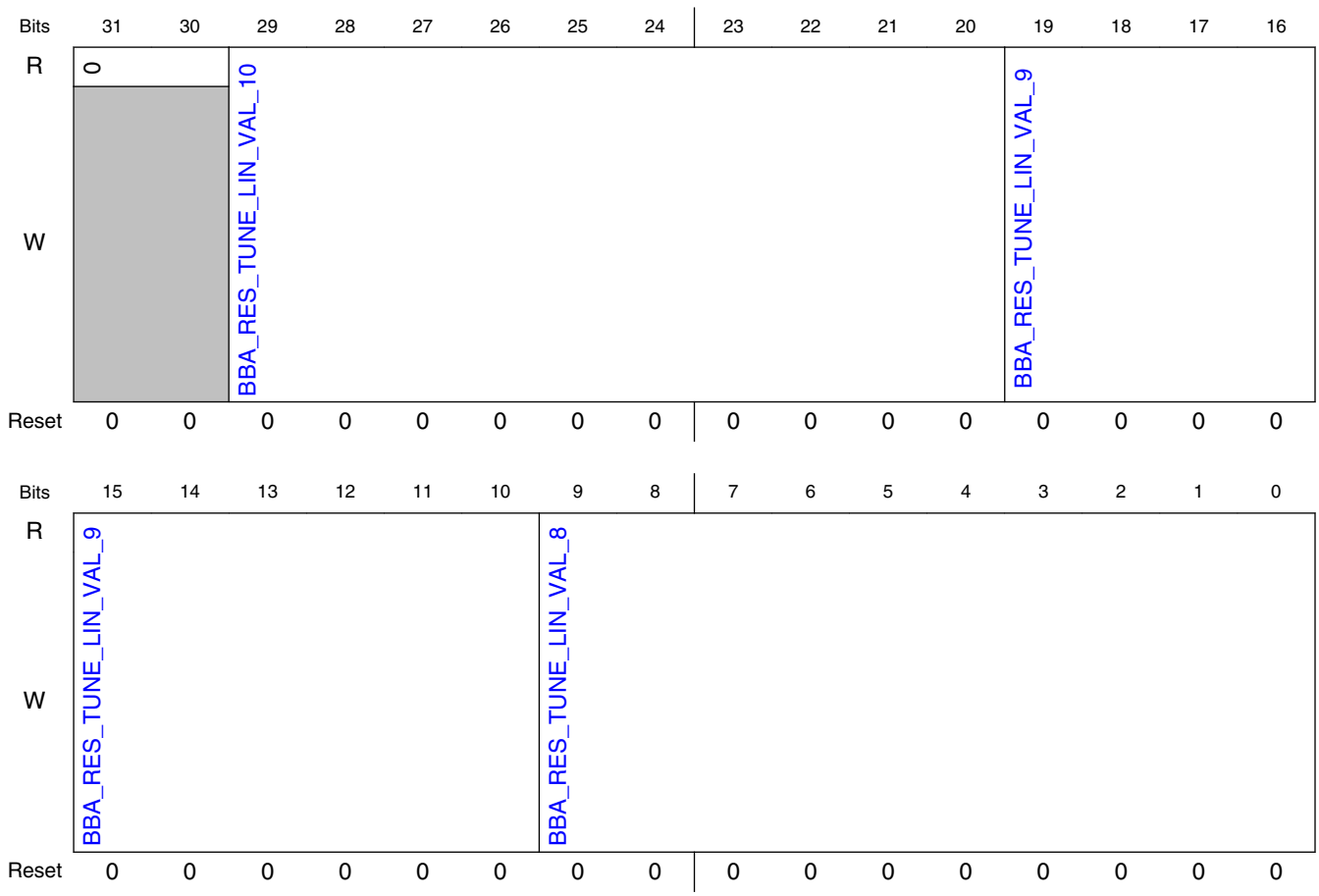
| Field | Function |
|---|---|
| 31-24<br>BBA_RES_TUNE_LIN_VAL_7 | BBA Resistor Tune Linear Gain Step 7<br>BBA linear gain value for index 7 (format: u6.2). Nominal value is 10^(21/20). Stored with 2 fractional bits, e.g. round([10^(21/20)]*2^2) = 45 decimal |
| 23-16<br>BBA_RES_TUNE_LIN_VAL_6 | BBA Resistor Tune Linear Gain Step 6<br>BBA linear gain value for index 6 (format: u5.3). Nominal value is 10^(18/20). Stored with 3 fractional bits, e.g. round([10^(18/20)]*2^3) = 64decimal |
| 15-8<br>BBA_RES_TUNE_LIN_VAL_5 | BBA Resistor Tune Linear Gain Step 5<br>BBA linear gain value for index 5 (format: u5.3). Nominal value is 10^(15/20). Stored with 3 fractional bits, e.g. round([10^(15/20)]*2^3) = 45decimal |
| 7-0<br>BBA_RES_TUNE_LIN_VAL_4 | BBA Resistor Tune Linear Gain Step 4<br>BBA linear gain value for index 4 (format: u5.3). Nominal value is 10^(12/20). Stored with 3 fractional bits, e.g. round([10^(12/20)]*2^3) = 32decimal |

## A.2.3.32 BBA Resistor Tune Values 10..8 (BBA_RES_TUNE_LIN_VAL_10_8)

## A.2.3.32.1 Offset

| Register | Offset |
|---|---|
| BBA_RES_TUNE_LIN_VAL_10_8 | 7Ch |

## A.2.3.32.2 Diagram




## A.2.3.32.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-20<br>BBA_RES_TUNE_LIN_VAL_10 | BBA Resistor Tune Linear Gain Step 10<br>BBA linear gain value for index 10 (format: u7.3). Nominal value is 10^(30/20). Stored with 3 fractional bits, e.g. round([10^(30/20)]*2^3) = 253decimal |
| 19-10 | BBA Resistor Tune Linear Gain Step 9 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BBA_RES_TUNE_LIN_VAL_9 | BBA linear gain value for index 9 (format: u7.3). Nominal value is 10^(27/20). Stored with 3 fractional bits, e.g. round([10^(27/20)]*2^3) = 179decimal |
| 9-0<br>BBA_RES_TUNE_LIN_VAL_8 | BBA Resistor Tune Linear Gain Step 8<br>BBA linear gain value for index 8 (format: u7.3). Nominal value is 10^(24/20). Stored with 3 fractional bits, e.g. round([10^(24/20)]*2^3) = 127decimal |

## A.2.3.33 AGC Gain Tables Step 03..00 (AGC_GAIN_TBL_03_00)

### A.2.3.33.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_03_00 | 80h |

### A.2.3.33.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_03 | | | | BBA_GAIN_03 | | | | LNA_GAIN_02 | | | | BBA_GAIN_02 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_01 | | | | BBA_GAIN_01 | | | | LNA_GAIN_00 | | | | BBA_GAIN_00 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.33.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_03 | LNA Gain 03<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_03 | BBA Gain 03<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_02 | LNA Gain 02<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_02 | BBA Gain 02<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-12<br>LNA_GAIN_01 | LNA Gain 01<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_01 | BBA Gain 01<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_00 | LNA Gain 00<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_00 | BBA Gain 00<br>BBA GAIN. |

## A.2.3.34 AGC Gain Tables Step 07..04 (AGC_GAIN_TBL_07_04)

### A.2.3.34.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_07_04 | 84h |

### A.2.3.34.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_07 | | | | BBA_GAIN_07 | | | | LNA_GAIN_06 | | | | BBA_GAIN_06 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_05 | | | | BBA_GAIN_05 | | | | LNA_GAIN_04 | | | | BBA_GAIN_04 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.34.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_07 | LNA Gain 07<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_07 | BBA Gain 07<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-20<br>LNA_GAIN_06 | LNA Gain 06<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_06 | BBA Gain 06<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_05 | LNA Gain 05<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_05 | BBA Gain 05<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_04 | LNA Gain 04<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_04 | BBA Gain 04<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.35 AGC Gain Tables Step 11..08 (AGC_GAIN_TBL_11_08)

### A.2.3.35.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_11_08 | 88h |

### A.2.3.35.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_11 | | | | BBA_GAIN_11 | | | | LNA_GAIN_10 | | | | BBA_GAIN_10 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_09 | | | | BBA_GAIN_09 | | | | LNA_GAIN_08 | | | | BBA_GAIN_08 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.35.3 Fields

| Field | Function |
|---|---|
| 31-28 | LNA Gain 11 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LNA_GAIN_11 | LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_11 | BBA Gain 11<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_10 | LNA Gain 10<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_10 | BBA Gain 10<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_09 | LNA Gain 09<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_09 | BBA Gain 09<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_08 | LNA Gain 08<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_08 | BBA Gain 08<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.36 AGC Gain Tables Step 15..12 (AGC_GAIN_TBL_15_12)

### A.2.3.36.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_15_12 | 8Ch |

### A.2.3.36.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_15 | | | | BBA_GAIN_15 | | | | LNA_GAIN_14 | | | | BBA_GAIN_14 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_13 | | | | BBA_GAIN_13 | | | | LNA_GAIN_12 | | | | BBA_GAIN_12 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.36.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_15 | LNA Gain 15<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_15 | BBA Gain 15<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_14 | LNA Gain 14<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_14 | BBA Gain 14<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_13 | LNA Gain 13<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_13 | BBA Gain 13<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_12 | LNA Gain 12<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_12 | BBA Gain 12<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.37 AGC Gain Tables Step 19..16 (AGC_GAIN_TBL_19_16)

### A.2.3.37.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_19_16 | 90h |

### A.2.3.37.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_19 | | | | BBA_GAIN_19 | | | | LNA_GAIN_18 | | | | BBA_GAIN_18 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_17 | | | | BBA_GAIN_17 | | | | LNA_GAIN_16 | | | | BBA_GAIN_16 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.37.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_19 | LNA Gain 19<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_19 | BBA Gain 193<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_18 | LNA Gain 18<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_18 | BBA Gain 18<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_17 | LNA Gain 17<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_17 | BBA Gain 17<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_16 | LNA Gain 16<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_16 | BBA Gain 16<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.38 AGC Gain Tables Step 23..20 (AGC_GAIN_TBL_23_20)

### A.2.3.38.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_23_20 | 94h |

### A.2.3.38.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_23 | | | | BBA_GAIN_23 | | | | LNA_GAIN_22 | | | | BBA_GAIN_22 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LNA_GAIN_21 | | | | BBA_GAIN_21 | | | | LNA_GAIN_20 | | | | BBA_GAIN_20 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.38.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LNA_GAIN_23 | LNA Gain 23<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 27-24<br>BBA_GAIN_23 | BBA Gain 23<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 23-20<br>LNA_GAIN_22 | LNA Gain 22<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_22 | BBA Gain 22<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_21 | LNA Gain 21<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_21 | BBA Gain 21<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_20 | LNA Gain 20<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_20 | BBA Gain 20<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.39 AGC Gain Tables Step 26..24 (AGC_GAIN_TBL_26_24)

### A.2.3.39.1 Offset

| Register | Offset |
|---|---|
| AGC_GAIN_TBL_26_24 | 98h |

### A.2.3.39.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | LNA_GAIN_26 | | | | BBA_GAIN_26 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LNA_GAIN_25 | | | | BBA_GAIN_25 | | | | LNA_GAIN_24 | | | | BBA_GAIN_24 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.39.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-20<br>LNA_GAIN_26 | LNA Gain 26<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 19-16<br>BBA_GAIN_26 | BBA Gain 26<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 15-12<br>LNA_GAIN_25 | LNA Gain 25<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 11-8<br>BBA_GAIN_25 | BBA Gain 25<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |
| 7-4<br>LNA_GAIN_24 | LNA Gain 24<br>LNA GAIN. See Table 64-23 for the mapping of LNA gain indexes to gain values. |
| 3-0<br>BBA_GAIN_24 | BBA Gain 24<br>BBA GAIN 0=0 dB, 1=3 dB, 2=6 dB … A=30 dB. |

## A.2.3.40 DCOC Offset (DCOC_OFFSET_0 - DCOC_OFFSET_26)

### A.2.3.40.1 Offset

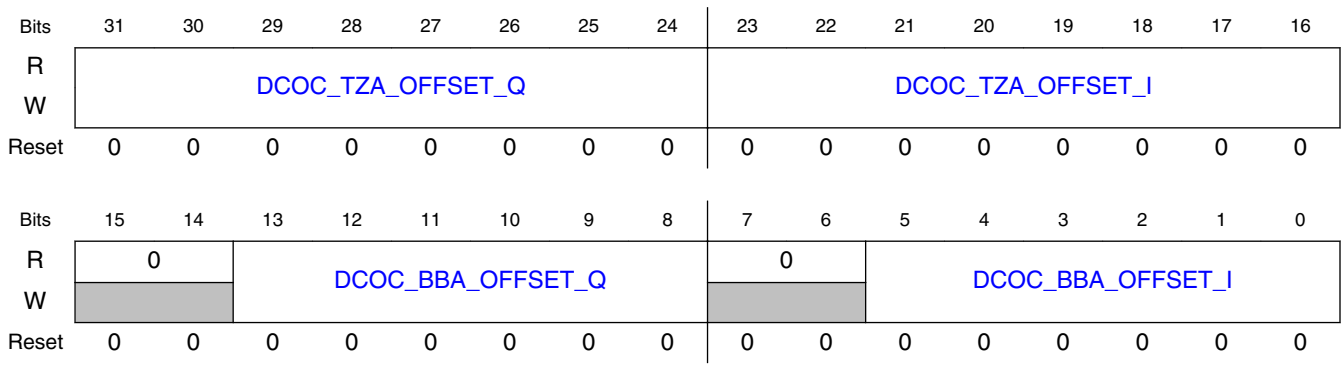
For a = 0 to 26:

| Register | Offset |
|---|---|
| DCOC_OFFSET_a | A0h + (a × 4h) |

### A.2.3.40.2 Function

DCOC Offset Registers. These registers can only be accessed when the radio oscillator clock is active.

### A.2.3.40.3 Diagram




### A.2.3.40.4 Fields

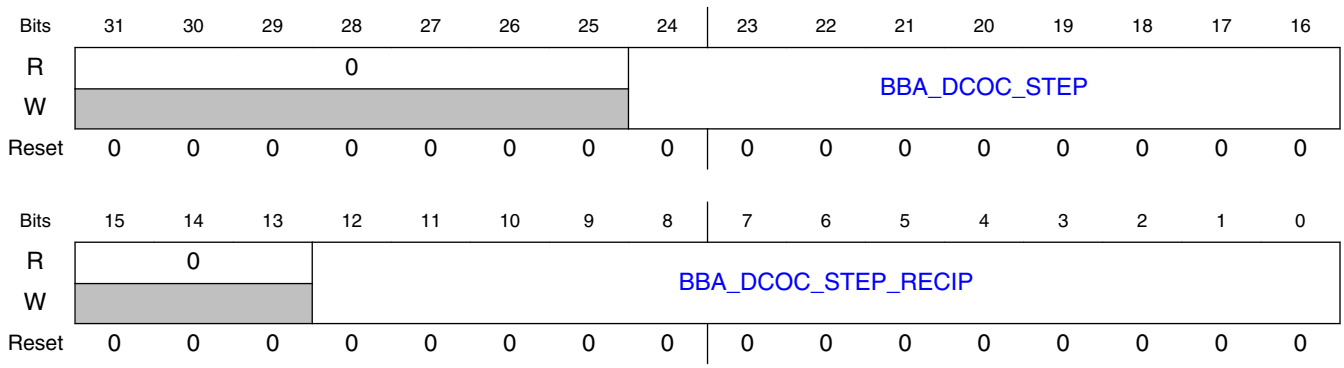
| Field | Function |
|---|---|
| 31-24<br>DCOC_TZA_OFFSET_Q | DCOC TZA Q-channel offset<br>DCOC TZA Q-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 23-16<br>DCOC_TZA_OFFSET_I | DCOC TZA I-channel offset<br>DCOC TZA I-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 15-14<br>— | Reserved. |
| 13-8<br>DCOC_BBA_OFFSET_Q | DCOC BBA Q-channel offset<br>DCOC BBA Q-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |
| 7-6<br>— | Reserved. |
| 5-0<br>DCOC_BBA_OFFSET_I | DCOC BBA I-channel offset<br>DCOC BBA I-channel offset. When RX_DCOC_CAL_EN=1, this table is generated by the DCOC during calibration. When RX_DCOC_CAL_EN=0, this table may optionally be written by software. |

# A.2.3.41 DCOC BBA DAC Step (DCOC_BBA_STEP)

## A.2.3.41.1 Offset

| Register | Offset |
|---|---|
| DCOC_BBA_STEP | 10Ch |

## A.2.3.41.2 Diagram




## A.2.3.41.3 Fields

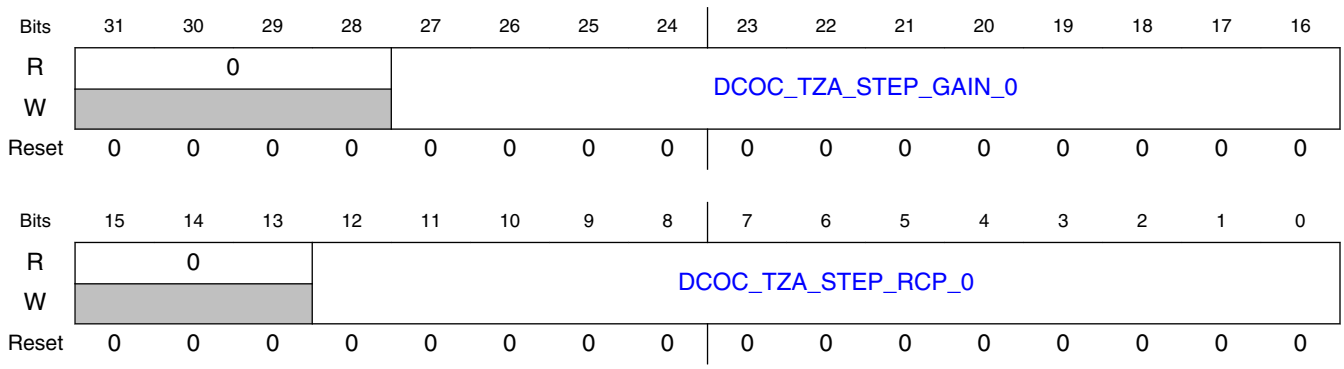
| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-16<br>BBA_DCOC_STEP | DCOC BBA Step Size<br>DCOC BBA Step Size (format: u6.3). This is the BBA DAC resolution in mV (1.2e3/32*63.44/120 = 19.83mV) times the ADC/decimator gain. This value is stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>BBA_DCOC_STEP_RECIP | DCOC BBA Reciprocal of Step Size<br>DCOC BBA Reciprocal of Step Size (format: u.[00]13). This the reciprocal of the BBA DCOC STEP value. This value is stored as a 15 bit fraction (though only 13 bits are programmed). |

# A.2.3.42 DCOC TZA DAC Step 0 (DCOC_TZA_STEP_0)

## A.2.3.42.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_0 | 110h |

## A.2.3.42.2 Diagram




## A.2.3.42.3 Fields

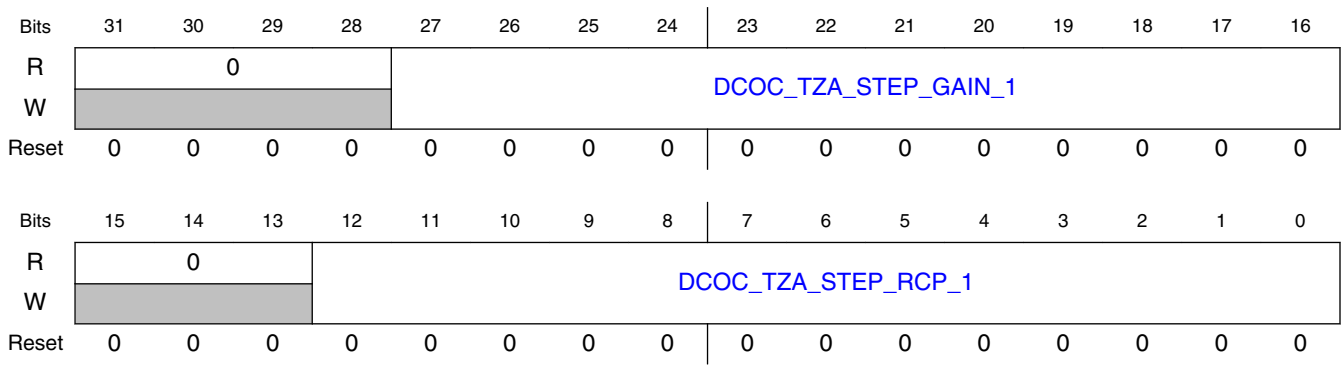
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_0 | DCOC_TZA_STEP_GAIN_0<br>DCOC TZA Step Size 0 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 0. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_0 | DCOC_TZA_STEP_RCP_0<br>DCOC TZA Reciprocal of Step Size 0, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_0. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.43 DCOC TZA DAC Step 1 (DCOC_TZA_STEP_1)

## A.2.3.43.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_1 | 114h |

## A.2.3.43.2 Diagram




## A.2.3.43.3 Fields

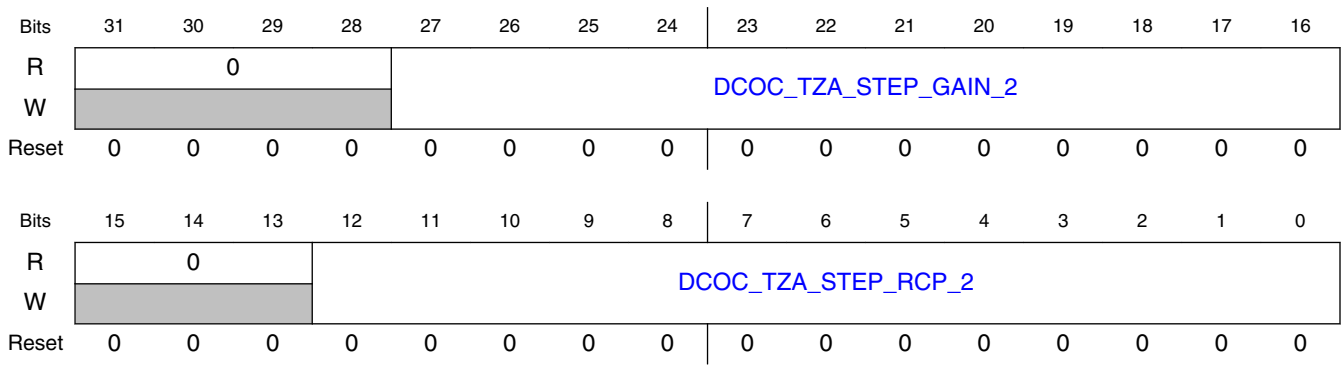
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_1 | DCOC_TZA_STEP_GAIN_1<br>DCOC TZA Step Size 1 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 1. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_1 | DCOC_TZA_STEP_RCP_1<br>DCOC TZA Reciprocal of Step Size 1, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_1. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.44 DCOC TZA DAC Step 2 (DCOC_TZA_STEP_2)

## A.2.3.44.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_2 | 118h |

## A.2.3.44.2 Diagram




## A.2.3.44.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_2 | DCOC_TZA_STEP_GAIN_2<br>DCOC TZA Step Size 2 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 2. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_2 | DCOC_TZA_STEP_RCP_2<br>DCOC TZA Reciprocal of Step Size 2, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_2. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.45 DCOC TZA DAC Step 3 (DCOC_TZA_STEP_3)

## A.2.3.45.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_3 | 11Ch |

## A.2.3.45.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_TZA_STEP_GAIN_3 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_3 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.45.3 Fields

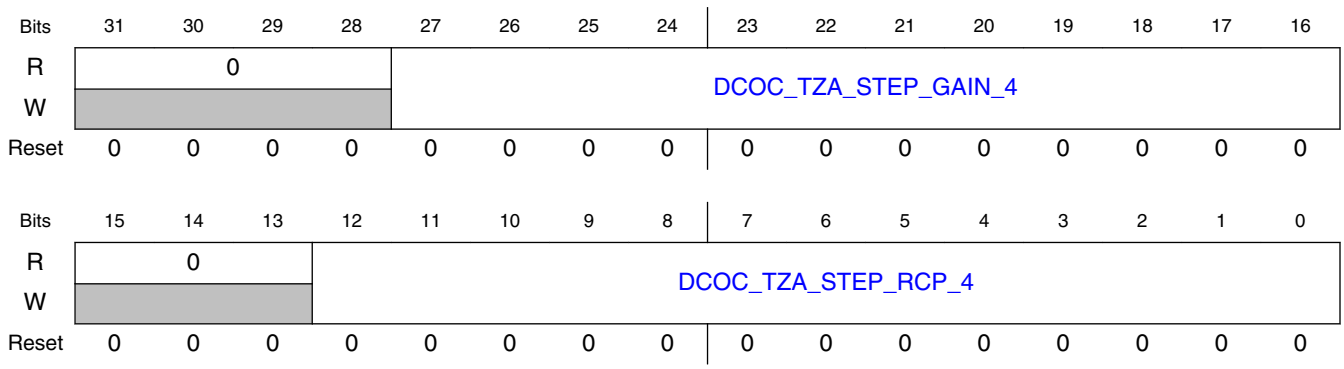
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_3 | DCOC_TZA_STEP_GAIN_3<br>DCOC TZA Step Size 3 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 3. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_3 | DCOC_TZA_STEP_RCP_3<br>DCOC TZA Reciprocal of Step Size 3, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_3. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.46 DCOC TZA DAC Step 4 (DCOC_TZA_STEP_4)

## A.2.3.46.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_4 | 120h |

## A.2.3.46.2 Diagram




## A.2.3.46.3 Fields

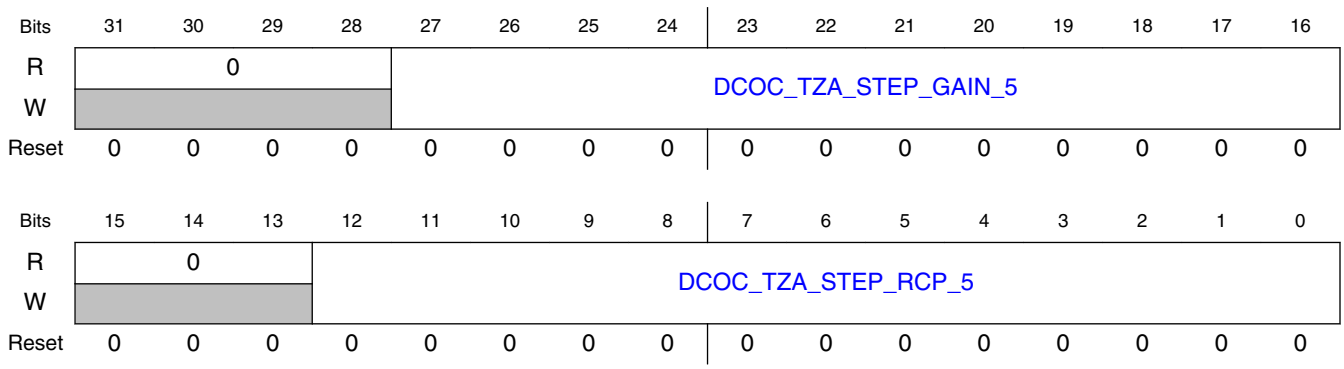
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_4 | DCOC_TZA_STEP_GAIN_4<br>DCOC TZA Step Size 4 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 4. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_4 | DCOC_TZA_STEP_RCP_4<br>DCOC TZA Reciprocal of Step Size 4, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_4. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.47 DCOC TZA DAC Step 5 (DCOC_TZA_STEP_5)

## A.2.3.47.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_5 | 124h |

## A.2.3.47.2 Diagram




## A.2.3.47.3 Fields

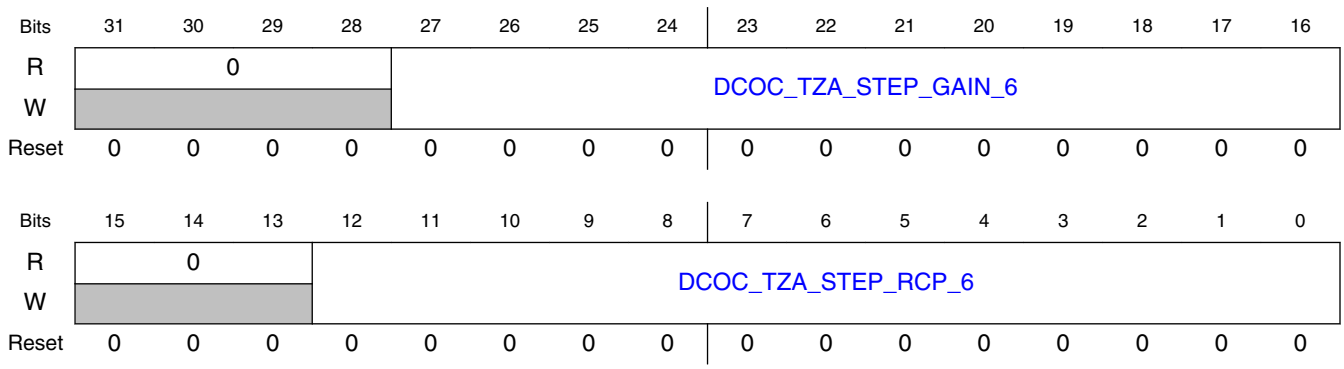
| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_5 | DCOC_TZA_STEP_GAIN_5<br>DCOC TZA Step Size 5 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 5. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_5 | DCOC_TZA_STEP_RCP_5<br>DCOC TZA Reciprocal of Step Size 5, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_5. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.48 DCOC TZA DAC Step 6 (DCOC_TZA_STEP_6)

## A.2.3.48.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_6 | 128h |

## A.2.3.48.2 Diagram




## A.2.3.48.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_TZA_STEP_GAIN_6 | DCOC_TZA_STEP_GAIN_6<br>DCOC TZA Step Size 6 with gain (format u9.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 6. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_6 | DCOC_TZA_STEP_RCP_6<br>DCOC TZA Reciprocal of Step Size 6, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_6. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.49 DCOC TZA DAC Step 7 (DCOC_TZA_STEP_7)

## A.2.3.49.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_7 | 12Ch |

## A.2.3.49.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_GAIN_7 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DCOC_TZA_STEP_RCP_7 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.49.3 Fields

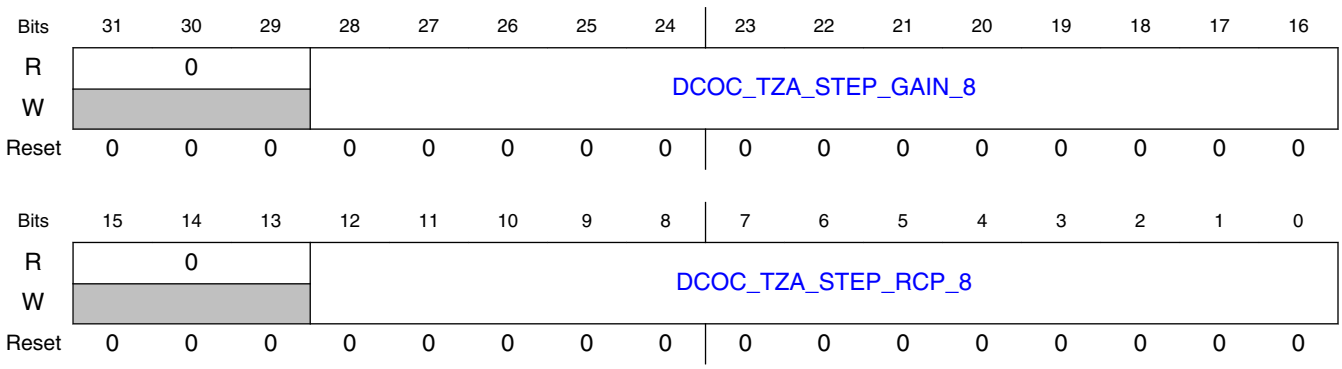
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DCOC_TZA_STEP_GAIN_7 | DCOC_TZA_STEP_GAIN_7<br>DCOC TZA Step Size 7 with gain (format u10.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 7. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_7 | DCOC_TZA_STEP_RCP_7<br>DCOC TZA Reciprocal of Step Size 7, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_7. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.50 DCOC TZA DAC Step 5 (DCOC_TZA_STEP_8)

## A.2.3.50.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_8 | 130h |

## A.2.3.50.2 Diagram




## A.2.3.50.3 Fields

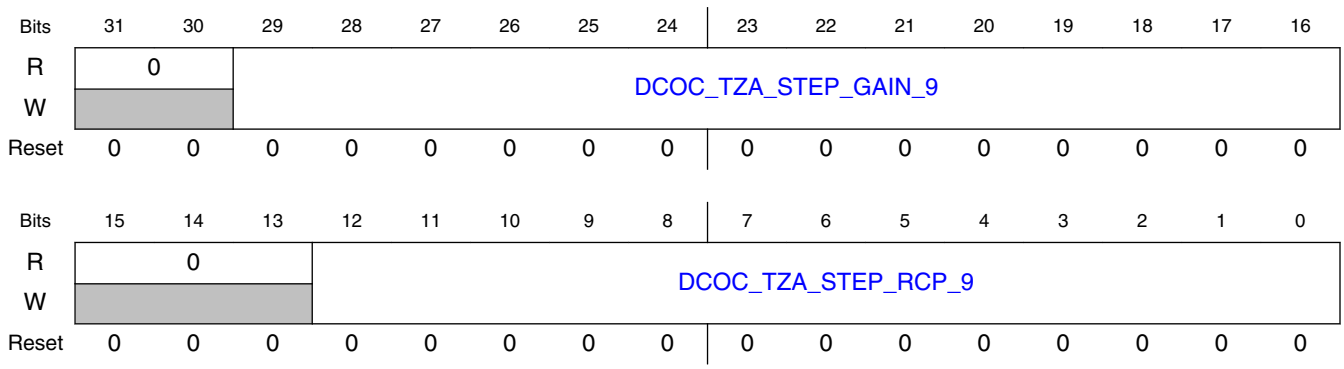
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DCOC_TZA_STEP_GAIN_8 | DCOC_TZA_STEP_GAIN_8<br>DCOC TZA Step Size 8 with gain (format u10.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 8. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_8 | DCOC_TZA_STEP_RCP_8<br>DCOC TZA Reciprocal of Step Size 8, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_8. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.51 DCOC TZA DAC Step 9 (DCOC_TZA_STEP_9)

### A.2.3.51.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_9 | 134h |

### A.2.3.51.2 Diagram




### A.2.3.51.3 Fields

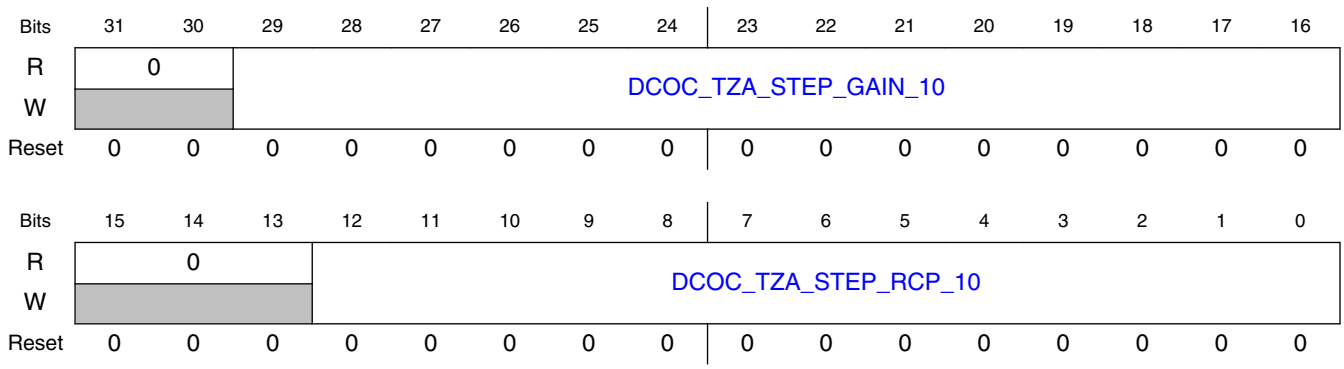
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-16<br>DCOC_TZA_STEP_GAIN_9 | DCOC_TZA_STEP_GAIN_9<br>DCOC TZA Step Size 9 with gain (format u11.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 9. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_9 | DCOC_TZA_STEP_RCP_9<br>DCOC TZA Reciprocal of Step Size 9, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_9. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

## A.2.3.52 DCOC TZA DAC Step 10 (DCOC_TZA_STEP_10)

## A.2.3.52.1 Offset

| Register | Offset |
|---|---|
| DCOC_TZA_STEP_10 | 138h |

## A.2.3.52.2 Diagram




## A.2.3.52.3 Fields

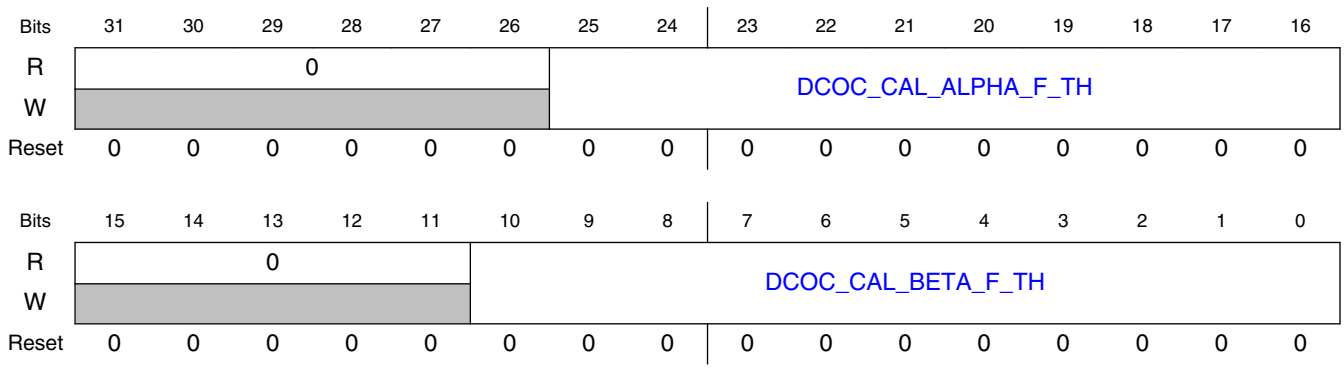
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-16<br>DCOC_TZA_STEP_GAIN_10 | DCOC_TZA_STEP_GAIN_10<br>DCOC TZA Step Size 10 with gain (format u11.3). This is the TZA DAC resolution in mV (1.2e3/128*22.9/34 = 6.31mV) times the ADC/decimator gain times BBA gain for BBA index 10. This value iis stored in the register with 3 fractional bits. |
| 15-13<br>— | Reserved. |
| 12-0<br>DCOC_TZA_STEP_RCP_10 | DCOC_TZA_STEP_RCP_10<br>DCOC TZA Reciprocal of Step Size 10, (format: u.[00]13). This the reciprocal of the DCOC_TZA_STEP_GAIN_10. The value is stored as a 15bit fractional value (though only 13 bits are programmed). |

# A.2.3.53 DCOC Calibration Fail Thresholds (DCOC_CAL_FAIL_TH)

## A.2.3.53.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_FAIL_TH | 160h |

## A.2.3.53.2 Diagram




## A.2.3.53.3 Fields

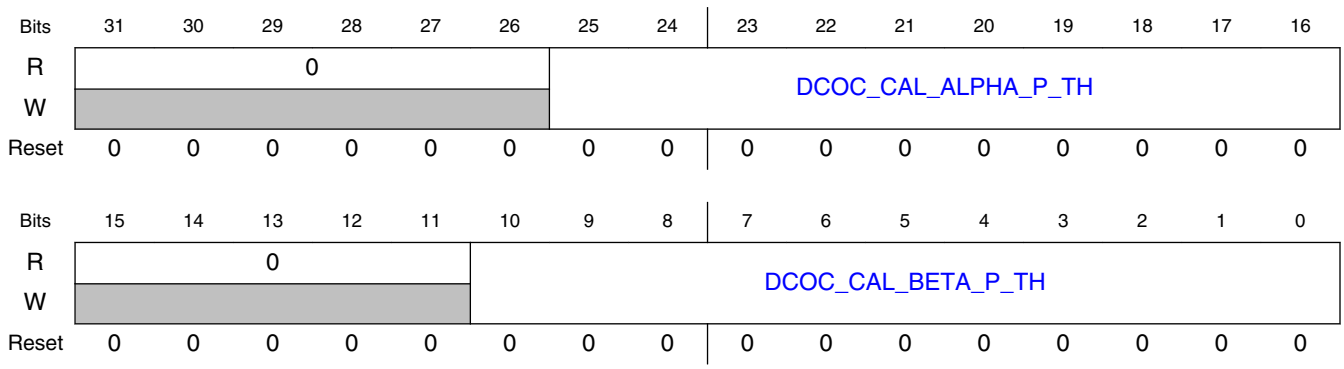
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>DCOC_CAL_ALPHA_F_TH | DCOC Calibration Alpha Fail Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the alpha-hat I and Q estimates are compared gainst this fail threshold. If any of the estimates are greater than the fail threshold, the calibration is considered a failure. Format is u4.6. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_BETA_F_TH | DCOC Calibration Beta Fail Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the beta-hat I and Q estimates are compared against this fail threshold. If any of the estimates are greater than the fail threshold, the calibration is considered a failure. Format is u11. |

# A.2.3.54 DCOC Calibration Pass Thresholds (DCOC_CAL_PASS_TH)

### A.2.3.54.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_PASS_TH | 164h |

### A.2.3.54.2 Diagram




### A.2.3.54.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>DCOC_CAL_ALPHA_P_TH | DCOC Calibration Alpha Pass Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the alpha-hat I and Q estimates are compared against this pass threshold. If all of the estimates are less than the pass threshold, the calibration is considered as passing. Format is u4.6. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_BETA_P_TH | DCOC Calibration Beta Pass Threshold<br>If DCOC_CTRL_0[DCOC_CHECK_EN] is set, the absolute value of the beta-hat I and Q estimates are compared against this pass threshold. If all of the estimates are less than the pass threshold, the calibration is considered as passing. Format is u11. |

## A.2.3.55 DCOC Calibration Alpha (DCOC_CAL_ALPHA)

## A.2.3.55.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_ALPHA | 168h |

## A.2.3.55.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_CAL_ALPHA_Q | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DCOC_CAL_ALPHA_I | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.55.3 Fields

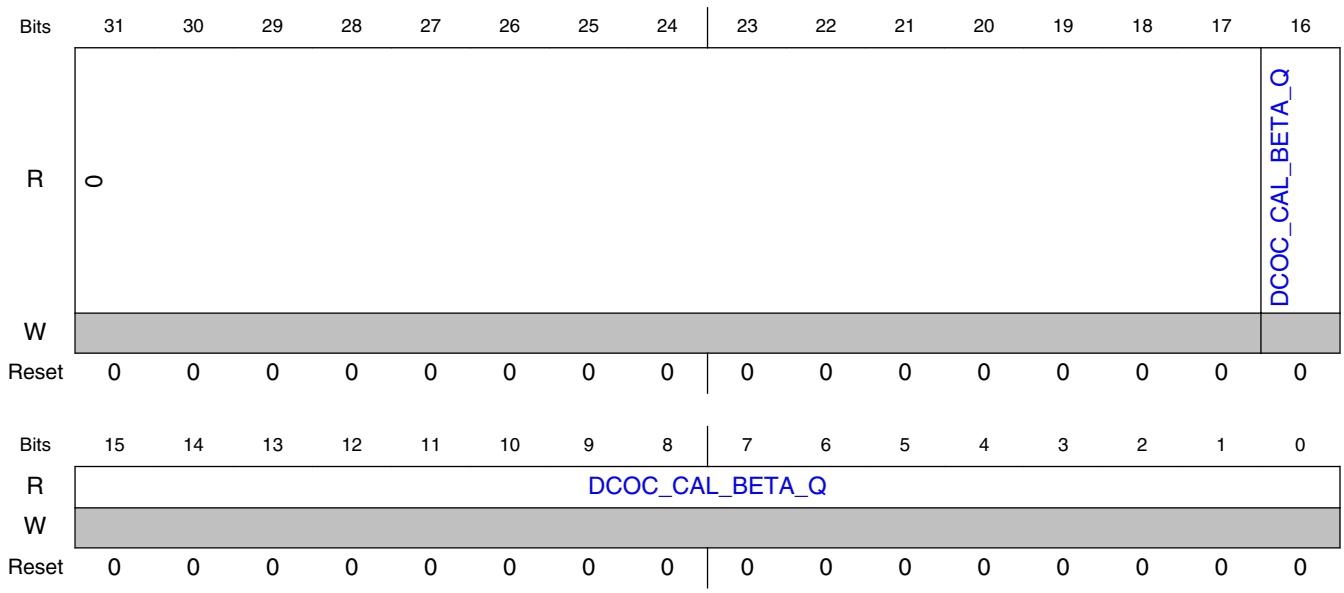
| Field | Function |
|---|---|
| 31-27<br>— | Reserved. |
| 26-16<br>DCOC_CAL_ALPHA_Q | DCOC_CAL_ALPHA_Q<br>DCOC Calibration Q-channel ALPHA. This read-only value represents the Q channel estimate of the ALPHA DC component calculated in DCOC calibration. Format s4.6. This is provided for debug/ characterization purposes. |
| 15-11<br>— | Reserved. |
| 10-0<br>DCOC_CAL_ALPHA_I | DCOC Calibration I-channel ALPHA constant<br>DCOC Calibration I-channel ALPHA. This read-only value represents the I channel estimate of the ALPHA DC component calculated in DCOC calibration. Format s4.6. This is provided for debug/ characterization purposes. |

# A.2.3.56 DCOC Calibration Beta Q (DCOC_CAL_BETA_Q)

### A.2.3.56.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_BETA_Q | 16Ch |

### A.2.3.56.2 Diagram




### A.2.3.56.3 Fields

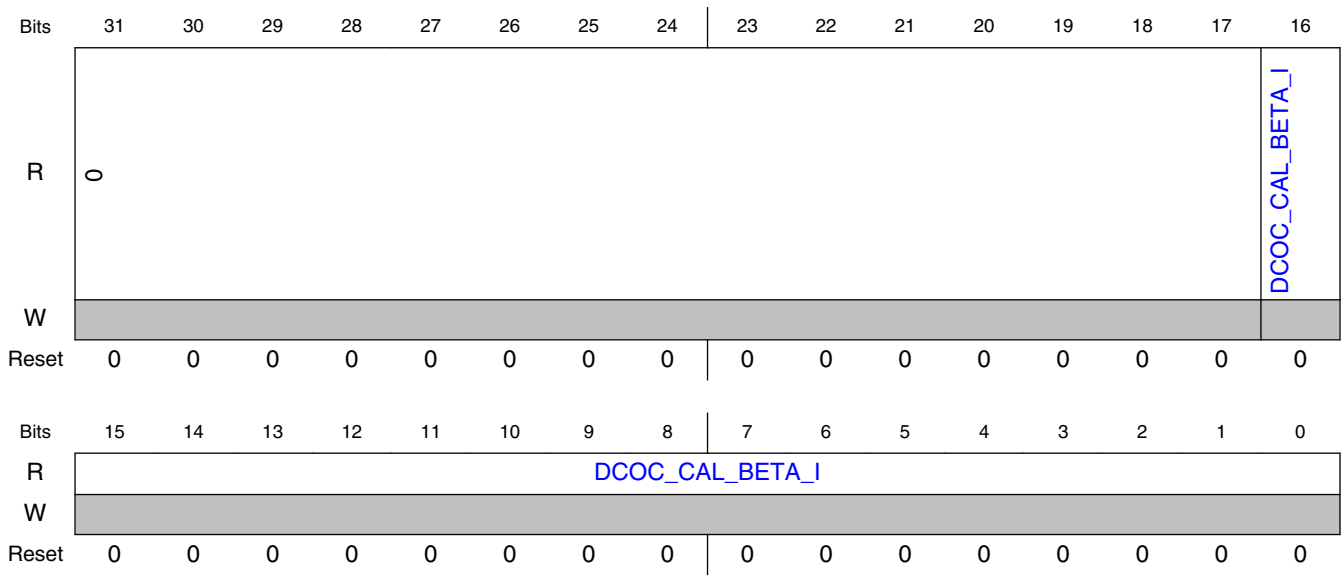
| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16-0<br>DCOC_CAL_BETA_Q | DCOC_CAL_BETA_Q<br>DCOC Calibration Q-channel BETA. This read-only value represents the Q channel estimate of the BETA DC component calculated in DCOC calibration. Format s11.5. This is provided for debug/characterization purposes. |

## A.2.3.57 DCOC Calibration Beta I (DCOC_CAL_BETA_I)

### A.2.3.57.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_BETA_I | 170h |

### A.2.3.57.2 Diagram




### A.2.3.57.3 Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16-0<br>DCOC_CAL_BETA_I | DCOC_CAL_BETA_I<br>DCOC Calibration I-channel BETA. This read-only value represents the I channel estimate of the BETA DC component calculated in DCOC calibration. Format s11.5. This is provided for debug/characterization purposes. |

## A.2.3.58 DCOC Calibration Gamma (DCOC_CAL_GAMMA)

### A.2.3.58.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_GAMMA | 174h |

### A.2.3.58.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCOC_CAL_GAMMA_Q | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCOC_CAL_GAMMA_I | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.58.3 Fields

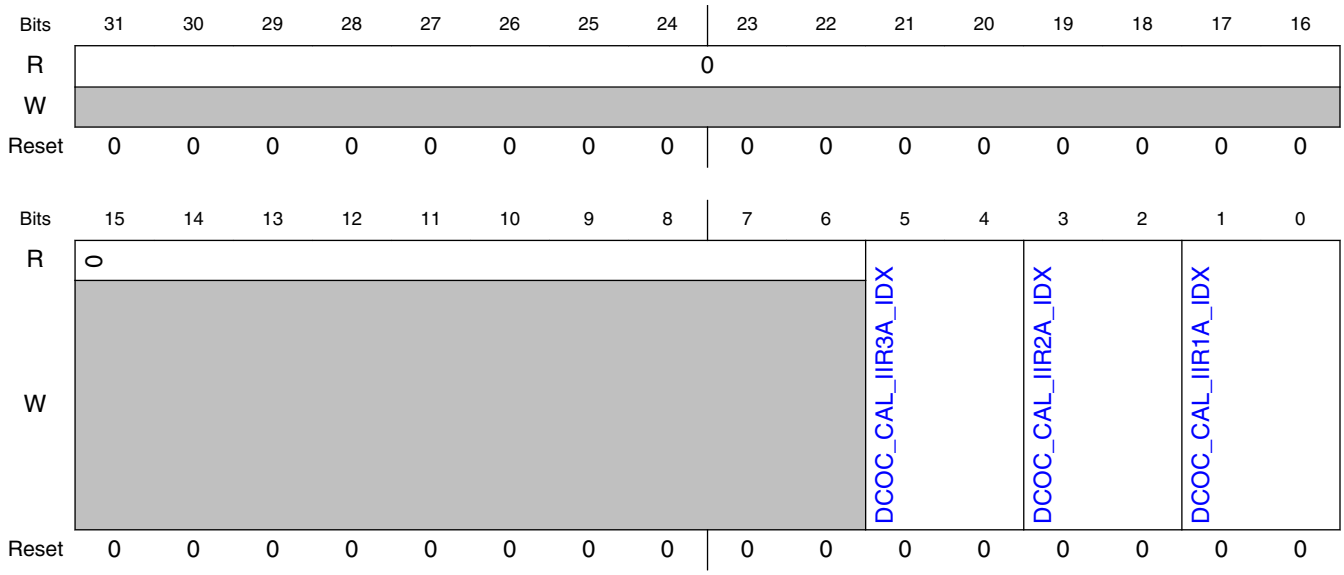
| Field | Function |
|---|---|
| 31-16<br>DCOC_CAL_GAMMA_Q | DCOC_CAL_GAMMA_Q<br>DCOC Calibration Q-channel GAMMA. This read-only value represents the Q channel estimate of the GAMMA DC component calculated in DCOC calibration. Format s11.4. This is provided for debug/ characterization purposes. |
| 15-0<br>DCOC_CAL_GAMMA_I | DCOC_CAL_GAMMA_I<br>DCOC Calibration I-channel GAMMA. This read-only value represents the I channel estimate of the GAMMA DC component calculated in DCOC calibration. Format s11.4. This is provided for debug/ characterization purposes. |

## A.2.3.59 DCOC Calibration IIR (DCOC_CAL_IIR)

### A.2.3.59.1 Offset

| Register | Offset |
|---|---|
| DCOC_CAL_IIR | 178h |

## A.2.3.59.2 Diagram




## A.2.3.59.3 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-4<br>DCOC_CAL_IIR 3A_IDX | DCOC Calibration IIR 3A Index<br>DCOC Calibration IIR 3A Index. Defines the filter coefficient use for the 3rd IIR filter in the DCOC calibration DC estimator.<br>00b - 1/4<br>01b - 1/8<br>10b - 1/16<br>11b - 1/32 |
| 3-2<br>DCOC_CAL_IIR 2A_IDX | DCOC Calibration IIR 2A Index<br>DCOC Calibration IIR 2A Index. Defines the filter coefficient use for the 2nd IIR filter in the DCOC calibration DC estimator.<br>00b - 1/1<br>01b - 1/4<br>10b - 1/8<br>11b - 1/16 |
| 1-0<br>DCOC_CAL_IIR 1A_IDX | DCOC Calibration IIR 1A Index<br>DCOC Calibration IIR 1A Index. Defines the filter coefficient use for the 1st IIR filter in the DCOC calibration DC estimator.<br>00b - 1/1<br>01b - 1/4<br>10b - 1/8<br>11b - 1/16 |

# A.2.3.60 DCOC Calibration Result (DCOC_CAL1 - DCOC_CAL3)

## A.2.3.60.1 Offset

For a = 1 to 3:

| Register | Offset |
|---|---|
| DCOC_CALa | 17Ch + (a × 4h) |

## A.2.3.60.2 Function

Result of one of the calibration iterations.

## A.2.3.60.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_CAL_RES_Q | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DCOC_CAL_RES_I | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.60.4 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-16<br>DCOC_CAL_RES_Q | DCOC Calibration Result - Q Channel<br>Q channel DCOC calibration result. This value represents the DCOC's Q channel DC estimate for the nth calibration gain setting. Format s11. This is provided for debug/chacterization purposes. |
| 15-12<br>— | Reserved. |
| 11-0<br>DCOC_CAL_RES_I | DCOC Calibration Result - I Channel<br>I channel DCOC calibration result. This value represents the DCOC's I channel DC estimate for the nth calibration gain setting. Format s11. This is provided for debug/chacterization purposes. |

# A.2.3.61 RX_DIG CCA ED LQI Control Register 0 (CCA_ED_LQI_CTRL_0)

## A.2.3.61.1 Offset

| Register | Offset |
|---|---|
| CCA_ED_LQI_CTRL_0 | 190h |

## A.2.3.61.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | SNR_ADJ | | | | | | LQI_CNTR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CORR_CNTR_THRESH | | | | | | | | LQI_CORR_THRESH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.61.3 Fields

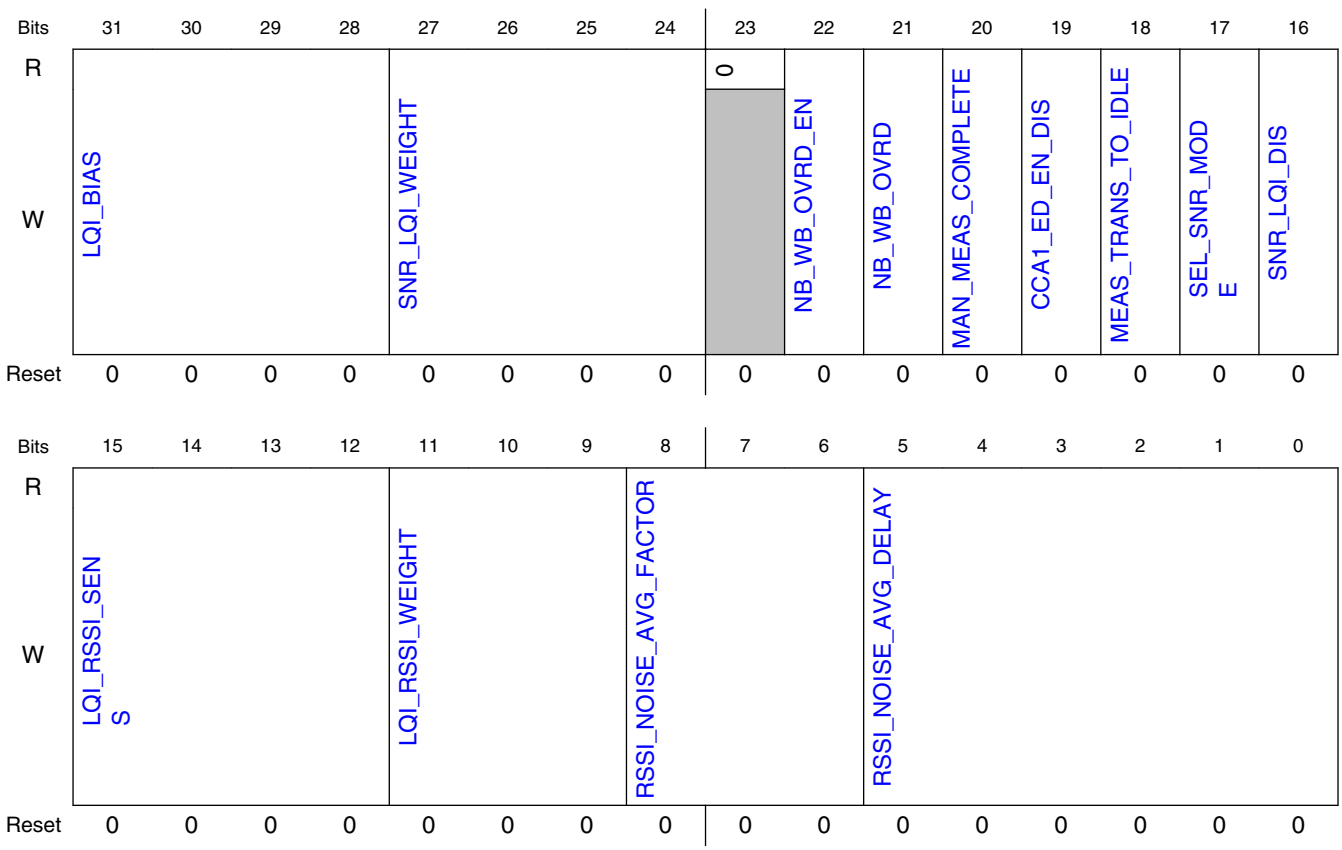
| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-24<br>SNR_ADJ | SNR calculation adjustment<br>Signed value with quarter dB resolution. |
| 23-16<br>LQI_CNTR | LQI Counter<br>Duration of LQI in uS. |
| 15-8<br>CORR_CNTR_THRESH | Correlation Count Threshold<br>Threshold used to compare the counted correlation magnitudes exceeding LQI_CORR_THRESH. |
| 7-0<br>LQI_CORR_THRESH | LQI Correlation Threshold<br>Threshold used to compare correlation magnitude values from the PHY |

# A.2.3.62 RX_DIG CCA ED LQI Control Register 1 (CCA_ED_LQI_CTRL_1)

## A.2.3.62.1 Offset

| Register | Offset |
|---|---|
| CCA_ED_LQI_CTRL_1 | 194h |

## A.2.3.62.2 Diagram




## A.2.3.62.3 Fields

| Field | Function |
|---|---|
| 31-28<br>LQI_BIAS | LQI Bias.<br>Bias used for LQI calculation. Applied bias = -36 + 2*LQI_BIAS. |
| 27-24 | SNR LQI Weight<br>SNR weight used for LQI calculation |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SNR_LQI_WEIGHT | 0000b - 0.0<br>0001b - 1.0<br>0010b - 1.125<br>0011b - 1.25<br>0100b - 1.375<br>0101b - 1.5<br>0110b - 1.625<br>0111b - 1.75<br>1000b - 1.875<br>1001b - 2.0<br>1010b - 2.125<br>1011b - 2.25<br>1100b - 2.375<br>1101b - 2.5<br>1110b - 2.625<br>1111b - 2.75 |
| 23<br>— | Reserved. |
| 22<br>NB_WB_OVRD_EN | Narrowband Wideband Override Enable<br>Override enable associated with NB_WB_OVRD. Intended to be used only for debug.<br>0b - Normal operation<br>1b - RSSI narrowband/wideband selection is via NB_WB_OVRD |
| 21<br>NB_WB_OVRD | Narrowband Wideband Override<br>Controls the RSSI's wideband/narrowband selection when NB_WB_OVRD_EN is set. Intended to be used only for debug.<br>0b - RSSI forced to be in Wideband mode if NB_WB_OVRD_EN is set<br>1b - RSSI forced to be in Narrowband mode if NB_WB_OVRD_EN is set |
| 20<br>MAN_MEAS_COMPLETE | Manual measurement complete<br>0b - Normal operation<br>1b - Manually asserts the measurement complete signal for the RX_DIG CCA/ED/LQI blocks. Intended to be used only for debug. |
| 19<br>CCA1_ED_EN_DIS | CCA1_ED_EN Disable<br>0b - Normal operation<br>1b - CCA1_ED_EN input is disabled |
| 18<br>MEAS_TRANS_TO_IDLE | Measurement Transition to IDLE<br>Establishes the state machine transition following an LQI or CCA1/ED measurement<br>0b - Module transitions to RSSI state<br>1b - Module transitions to IDLE state |
| 17<br>SEL_SNR_MODE | Select SNR Mode<br>Selects the SNR input used in the LQI calculator block.<br>0b - SNR estimate<br>1b - Mapped correlation magnitude |
| 16<br>SNR_LQI_DIS | SNR LQI Disable<br>0b - Normal operation.<br>1b - The RX_DIG CCA/ED/LQI block ignores the AA match input which starts an LQI measurement. |
| 15-12<br>LQI_RSSI_SENS | LQI RSSI Sensitivity<br>Unsigned integer used for calculation of LQI sensitivity. Sensitivity = -103 + LQI_RSSI_SENS. |
| 11-9 | LQI RSSI Weight |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LQI_RSSI_WEIGHT | RSSI weight used for LQI calculation<br>000b - 2.0<br>001b - 2.125<br>010b - 2.25<br>011b - 2.375<br>100b - 2.5<br>101b - 2.625<br>110b - 2.75<br>111b - 2.875 |
| 8-6<br>RSSI_NOISE_AVG_FACTOR | RSSI Noise Averaging Factor<br>Factor used for RSSI and SNR averaging.<br>000b - 1<br>001b - 64<br>010b - 70<br>011b - 128<br>100b - 139<br>101b - 256<br>110b - 277<br>111b - 512 |
| 5-0<br>RSSI_NOISE_AVG_DELAY | RSSI Noise Averaging Delay<br>Programmable delay used to delay the enable of averagers "RSSI Averager" and "Noise Averager" after a cca1_ed_trig assertion (in wideband mode) or after a aa_sfd_matched assertion (in narrowband mode). The delay is expressed in ticks of frequenc Fs/N, where Fs is the ADC decimator output sampling frequency in wideband mode or the base-band sampling frequency in the narrowband mode. |

## A.2.3.63 RX_DIG CCA ED LQI Status Register 0 (CCA_ED_LQI_STAT_0)

### A.2.3.63.1 Offset

| Register | Offset |
|---|---|
| CCA_ED_LQI_STAT_0 | 198h |

## A.2.3.63.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MEAS_COMPLETE | CCA1_STATE | SNR_OUT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ED_OUT | | | | | | | | LQI_OUT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.63.3 Fields

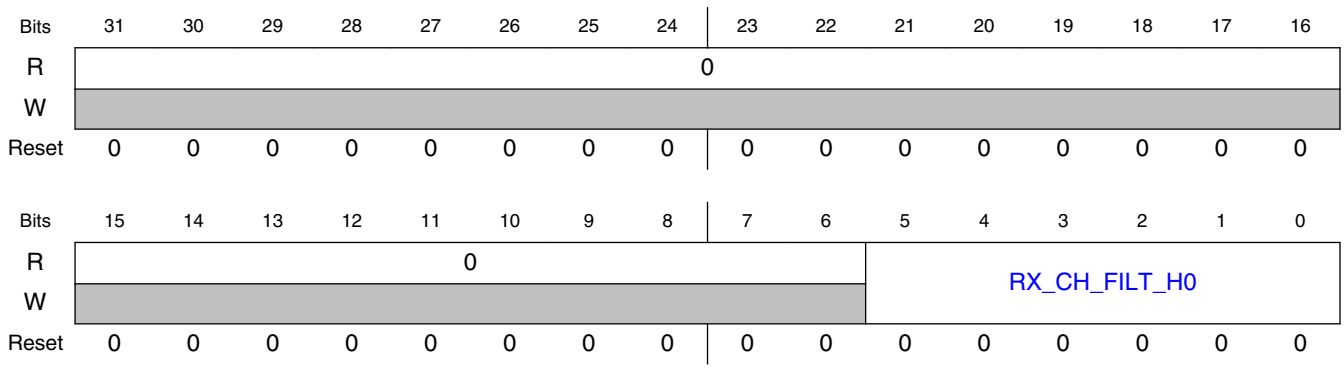
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>MEAS_COMPLETE | Measurement Complete<br>Set upon measurement complete |
| 24<br>CCA1_STATE | CCA1 State<br>Reflects the state of CCA1 channel state |
| 23-16<br>SNR_OUT | SNR output<br>Reflects the SNR measurement value. |
| 15-8<br>ED_OUT | ED output<br>Reflects the Energy Detect (DC) measurement value. |
| 7-0<br>LQI_OUT | LQI output<br>Reflects the LQI measurement value. |

# A.2.3.64 Receive Channel Filter Coefficient 0 (RX_CHF_COEF_0)

### A.2.3.64.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_0 | 1A0h |

### A.2.3.64.2 Diagram




### A.2.3.64.3 Fields

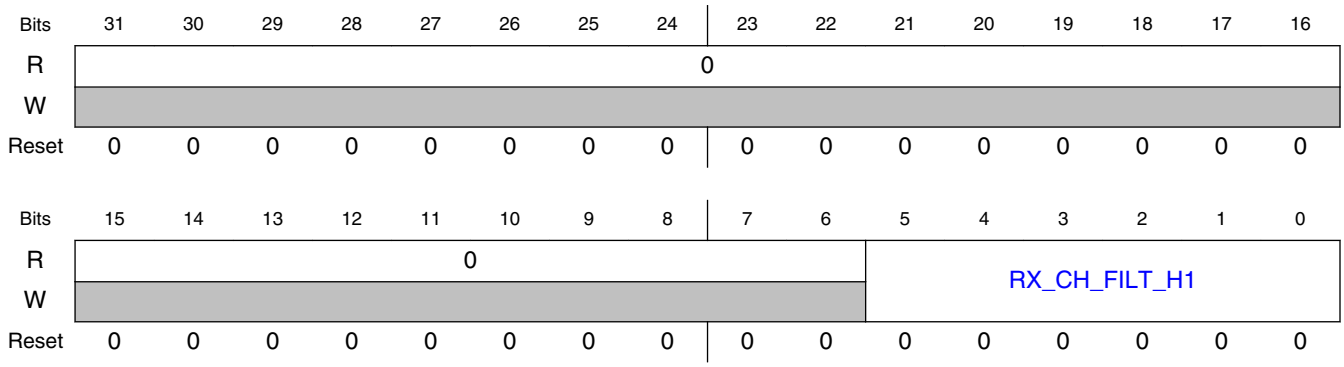
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>RX_CH_FILT_H0 | RX Channel Filter Coefficient 0<br>RX Channel Filter Coefficient 0, 6-bit signed fractional |

## A.2.3.65 Receive Channel Filter Coefficient 1 (RX_CHF_COEF_1)

### A.2.3.65.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_1 | 1A4h |

### A.2.3.65.2 Diagram




### A.2.3.65.3 Fields

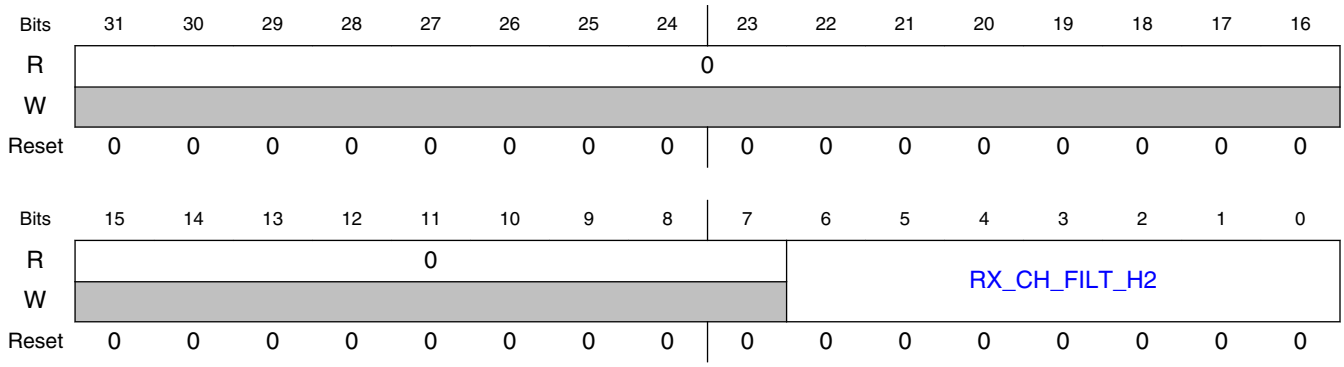
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>RX_CH_FILT_H1 | RX Channel Filter Coefficient 1<br>RX Channel Filter Coefficient 1, 6-bit signed fractional |

## A.2.3.66 Receive Channel Filter Coefficient 2 (RX_CHF_COEF_2)

### A.2.3.66.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_2 | 1A8h |

### A.2.3.66.2 Diagram




### A.2.3.66.3 Fields

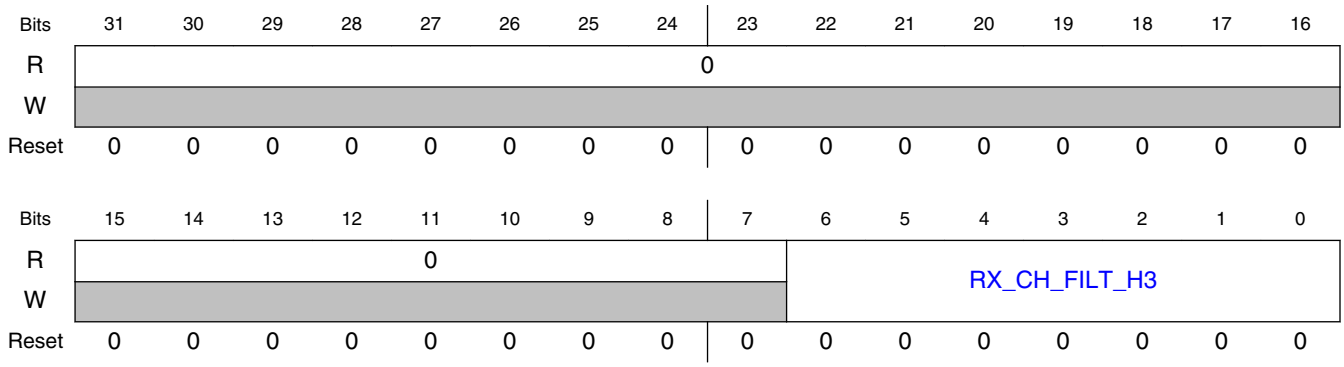
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H2 | RX Channel Filter Coefficient 2<br>RX Channel Filter Coefficient 2, 7-bit signed fractional |

## A.2.3.67 Receive Channel Filter Coefficient 3 (RX_CHF_COEF_3)

### A.2.3.67.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_3 | 1ACh |

## A.2.3.67.2 Diagram




## A.2.3.67.3 Fields

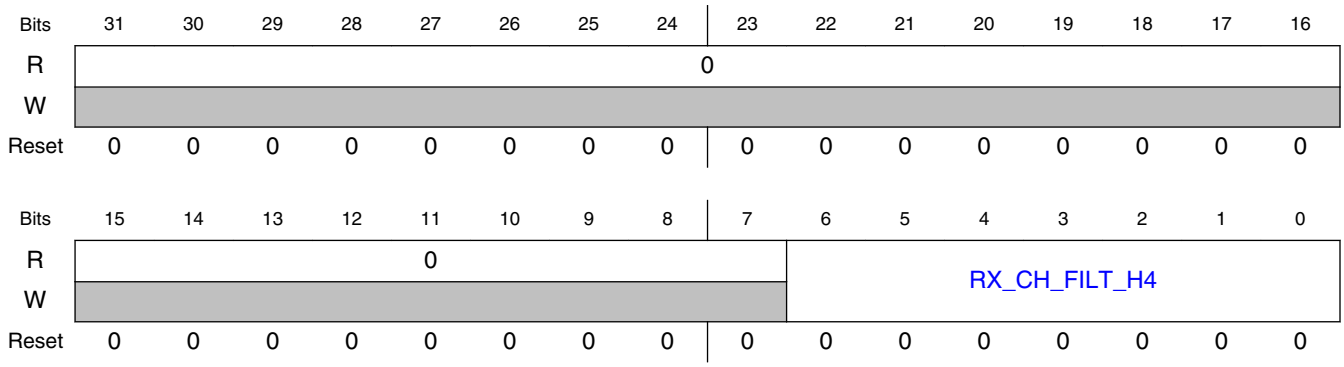
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H3 | RX Channel Filter Coefficient 3<br>RX Channel Filter Coefficient 3, 7-bit signed fractional |

# A.2.3.68 Receive Channel Filter Coefficient 4 (RX_CHF_COEF_4)

## A.2.3.68.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_4 | 1B0h |

### A.2.3.68.2 Diagram




### A.2.3.68.3 Fields

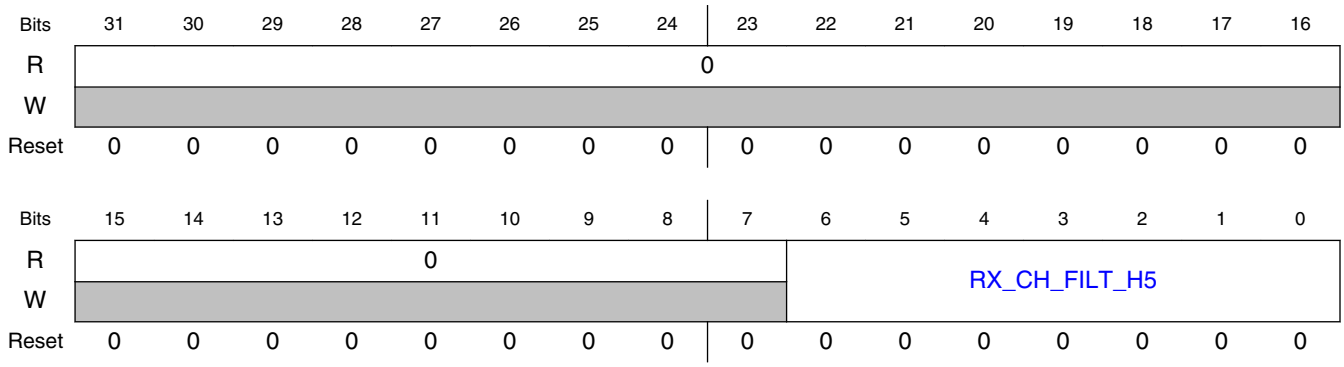
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H4 | RX Channel Filter Coefficient 4<br>RX Channel Filter Coefficient 4, 7-bit signed fractional |

## A.2.3.69 Receive Channel Filter Coefficient 5 (RX_CHF_COEF_5)

### A.2.3.69.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_5 | 1B4h |

## A.2.3.69.2 Diagram




## A.2.3.69.3 Fields

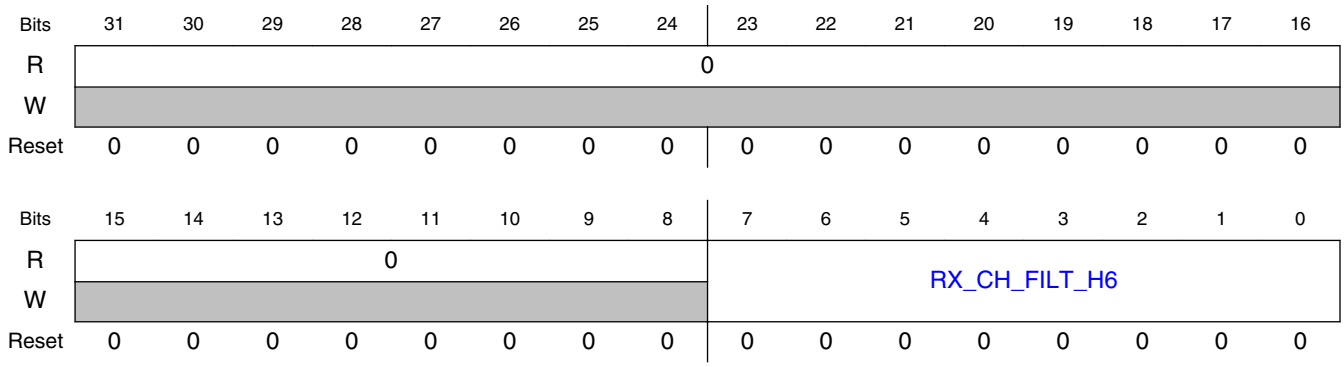
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>RX_CH_FILT_H5 | RX Channel Filter Coefficient 5<br>RX Channel Filter Coefficient 5, 7-bit signed fractional |

# A.2.3.70 Receive Channel Filter Coefficient 6 (RX_CHF_COEF_6)

## A.2.3.70.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_6 | 1B8h |

### A.2.3.70.2 Diagram




### A.2.3.70.3 Fields

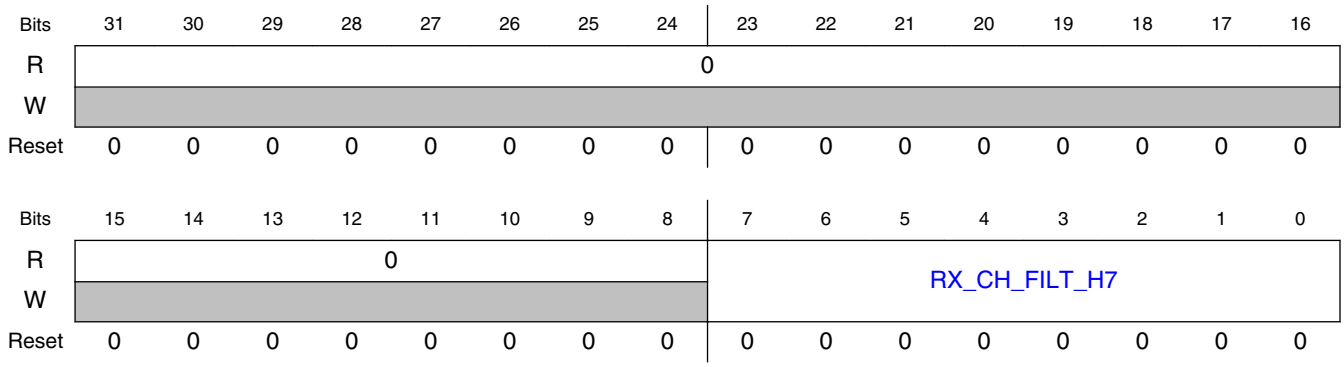
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>RX_CH_FILT_H6 | RX Channel Filter Coefficient 6<br>RX Channel Filter Coefficient 6, 8-bit signed fractional |

## A.2.3.71 Receive Channel Filter Coefficient 7 (RX_CHF_COEF_7)

### A.2.3.71.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_7 | 1BCh |

## A.2.3.71.2 Diagram




## A.2.3.71.3 Fields

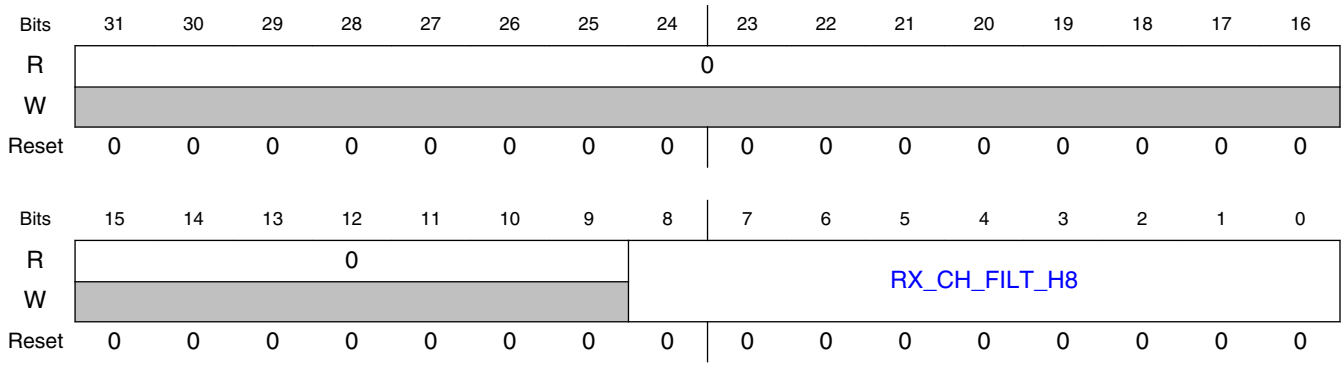
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>RX_CH_FILT_H7 | RX Channel Filter Coefficient 7<br>RX Channel Filter Coefficient 7, 8-bit signed fractional |

# A.2.3.72 Receive Channel Filter Coefficient 8 (RX_CHF_COEF_8)

## A.2.3.72.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_8 | 1C0h |

### A.2.3.72.2 Diagram




### A.2.3.72.3 Fields

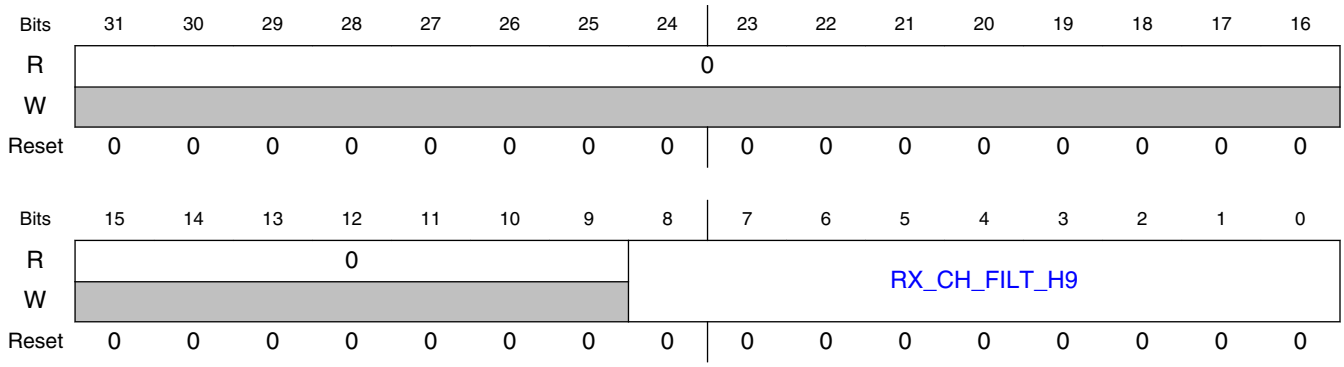
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>RX_CH_FILT_H8 | RX Channel Filter Coefficient 8<br>RX Channel Filter Coefficient 8, 9-bit signed fractional |

## A.2.3.73 Receive Channel Filter Coefficient 9 (RX_CHF_COEF_9)

### A.2.3.73.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_9 | 1C4h |

## A.2.3.73.2 Diagram




## A.2.3.73.3 Fields

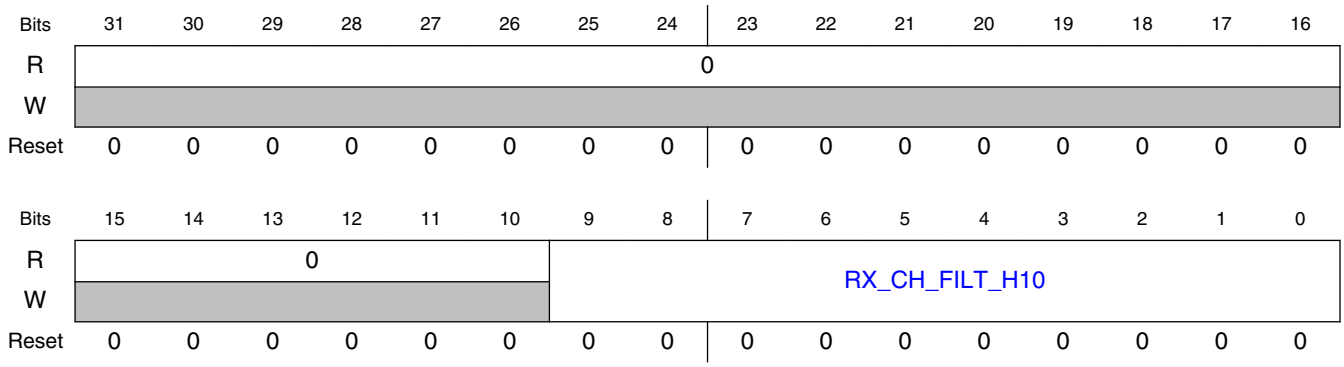
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>RX_CH_FILT_H9 | RX Channel Filter Coefficient 9<br>RX Channel Filter Coefficient 9, 9-bit signed fractional |

# A.2.3.74 Receive Channel Filter Coefficient 10 (RX_CHF_COEF_10)

## A.2.3.74.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_10 | 1C8h |

### A.2.3.74.2 Diagram




### A.2.3.74.3 Fields

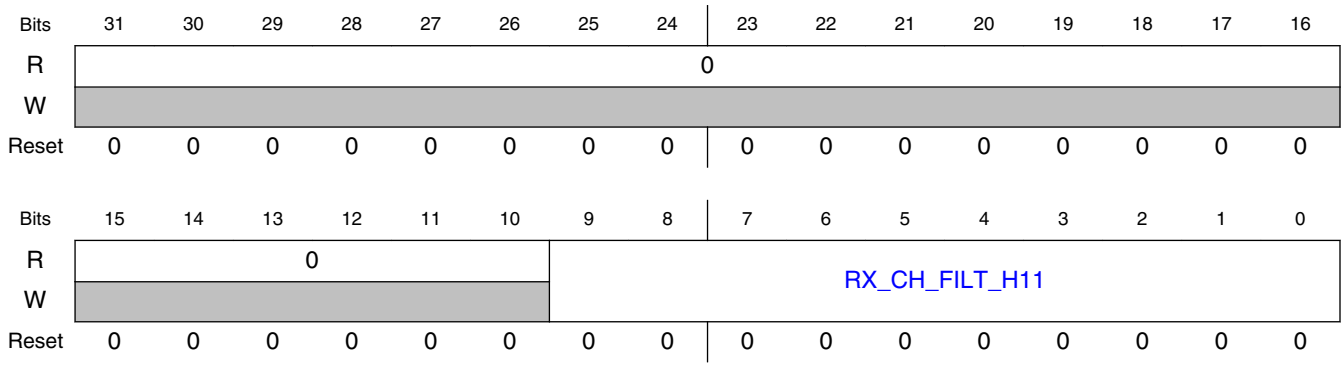
| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RX_CH_FILT_H10 | RX Channel Filter Coefficient 10<br>RX Channel Filter Coefficient 10, 10-bit signed fractional |

## A.2.3.75 Receive Channel Filter Coefficient 11 (RX_CHF_COEF_11)

### A.2.3.75.1 Offset

| Register | Offset |
|---|---|
| RX_CHF_COEF_11 | 1CCh |

### A.2.3.75.2 Diagram




### A.2.3.75.3 Fields

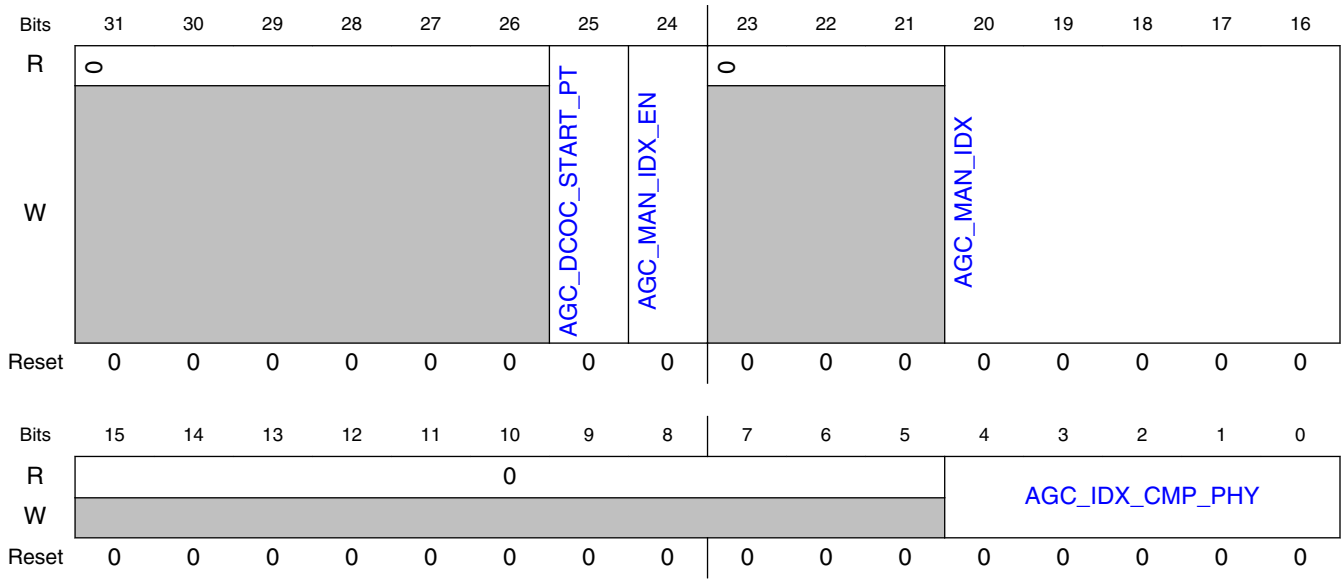
| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RX_CH_FILT_H11 | RX Channel Filter Coefficient 11<br>RX Channel Filter Coefficient 11, 10-bit signed fractional |

## A.2.3.76 AGC Manual AGC Index (AGC_MAN_AGC_IDX)

### A.2.3.76.1 Offset

| Register | Offset |
|---|---|
| AGC_MAN_AGC_IDX | 1D0h |

## A.2.3.76.2 Diagram




## A.2.3.76.3 Fields

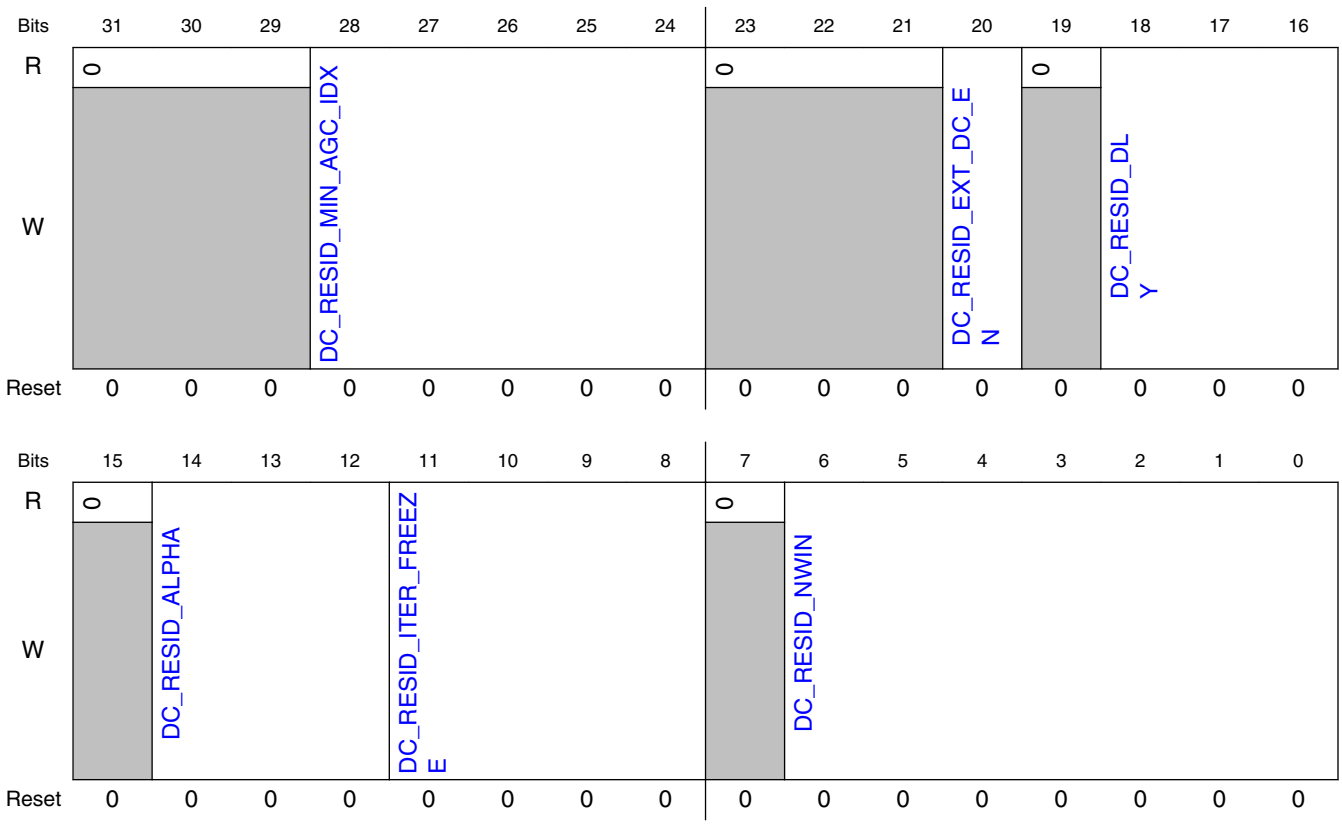
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>AGC_DCOC_START_PT | AGC DCOC Start Point<br>When set, the starting value for the AGC gain table index after DCOC calibration will be AGC_MAN_IDX instead of index 26. |
| 24<br>AGC_MAN_IDX_EN | AGC Manual Index Enable<br>When set, the AGC gain table index is overridden using AGC_MAN_IDX. |
| 23-21<br>— | Reserved. |
| 20-16<br>AGC_MAN_IDX | AGC Manual Index<br>AGC gain table index override value (when AGC_MAN_IDX_EN is set), or AGC gain table index starting point after DCOC calibration (when AGC_DCOC_START_PT is set). |
| 15-5<br>— | Reserved. |
| 4-0<br>AGC_IDX_CMP_PHY | AGC Index Compare threshold for PHY<br>The AGC's agc_thresh_below output (which is connected to the PHY's pd_mode_switch input) is asserted when the current AGC index is less than this threshold. |

# A.2.3.77 DC Residual Control (DC_RESID_CTRL)

## A.2.3.77.1 Offset

| Register | Offset |
|---|---|
| DC_RESID_CTRL | 1D4h |

## A.2.3.77.2 Diagram




## A.2.3.77.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-24<br>DC_RESID_MIN_AGC_IDX | DC Residual Minimum AGC Table Index<br>Specifies the minimum AGC table index value at which DC residual can be enabled. E.g., if this is 5'd0, then the DC residual is enabled for all AGC gain table indexes (assuming RX_DC_RESID_EN is set) if this is 5'd20, then tracking is enabled for AGC gain table indexes 20-26 (assuming RX_DC_RESID_EN is set). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-21<br>— | Reserved. |
| 20<br>DC_RESID_EXT_DC_EN | DC Residual External DC Enable<br>0b - External DC disable. The DC Residual activates at a delay specified by DC_RESID_DLY after an AGC gain change pulse. The DC Residual is initialized with a DC offset of 0.<br>1b - External DC enable. The DC residual activates after the DCOC's tracking hold timer expires. The DC Residual is initialized with the DC estimate from the DCOC tracking estimator. |
| 19<br>— | Reserved. |
| 18-16<br>DC_RESID_DLY | DC Residual Delay<br>The delay from activation of the DC residual block to the time when it starts processing. For a 32MHz reference clock, the delay in microseconds matches the value programmed. For other clock frequencies, the delay is scaled based on the clock period. Supported values: 0-7. This delay is only used when EXT_DC_EN=0. |
| 15<br>— | Reserved. |
| 14-12<br>DC_RESID_ALPHA | DC Residual Alpha<br>The Alpha parameter controls the rate at which the DC estimate is updated. The update factor is 2^(-Alpha). |
| 11-8<br>DC_RESID_ITER_FREEZE | DC Residual Iteration Freeze<br>Number of windows of DC_RESID_NWIN samples before the DC is frozen. Supported values: 1-8 |
| 7<br>— | Reserved. |
| 6-0<br>DC_RESID_NWIN | DC Residual NWIN<br>Number of samples in a window. |

## A.2.3.78 DC Residual Estimate (DC_RESID_EST)

### A.2.3.78.1 Offset

| Register | Offset |
|---|---|
| DC_RESID_EST | 1D8h |

## A.2.3.78.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DC_RESID_OFFSET_Q | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | DC_RESID_OFFSET_I | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.3.78.3 Fields

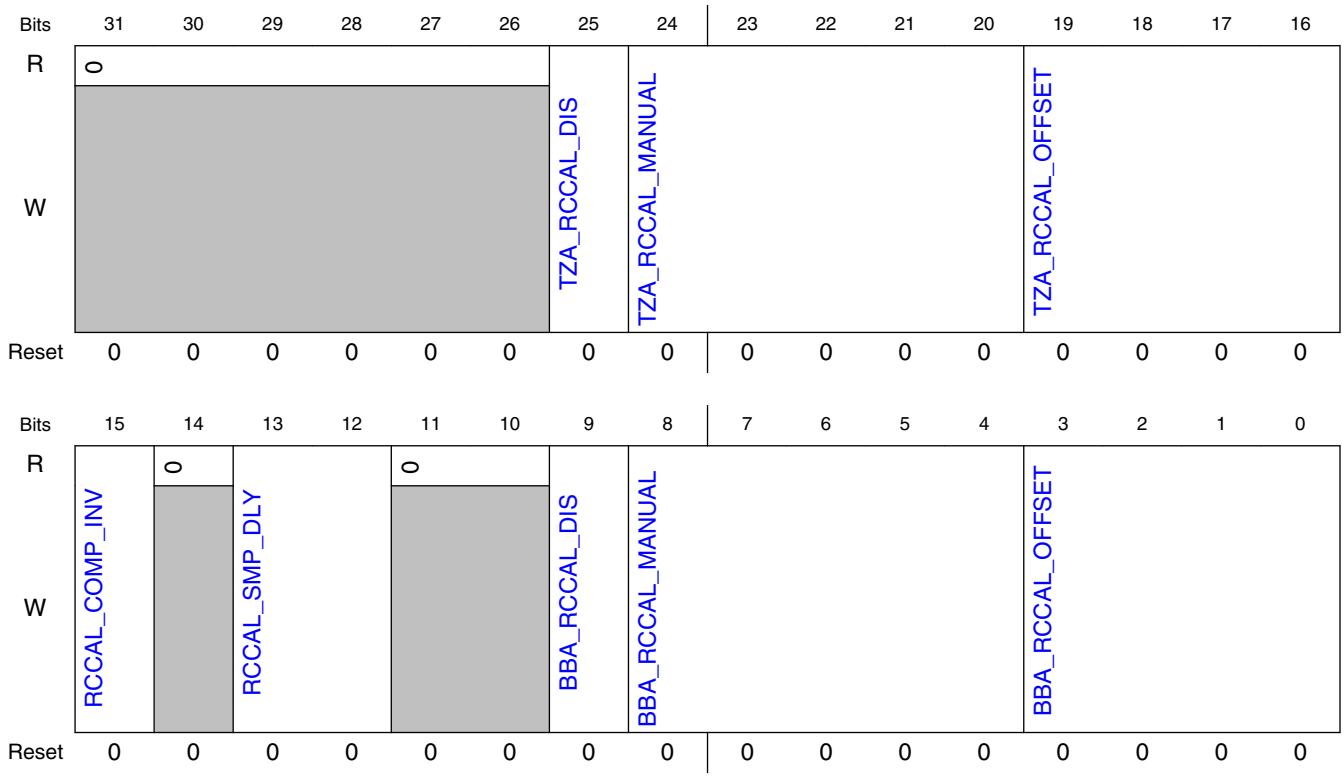
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>DC_RESID_OFFSET_Q | DC Residual Offset Q<br>Reflects the current DC residual offset estimate for Q channel. Format is s11.1. This is provided for debug and characterization purposes only. |
| 15-13<br>— | Reserved. |
| 12-0<br>DC_RESID_OFFSET_I | DC Residual Offset I<br>Reflects the current DC residual offset estimate for I channel. Format is s11.1. This is provided for debug and characterization purposes only. |

# A.2.3.79 RX RC Calibration Control0 (RX_RCCAL_CTRL0)

## A.2.3.79.1 Offset

| Register | Offset |
|---|---|
| RX_RCCAL_CTRL0 | 1DCh |

## A.2.3.79.2 Diagram




## A.2.3.79.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>TZA_RCCAL_DIS | TZA RC Calibration Disable<br>0b - TZA RC Calibration is enabled<br>1b - TZA RC Calibration is disabled |
| 24-20<br>TZA_RCCAL_MANUAL | TZA RC Calibration manual value<br>If TZA_RCCAL_DIS bit is set, this value is used for the TZA calibration. |
| 19-16<br>TZA_RCCAL_OFFSET | TZA RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the TZA. Format is s3. |
| 15<br>RCCAL_COMP_INV | RC Calibration comp_out Invert<br>0b - The comp_out signal polarity is NOT inverted<br>1b - The comp_out signal polarity is inverted |
| 14<br>— | Reserved. |
| 13-12 | RC Calibration Sample Delay |

*Table continues on the next page...*

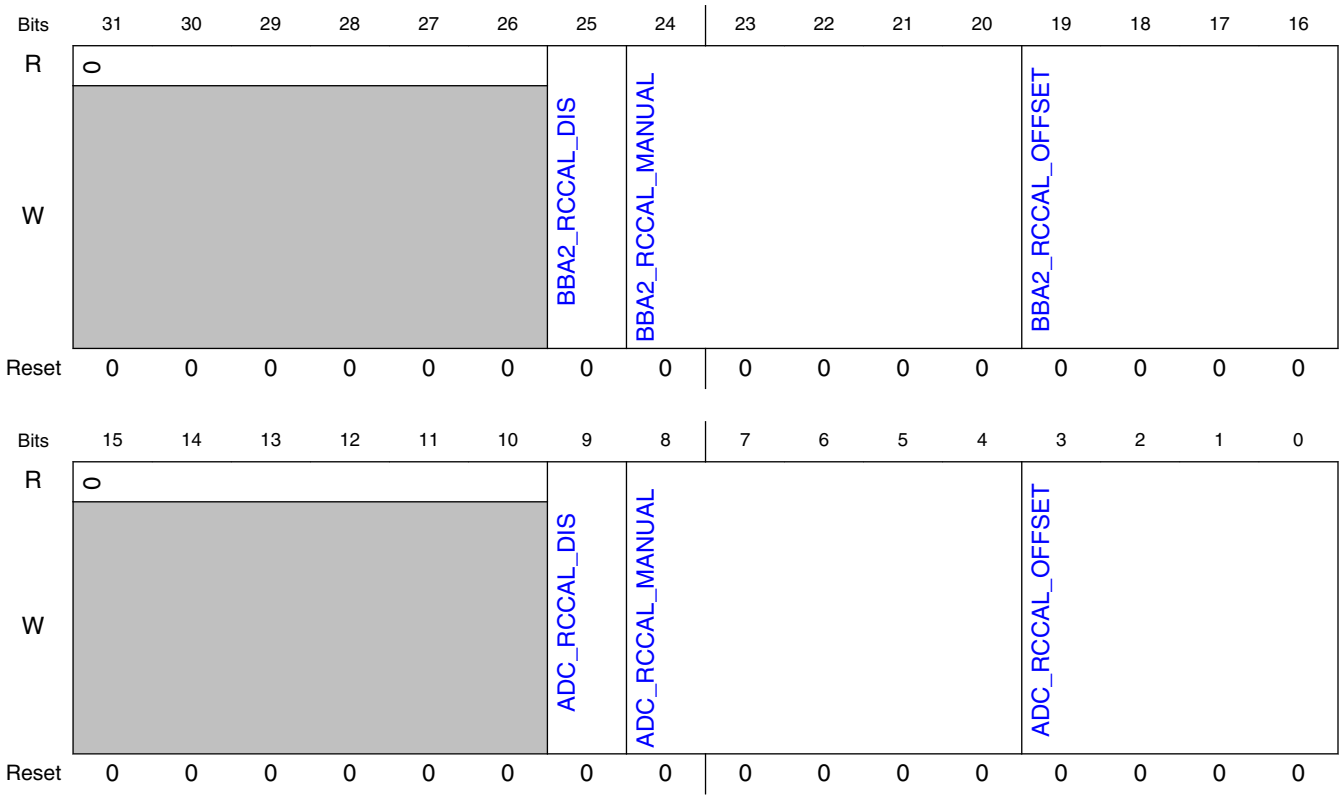
| Field | Function |
|---|---|
| RCCAL_SMP_DLY | 00b - The comp_out signal is sampled 0 clk cycle after sample signal is deasserted<br>01b - The comp_out signal is sampled 1 clk cycle after sample signal is deasserted<br>10b - The comp_out signal is sampled 2 clk cycle after sample signal is deasserted<br>11b - The comp_out signal is sampled 3 clk cycle after sample signal is deasserted |
| 11-10<br>— | Reserved. |
| 9<br>BBA_RCCAL_DIS | BBA RC Calibration Disable<br>0b - BBA RC Calibration is enabled<br>1b - BBA RC Calibration is disabled |
| 8-4<br>BBA_RCCAL_MANUAL | BBA RC Calibration manual value<br>If BBA_RCCAL_DIS bit is set, this value is used for the BBA calibration. |
| 3-0<br>BBA_RCCAL_OFFSET | BBA RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the BBA. Format is s3. |

# A.2.3.80 RX RC Calibration Control1 (RX_RCCAL_CTRL1)

## A.2.3.80.1 Offset

| Register | Offset |
|---|---|
| RX_RCCAL_CTRL1 | 1E0h |

## A.2.3.80.2 Diagram




## A.2.3.80.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>BBA2_RCCAL_DIS | BBA2 RC Calibration Disable<br>0b - BBA2 RC Calibration is enabled<br>1b - BBA2 RC Calibration is disabled |
| 24-20<br>BBA2_RCCAL_MANUAL | BBA2 RC Calibration manual value<br>If BBA2_RCCAL_DIS bit is set, this value is used for the BBA2 calibration. |
| 19-16<br>BBA2_RCCAL_OFFSET | BBA2 RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the BBA2. Format is s3. |
| 15-10<br>— | Reserved. |
| 9<br>ADC_RCCAL_DIS | ADC RC Calibration Disable<br>0b - ADC RC Calibration is enabled<br>1b - ADC RC Calibration is disabled |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8-4<br>ADC_RCCAL_MANUAL | ADC RC Calibration manual value<br>If ADC_RCCAL_DIS bit is set, this value is used for the ADC calibration. |
| 3-0<br>ADC_RCCAL_OFFSET | ADC RC Calibration value offset<br>Offset added to the RC Calibration code to determine the calibration value used for the ADC. Format is s3. |

## A.2.3.81 RX RC Calibration Status (RX_RCCAL_STAT)

### A.2.3.81.1 Offset

| Register | Offset |
|---|---|
| RX_RCCAL_STAT | 1E4h |

### A.2.3.81.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | TZA_RCCAL | | | | | BBA_RCCAL | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BBA2_RCCAL | | | | | ADC_RCCAL | | | | | RCCAL_CODE | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### A.2.3.81.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-21<br>TZA_RCCAL | TZA RC Calibration<br>The RC Calibration value used for TZA |
| 20-16<br>BBA_RCCAL | BBA RC Calibration<br>The RC Calibration value used for BBA |
| 15 | Reserved. |

*Table continues on the next page...*

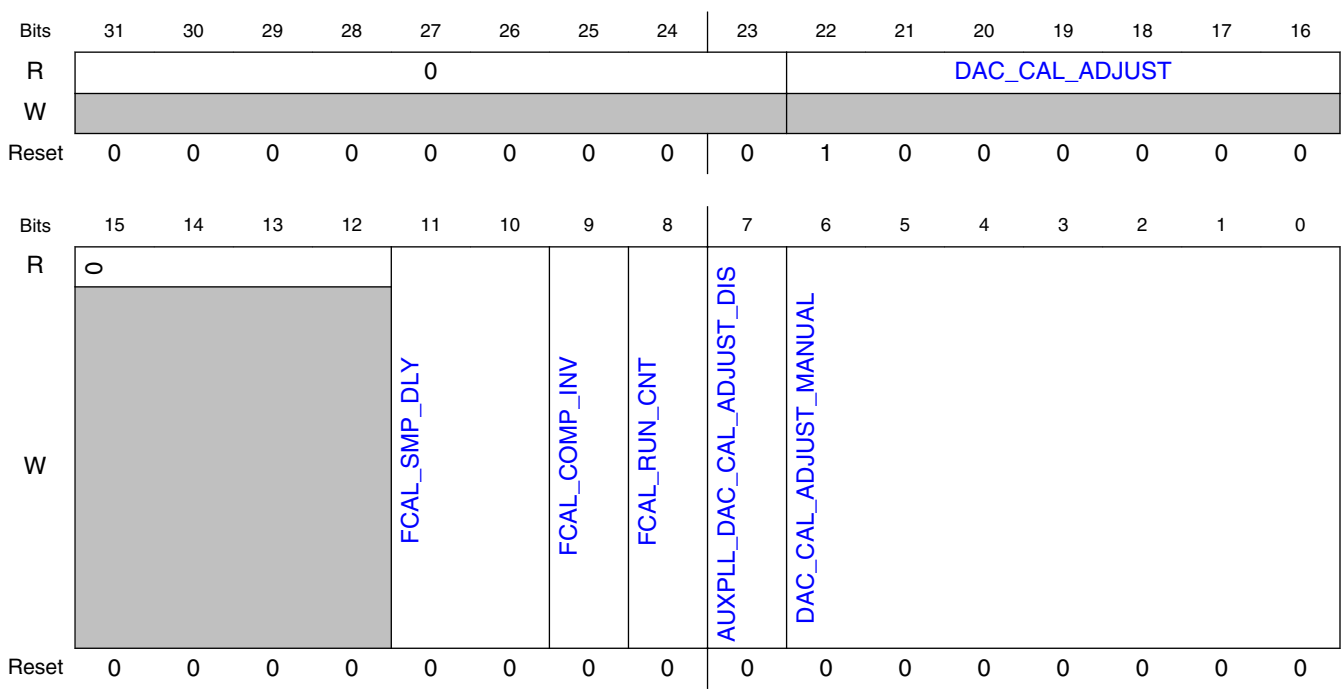
| Field | Function |
|---|---|
| — | |
| 14-10<br>BBA2_RCCAL | BBA2 RC Calibration<br>The RC Calibration value used for BBA2 |
| 9-5<br>ADC_RCCAL | ADC RC Calibration<br>The RC Calibration value used for ADC |
| 4-0<br>RCCAL_CODE | RC Calibration code<br>The RC Calibration code currently applied to the calibration circuit |

## A.2.3.82 Aux PLL Frequency Calibration Control (AUXPLL_FCAL_CTRL)

### A.2.3.82.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CTRL | 1E8h |

### A.2.3.82.2 Diagram



### A.2.3.82.3 Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved. |
| 22-16<br>DAC_CAL_ADJUST | Aux PLL DAC Calibration Adjust value<br>The DAC calibration adjust value applied to the calibration circuit. |
| 15-12<br>— | Reserved. |
| 11-10<br>FCAL_SMP_DLY | Aux PLL Frequency Calibration Sample Delay<br>00b - The count signal is sampled 1 clk cycle after fcal_run signal is deasserted<br>01b - The count signal is sampled 2 clk cycle after fcal_run signal is deasserted<br>10b - The count signal is sampled 3 clk cycle after fcal_run signal is deasserted<br>11b - The count signal is sampled 4 clk cycle after fcal_run signal is deasserted |
| 9<br>FCAL_COMP_INV | Aux PLL Frequency Calibration Comparison Invert<br>0b - (Default) The comparison associated with the count is not inverted.<br>1b - The comparison associated with the count is inverted |
| 8<br>FCAL_RUN_CNT | Aux PLL Frequency Calibration Run Count<br>0b - Run count is 256 clock cycles<br>1b - Run count is 512 clock cycles |
| 7<br>AUXPLL_DAC_CAL_ADJUST_DIS | Aux PLL Frequency Calibration Disable<br>0b - Calibration is enabled<br>1b - Calibration is disabled |
| 6-0<br>DAC_CAL_ADJUST_MANUAL | Aux PLL Frequency DAC Calibration Adjust Manual value<br>If AUXPLL_DAC_CAL_ADJUST_DIS bit is set, this DAC calibration adjust value is applied to the calibration circuit. |

## A.2.3.83 Aux PLL Frequency Calibration Count 6 (AUXPLL_FCAL_CNT6)

### A.2.3.83.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT6 | 1ECh |

### A.2.3.83.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_BESTDIFF | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_6 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.83.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_BESTDIFF | Aux PLL Frequency Calibration Best Difference<br>The smallest absolute difference between the count value output by the calibration circuit and the expected count value found during the calibration sequence. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_6 | Aux PLL Frequency Calibration Count 6<br>The count value output by the calibration circuit for calibration phase 6. |

## A.2.3.84 Aux PLL Frequency Calibration Count 5 and 4 (AUXPLL_FCAL_CNT5_4)

### A.2.3.84.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT5_4 | 1F0h |

### A.2.3.84.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_5 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_4 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.84.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_5 | Aux PLL Frequency Calibration Count 5<br>The count value output by the calibration circuit for calibration phase 5. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_4 | Aux PLL Frequency Calibration Count 4<br>The count value output by the calibration circuit for calibration phase 4. |

## A.2.3.85 Aux PLL Frequency Calibration Count 3 and 2 (AUXPLL_FCAL_CNT3_2)

### A.2.3.85.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT3_2 | 1F4h |

### A.2.3.85.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_3 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_2 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.85.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_3 | Aux PLL Frequency Calibration Count 3<br>The count value output by the calibration circuit for calibration phase 3. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_2 | Aux PLL Frequency Calibration Count 2<br>The count value output by the calibration circuit for calibration phase 2. |

## A.2.3.86 Aux PLL Frequency Calibration Count 1 and 0 (AUXPLL_FCAL_CNT1_0)

### A.2.3.86.1 Offset

| Register | Offset |
|---|---|
| AUXPLL_FCAL_CNT1_0 | 1F8h |

### A.2.3.86.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_1 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FCAL_COUNT_0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.3.86.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25-16<br>FCAL_COUNT_1 | Frequency Calibration Count 1<br>The count value output by the calibration circuit for calibration phase 1. |
| 15-10<br>— | Reserved. |
| 9-0<br>FCAL_COUNT_0 | Frequency Calibration Count 0<br>The count value output by the calibration circuit for calibration phase 0. |

## A.2.4 XCVR_TSM register descriptions

### A.2.4.1 XCVR_TSM_ADDR Memory map

XCVR_TSM base address: 4103_02C0h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TSM CONTROL (CTRL) | 32 | RW | FF00_4000h |
| 4h | TSM END OF SEQUENCE (END_OF_SEQ) | 32 | RW | 6766_6A63h |
| 8h | PA POWER (PA_POWER) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| Ch | PA RAMP TABLE 0 (PA_RAMP_TBL0) | 32 | RW | 0604_0201h |
| 10h | PA RAMP TABLE 1 (PA_RAMP_TBL1) | 32 | RW | 1410_0C08h |
| 14h | PA RAMP TABLE 2 (PA_RAMP_TBL2) | 32 | RW | 2822_1C18h |
| 18h | PA RAMP TABLE 3 (PA_RAMP_TBL3) | 32 | RW | 3C36_302Ch |
| 24h | TSM RECYCLE COUNT (RECYCLE_COUNT) | 32 | RW | 001A_0464h |
| 28h | TSM FAST WARMUP CONTROL 1 (FAST_CTRL1) | 32 | RW | 0000_FF00h |
| 2Ch | TSM FAST WARMUP CONTROL 2 (FAST_CTRL2) | 32 | RW | FFFF_FFFFh |
| 30h | TSM_TIMING00 (TIMING00) | 32 | RW | 6700_6A00h |
| 34h | TSM_TIMING01 (TIMING01) | 32 | RW | 6700_6A00h |
| 38h | TSM_TIMING02 (TIMING02) | 32 | RW | 6700_FFFFh |
| 3Ch | TSM_TIMING03 (TIMING03) | 32 | RW | 6700_6A00h |
| 40h | TSM_TIMING04 (TIMING04) | 32 | RW | 6700_6A00h |
| 44h | TSM_TIMING05 (TIMING05) | 32 | RW | 6700_6A00h |
| 48h | TSM_TIMING06 (TIMING06) | 32 | RW | 6700_6A00h |
| 4Ch | TSM_TIMING07 (TIMING07) | 32 | RW | 0500_0500h |
| 50h | TSM_TIMING08 (TIMING08) | 32 | RW | 0300_0300h |
| 54h | TSM_TIMING09 (TIMING09) | 32 | RW | 0300_0300h |
| 58h | TSM_TIMING10 (TIMING10) | 32 | RW | 6703_6A03h |
| 5Ch | TSM_TIMING11 (TIMING11) | 32 | RW | FFFF_6A03h |
| 60h | TSM_TIMING12 (TIMING12) | 32 | RW | 6703_FFFFh |
| 64h | TSM_TIMING13 (TIMING13) | 32 | RW | 1600_4A00h |
| 68h | TSM_TIMING14 (TIMING14) | 32 | RW | 672F_645Fh |
| 6Ch | TSM_TIMING15 (TIMING15) | 32 | RW | 6703_6A03h |
| 70h | TSM_TIMING16 (TIMING16) | 32 | RW | 671A_FFFFh |
| 74h | TSM_TIMING17 (TIMING17) | 32 | RW | FFFF_6A5Ah |
| 78h | TSM_TIMING18 (TIMING18) | 32 | RW | 6705_6A05h |
| 7Ch | TSM_TIMING19 (TIMING19) | 32 | RW | 1605_4A05h |
| 80h | TSM_TIMING20 (TIMING20) | 32 | RW | 6705_6A05h |
| 84h | TSM_TIMING21 (TIMING21) | 32 | RW | 6704_6A04h |
| 88h | TSM_TIMING22 (TIMING22) | 32 | RW | 6704_FFFFh |
| 8Ch | TSM_TIMING23 (TIMING23) | 32 | RW | FFFF_6A04h |
| 90h | TSM_TIMING24 (TIMING24) | 32 | RW | 1600_4A00h |
| 94h | TSM_TIMING25 (TIMING25) | 32 | RW | 671B_FFFFh |
| 98h | TSM_TIMING26 (TIMING26) | 32 | RW | FFFF_6A5Ah |
| 9Ch | TSM_TIMING27 (TIMING27) | 32 | RW | 671E_FFFFh |
| A0h | TSM_TIMING28 (TIMING28) | 32 | RW | 1F1E_FFFFh |
| A4h | TSM_TIMING29 (TIMING29) | 32 | RW | 671C_FFFFh |
| A8h | TSM_TIMING30 (TIMING30) | 32 | RW | 671E_FFFFh |
| ACh | TSM_TIMING31 (TIMING31) | 32 | RW | 671D_FFFFh |
| B0h | TSM_TIMING32 (TIMING32) | 32 | RW | 671B_FFFFh |

*Table continues on the next page...*

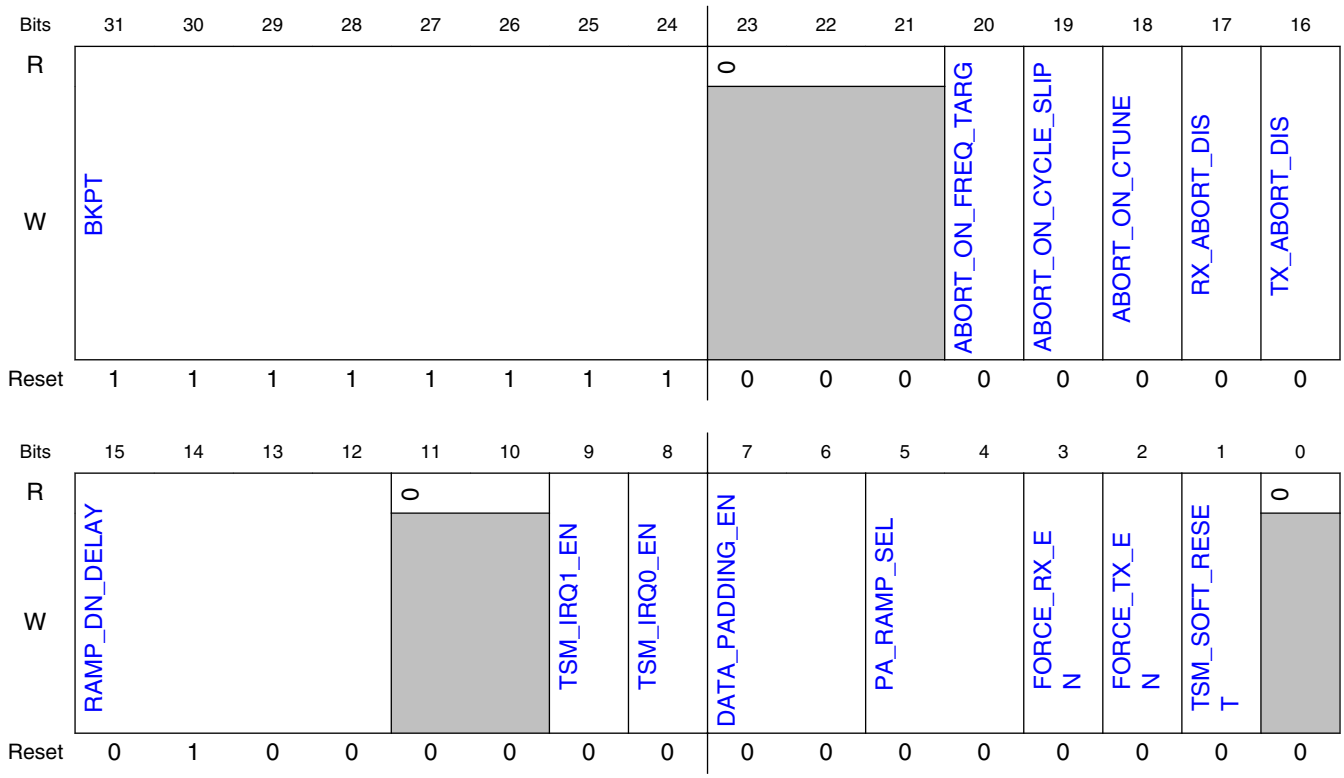
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| B4h | TSM_TIMING33 (TIMING33) | 32 | RW | 671E_FFFFh |
| B8h | TSM_TIMING34 (TIMING34) | 32 | RW | 6705_6A05h |
| BCh | TSM_TIMING35 (TIMING35) | 32 | RW | FFFF_6A5Dh |
| C0h | TSM_TIMING36 (TIMING36) | 32 | RW | 6764_FFFFh |
| C4h | TSM_TIMING37 (TIMING37) | 32 | RW | 6564_FFFFh |
| C8h | TSM_TIMING38 (TIMING38) | 32 | RW | 670C_6A40h |
| CCh | TSM_TIMING39 (TIMING39) | 32 | RW | 6764_FFFFh |
| D0h | TSM_TIMING40 (TIMING40) | 32 | RW | 6724_FFFFh |
| D4h | TSM_TIMING41 (TIMING41) | 32 | RW | 2524_FFFFh |
| D8h | TSM_TIMING42 (TIMING42) | 32 | RW | FFFF_FFFFh |
| DCh | TSM_TIMING43 (TIMING43) | 32 | RW | FFFF_FFFFh |
| E0h | TSM_TIMING44 (TIMING44) | 32 | RW | FFFF_FFFFh |
| E4h | TSM_TIMING45 (TIMING45) | 32 | RW | FFFF_FFFFh |
| E8h | TSM_TIMING46 (TIMING46) | 32 | RW | FFFF_FFFFh |
| ECh | TSM_TIMING47 (TIMING47) | 32 | RW | FFFF_FFFFh |
| F0h | TSM_TIMING48 (TIMING48) | 32 | RW | FFFF_FFFFh |
| F4h | TSM_TIMING49 (TIMING49) | 32 | RW | FFFF_FFFFh |
| F8h | TSM_TIMING50 (TIMING50) | 32 | RW | FFFF_FFFFh |
| FCh | TSM_TIMING51 (TIMING51) | 32 | RW | 6703_FFFFh |
| 100h | TSM_TIMING52 (TIMING52) | 32 | RW | 1504_FFFFh |
| 104h | TSM_TIMING53 (TIMING53) | 32 | RW | 6704_FFFFh |
| 108h | TSM_TIMING54 (TIMING54) | 32 | RW | 1504_FFFFh |
| 10Ch | TSM_TIMING55 (TIMING55) | 32 | RW | 671E_FFFFh |
| 110h | TSM_TIMING56 (TIMING56) | 32 | RW | 671E_FFFFh |
| 114h | TSM_TIMING57 (TIMING57) | 32 | RW | 1A03_FFFFh |
| 118h | TSM_TIMING58 (TIMING58) | 32 | RW | FFFF_6A03h |
| 11Ch | TSM OVERRIDE REGISTER 0 (OVRD0) | 32 | RW | 0000_0000h |
| 120h | TSM OVERRIDE REGISTER 1 (OVRD1) | 32 | RW | 0000_0000h |
| 124h | TSM OVERRIDE REGISTER 2 (OVRD2) | 32 | RW | 0000_0000h |
| 128h | TSM OVERRIDE REGISTER 3 (OVRD3) | 32 | RW | 0000_0000h |

## A.2.4.2 TSM CONTROL (CTRL)

### A.2.4.2.1 Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

### A.2.4.2.2 Diagram




### A.2.4.2.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BKPT | TSM Breakpoint<br>Temporarily halt a TSM sequence during the warmup or warmdown phase. When the TSM counter matches the value of BKPT[7:0], breakpoint will take effect and the TSM counter will stop and hold its count. Breakpoint will remain in effect as long as BKPT[7:0] matches the TSM counter value. The TSM Breakpoint can be lifted by modifying the contents of this register. The default value of this register, 0xFF, is greater than the length of the longest possible sequence, so a breakpoint will never be triggered unless BKPT[7:0] is programmed to a value less than the length of sequence. |
| 23-21<br>— | Reserved |
| 20<br>ABORT_ON_FREQ_TARG | Abort On Frequency Target Lock Detect Failure<br>0b - don't allow TSM abort on Frequency Target Unlock Detect<br>1b - allow TSM abort on Frequency Target Unlock Detect |
| 19<br>ABORT_ON_CYCLE_SLIP | Abort On Cycle Slip Lock Detect Failure<br>0b - don't allow TSM abort on Cycle Slip Unlock Detect<br>1b - allow TSM abort on Cycle Slip Unlock Detect |
| 18 | Abort On Coarse Tune Lock Detect Failure<br>0b - don't allow TSM abort on Coarse Tune Unlock Detect |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ABORT_ON_CTUNE | 1b - allow TSM abort on Coarse Tune Unlock Detect |
| 17<br>RX_ABORT_DIS | Receive Abort Disable<br>RX Abort disable. When set, prevents PLL unlock events during RX sequences from aborting the sequence. |
| 16<br>TX_ABORT_DIS | Transmit Abort Disable<br>TX Abort disable. When set, prevents PLL unlock events during TX sequences from aborting the sequence. |
| 15-12<br>RAMP_DN_DELAY | PA Ramp Down Delay<br>Delays the start of the PA Ramp Down, relative to the start of the TSM warmdown, by *N* microseconds, where *N*=RAMP_DN_DELAY. Range is 0 to 15us. |
| 11-10<br>— | Reserved |
| 9<br>TSM_IRQ1_EN | TSM_IRQ1 Enable/Disable bit<br>TSM_IRQ1 Enable/Disable bit. Intended for debug purposes only.<br>0b - TSM_IRQ1 is disabled<br>1b - TSM_IRQ1 is enabled |
| 8<br>TSM_IRQ0_EN | TSM_IRQ0 Enable/Disable bit<br>TSM_IRQ0 Enable/Disable bit. Intended for debug purposes only.<br>0b - TSM_IRQ0 is disabled<br>1b - TSM_IRQ0 is enabled |
| 7-6<br>DATA_PADDING_EN | Data Padding Enable<br>Enables TX Data Padding. Data padding works in conjunction with PA ramping to minimize spectral transients during PA turn-on and turn-off. The nature of the data padding depends on the setting of XCVR_CTRL[PROTOCOL].<br>00b - Disable TX Data Padding<br>01b - Enable TX Data Padding |
| 5-4<br>PA_RAMP_SEL | PA Ramp Selection<br>Selects the ramp-rate, and thus the duration, for PA ramping. Ramp-rate is the rate at which the PA ramping logic steps through the PA Ramp Table. There are always 16 ramp steps, if ramping is enabled.<br>The following table lists the total ramp duration for each PA_RAMP_SEL setting:<br>0: No ramp<br>1: 1.0µs<br>2: 2.0µs<br>3: 4.0µs<br>The following table lists the duration of each ramp step for each PA_RAMP_SEL setting:<br>0: No ramp<br>1: 2 Reference Clocks (0.0625µs)<br>2: 4 Reference Clocks (0.125µs)<br>3: 8 Reference Clocks (0.25µs) |
| 3<br>FORCE_RX_EN | Force Receive Enable<br>Direct software control to launch a RX TSM sequence. Initiates RX Warmup on a 0 to 1 transition and RX Warmdown on a 1 to 0 transition.<br>0b - TSM Idle<br>1b - TSM executes a RX sequence |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 2<br>FORCE_TX_EN | Force Transmit Enable<br>Direct software control to launch a TX TSM sequence. Initiates a TX Warmup sequence on a 0 to 1 transition and a TX Warmdown sequence on a 1 to 0 transition.<br>0b - TSM Idle<br>1b - TSM executes a TX sequence |
| 1<br>TSM_SOFT_RESET | TSM Soft Reset<br>Writing 1 to this bit returns TSM to its IDLE state, with TSM_COUNT=0. Writing 0 to this bit removes the forced-reset condition and resumes normal operation. The bit is not self-clearing. For contingency purposes only<br>0b - TSM Soft Reset removed. Normal operation.<br>1b - TSM Soft Reset engaged. TSM forced to IDLE, and holds there until the bit is cleared. |
| 0<br>— | Reserved |

## A.2.4.3 TSM END OF SEQUENCE (END_OF_SEQ)

### A.2.4.3.1 Offset

| Register | Offset |
|---|---|
| END_OF_SEQ | 4h |

### A.2.4.3.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | END_OF_RX_WD | | | | | | | | END_OF_RX_WU | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | END_OF_TX_WD | | | | | | | | END_OF_TX_WU | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

### A.2.4.3.3 Fields

| Field | Function |
|---|---|
| 31-24 | End of RX Warmdown |

*Table continues on the next page...*

| Field | Function |
|---|---|
| END_OF_RX_WD | This register defines the point at which the TSM RX sequence warmdown completes, and the TSM returns to idle. The duration of the TSM warmdown phase is determined by: END_OF_RX_WD – END_OF_RX_WU. |
| 23-16<br>END_OF_RX_WU | End of RX Warmup<br>This register defines the length of the TSM RX warmup sequence. After the assertion of a RX sequence-initiating event, when the TSM counter reaches the count matching this register, it will stop and hold its count, and the TSM will transition from the WARMUP to the ON phase. |
| 15-8<br>END_OF_TX_WD | End of TX Warmdown<br>This register defines the point at which the TSM TX sequence warmdown completes, and the TSM returns to idle. The duration of the TSM warmdown phase is determined by: END_OF_TX_WD – END_OF_TX_WU. |
| 7-0<br>END_OF_TX_WU | End of TX Warmup<br>This register defines the length of the TSM TX warmup sequence. After the assertion of a TX sequence-initiating event, when the TSM counter reaches the count matching this register, it will stop and hold its count, and the TSM will transition from the WARMUP to the ON phase. |

## A.2.4.4 PA POWER (PA_POWER)

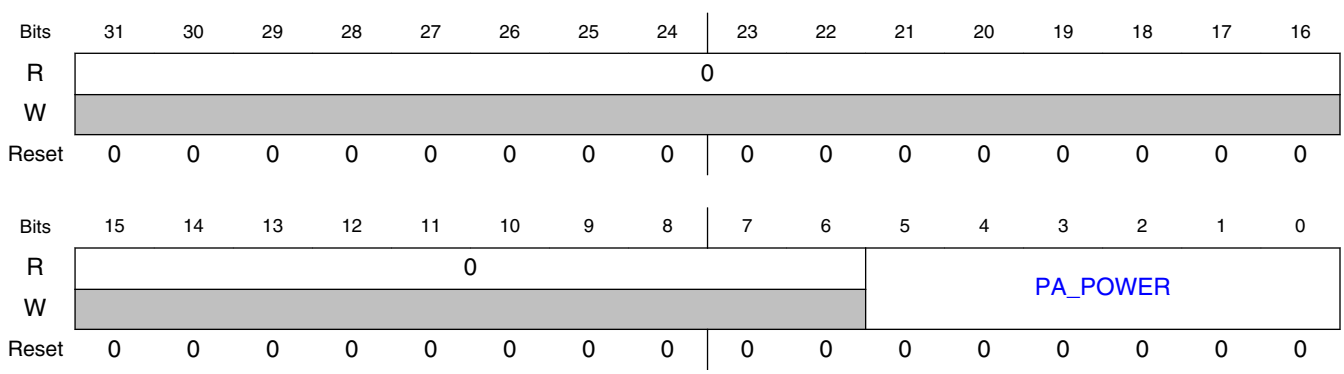
### A.2.4.4.1 Offset

| Register | Offset |
|---|---|
| PA_POWER | 8h |

### A.2.4.4.2 Function

This contents of this register are used as PA target power when XCVR_CTRL[TGT_PWR_SRC] = 0

### A.2.4.4.3 Diagram



#### A.2.4.4.4 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0<br>PA_POWER | PA POWER<br>This contents of this register are used as PA target power when XCVR_CTRL[TGT_PWR_SRC] = 0.The valid values for this field are 0,1,2,4,6,8,…,62 (in increasing order of PA power). That is, above 1, only even numbers are permitted in the PA_POWER bitfield. Odd values of 3..63 are not permitted and will result in unexpected behavior. |

### A.2.4.5 PA RAMP TABLE 0 (PA_RAMP_TBL0)

#### A.2.4.5.1 Offset

| Register | Offset |
|---|---|
| PA_RAMP_TBL0 | Ch |

#### A.2.4.5.2 Function

PA Ramp Table 0

#### A.2.4.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP3 | | | | | | 0 | | PA_RAMP2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP1 | | | | | | 0 | | PA_RAMP0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### A.2.4.5.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP3 | PA_RAMP3<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fourth ramp step. During PA ramp down, the contents of this register are the PA power value during the fourth-to-last ramp step. In both cases, PA_RAMP3 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP2 | PA_RAMP2<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the third ramp step. During PA ramp down, the contents of this register are the PA power value during the third-to-last ramp step. In both cases, PA_RAMP2 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP1 | PA_RAMP1<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the second ramp step. During PA ramp down, the contents of this register are the PA power value during the second-to-last ramp step. In both cases, PA_RAMP1 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP0 | PA_RAMP0<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, when TSM tx_pa_en transitions low to high, and then for the duration of the first ramp step. During PA ramp down, the contents of this register are the PA power value during the final ramp step. In both cases, PA_RAMP0 cannot exceed target power (enforced by PA ramping logic). When PA ramping is enabled, the contents of PA_RAMP0 are also presented to the PA during sequence-idle conditions. |

## A.2.4.6 PA RAMP TABLE 1 (PA_RAMP_TBL1)

### A.2.4.6.1 Offset

| Register | Offset |
|---|---|
| PA_RAMP_TBL1 | 10h |

### A.2.4.6.2 Function

PA Ramp Table 1

### A.2.4.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP7 | | | | | | 0 | | PA_RAMP6 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP5 | | | | | | 0 | | PA_RAMP4 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

### A.2.4.6.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP7 | PA_RAMP7<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the eighth ramp step. During PA ramp down, the contents of this register are the PA power value during the eighth-to-last ramp step. In both cases, PA_RAMP7 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP6 | PA_RAMP6<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the seventh ramp step. During PA ramp down, the contents of this register are the PA power value during the seventh-to-last ramp step. In both cases, PA_RAMP6 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP5 | PA_RAMP5<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the sixth ramp step. During PA ramp down, the contents of this register are the PA power value during the sixth-to-last ramp step. In both cases, PA_RAMP5 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP4 | PA_RAMP4 |

| Field | Function |
|---|---|
| | If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fifth ramp step. During PA ramp down, the contents of this register are the PA power value during the fifth-to-last ramp step. In both cases, PA_RAMP4 cannot exceed target power (enforced by PA ramping logic). |

## A.2.4.7 PA RAMP TABLE 2 (PA_RAMP_TBL2)

### A.2.4.7.1 Offset

| Register | Offset |
|---|---|
| PA_RAMP_TBL2 | 14h |

### A.2.4.7.2 Function

PA Ramp Table 2

### A.2.4.7.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP11 | | | | | | 0 | | PA_RAMP10 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP9 | | | | | | 0 | | PA_RAMP8 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

### A.2.4.7.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP11 | PA_RAMP11<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the twelvth ramp step. During PA ramp down, the |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | contents of this register are the PA power value during the twelvth-to-last ramp step. In both cases, PA_RAMP11 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP10 | PA_RAMP10<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the eleventh ramp step. During PA ramp down, the contents of this register are the PA power value during the eleventh-to-last ramp step. In both cases, PA_RAMP10 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP9 | PA_RAMP9<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the tenth ramp step. During PA ramp down, the contents of this register are the PA power value during the tenth-to-last ramp step. In both cases, PA_RAMP9 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP8 | PA_RAMP8<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the ninth ramp step. During PA ramp down, the contents of this register are the PA power value during the ninth-to-last ramp step. In both cases, PA_RAMP8 cannot exceed target power (enforced by PA ramping logic). |

## A.2.4.8 PA RAMP TABLE 3 (PA_RAMP_TBL3)

### A.2.4.8.1 Offset

| Register | Offset |
|---|---|
| PA_RAMP_TBL3 | 18h |

### A.2.4.8.2 Function

PA Ramp Table 3

## A.2.4.8.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP15 | | | | | | 0 | | PA_RAMP14 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | PA_RAMP13 | | | | | | 0 | | PA_RAMP12 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

## A.2.4.8.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>PA_RAMP15 | PA_RAMP15<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the sixteenth (final) ramp step. During PA ramp down, the contents of this register are the PA power value during the sixteenth-to-last (first) ramp step. In both cases, PA_RAMP15 cannot exceed target power (enforced by PA ramping logic). |
| 23-22<br>— | Reserved |
| 21-16<br>PA_RAMP14 | PA_RAMP14<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fifteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the fifteenth-to-last ramp step. In both cases, PA_RAMP14 cannot exceed target power (enforced by PA ramping logic). |
| 15-14<br>— | Reserved |
| 13-8<br>PA_RAMP13 | PA_RAMP13<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the fourteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the fourteenth-to-last ramp step. In both cases, PA_RAMP13 cannot exceed target power (enforced by PA ramping logic). |
| 7-6<br>— | Reserved |
| 5-0<br>PA_RAMP12 | PA_RAMP12<br>If PA ramping is enabled (TSM_CTRL[PA_RAMP_SEL] > 00), the contents of this register are presented to the PA during PA ramping, for the duration of the thirteenth ramp step. During PA ramp down, the contents of this register are the PA power value during the thirteenth-to-last ramp step. In both cases, PA_RAMP12 cannot exceed target power (enforced by PA ramping logic). |

## A.2.4.9 TSM RECYCLE COUNT (RECYCLE_COUNT)

### A.2.4.9.1 Offset

| Register | Offset |
|---|---|
| RECYCLE_COUNT | 24h |

### A.2.4.9.2 Function

This register contains the TSM Recycle "Jump-to" points for the 3 types of TSM Recycle

### A.2.4.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RECYCLE_COUNT2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RECYCLE_COUNT1 | | | | | | | | RECYCLE_COUNT0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

### A.2.4.9.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>RECYCLE_COUNT2 | TSM RX Recycle Count 2<br>The RECYCLE_COUNT2[7:0] register determines the TSM count value to which the TSM "recycles" when, in LPPS mode, an LPPS power save cycle completes and the receiver is switched back on, necessitating a foreshortened RX warmup. The intention is for this register to be programmed to a TSM count value that is just prior to the assertion of rx_lna_en, but there are no restrictions on programming this register. An LPPS recycle is a command from the LPPS state machine to TSM, requesting a jump from the TSM ON phase back to a programmable point in the WARMUP phase, and resume counting from there. |
| 15-8<br>RECYCLE_COUNT1 | TSM RX Recycle Count 1<br>The RECYCLE_COUNT1[7:0] register determines the TSM count value to which the TSM "recycles" when the 802.15.4 Sequence Manager (ZSM) state "RX_PAN1" is reached and the ZSM asserts tsm_recycle[1] to TSM. The intention is for this register to be programmed to a TSM count value such that |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | the TSM de-asserts, and then re-asserts its "pll_dig_en" output, to effectuate a Dual PAN on-the-fly channel change, but there are no restrictions on programming this register. An RX recycle is a command from ZSM to TSM, requesting a jump from the TSM ON phase back to a programmable point in the WARMUP phase, and resume counting from there. A recycle will result from the expiration of the Dual PAN Dwell Timer, at which point an RF-channel change is required. This necessitates the desassertion and reassertion of pll_dig_en, hence the return to the WARMUP phase at the appropriate point. |
| 7-0<br>RECYCLE_COUNT0 | TSM RX Recycle Count 0<br>The RECYCLE_COUNT0[7:0] register determines the TSM count value to which the TSM "recycles" when the 802.15.4 Sequence Manager (ZSM) state "RX_CYC" is reached and the ZSM asserts tsm_recycle[0] to TSM. This register also determines the TSM count value to which the TSM recycles when the ZSM state RX_CCCA is reached because tsm_recycle[0] is also asserted in this state. This register also determines the TSM count value to which the TSM recycles when theGeneric_FSK Link Layer asserts a tsm_recycle to TSM. The intention is for this register to be programmed to a TSM count value such that the TSM re-asserts its "rx_init" output, but there are no restrictions on programming this register. An RX recycle is a command from any protocol engine to TSM, requesting a jump from the TSM ON phase back to a programmable point in the WARMUP phase, and resume counting from there. A recycle will result from the reception of a packet with bad CRC or one which fails packet-filtering rules, or the end of a CCA operation in Continuous CCA mode which results in a channel indicating "busy". |

## A.2.4.10 TSM FAST WARMUP CONTROL 1 (FAST_CTRL1)

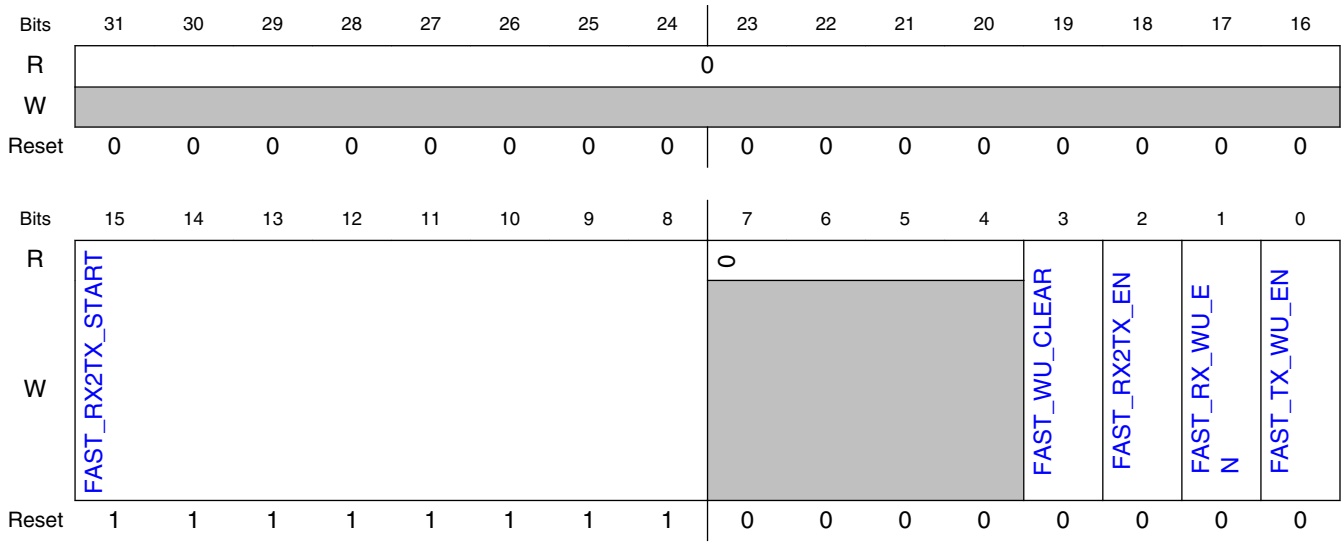
### A.2.4.10.1 Offset

| Register | Offset |
|---|---|
| FAST_CTRL1 | 28h |

### A.2.4.10.2 Function

This register provides enabling and control for Fast TSM Mode

## A.2.4.10.3 Diagram




## A.2.4.10.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>FAST_RX2TX_START | TSM "Jump-to" point for a Fast TSM RX-to-TX Transition.<br>During a Fast RX-to-TX transition (FAST_RX2TX_EN=1) the contents of this register represent the entry point (TSM count) in the TX warmup sequence to which the TSM jumps to when 802.15.4 ZSM asserts fast_rx2tx_wu. The TSM will jump from the RX “ON” phase to the TX “WARMUP” phase, starting at this count. |
| 7-4<br>— | Reserved |
| 3<br>FAST_WU_CLEAR | Fast TSM Warmup Clear State<br>This bit clears the RF channel memory in the PLL Digital Block, forcing the next TSM TX Warmup to be normal (not fast), and the next TSM RX Warmup to be normal (not fast), regardless of whether the RF channel has actually changed or not. This bit is not self-clearing. Write '1' to clear channel memory, then write '0' to proceed with TSM operations. |
| 2<br>FAST_RX2TX_EN | Fast TSM RX-to-TX Transition Enable<br>Intended for 802.15.4 only, this bit enables a special state transition in the 802.15.4 ZSM (Sequence Manager), combined with a TSM jump from the RX "ON" phase to the TX "Warmup" phase, to accelerate the CCA-to-TX transition in the 802.15.4 Sequence T and Sequence TR autosequences<br>0b - Disable Fast RX-to-TX transitions<br>1b - Enable Fast RX-to-TX transitions (if fast_rx2tx_wu is asserted by 802.15.4 ZSM) |
| 1<br>FAST_RX_WU_EN | Fast TSM RX Warmup Enable<br>0b - Fast TSM RX Warmups are disabled<br>1b - Fast TSM RX Warmups are enabled, if the RF channel has not changed since the last RX warmup, and for BLE mode, the RF channel is not an advertising channel. |
| 0 | Fast TSM TX Warmup Enable |

| Field | Function |
|---|---|
| FAST_TX_WU_EN | 0b - Fast TSM TX Warmups are disabled<br>1b - Fast TSM TX Warmups are enabled, if the RF channel has not changed since the last TX warmup, and for BLE mode, the RF channel is not an advertising channel. |

## A.2.4.11 TSM FAST WARMUP CONTROL 2 (FAST_CTRL2)

### A.2.4.11.1 Offset

| Register | Offset |
|---|---|
| FAST_CTRL2 | 2Ch |

### A.2.4.11.2 Function

This register provides configuration for Fast TSM Mode

### A.2.4.11.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FAST_DEST_RX | | | | | | | | FAST_START_RX | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FAST_DEST_TX | | | | | | | | FAST_START_TX | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.11.4 Fields

| Field | Function |
|---|---|
| 31-24<br>FAST_DEST_RX | Fast TSM RX "Jump-to" Point<br>During a Fast RX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, FAST_START_RX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>FAST_START_RX = 10<br>FAST_DEST_RX = 15 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The TSM will count as follows: … 7,8,9,15,16,17 … |
| 23-16<br>FAST_START_RX | Fast TSM RX "Jump-from" Point<br>During a Fast RX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast RX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_RX[7:0], thereby executing the jump. |
| 15-8<br>FAST_DEST_TX | Fast TSM TX "Jump-to" Point<br>During a Fast TX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, FAST_START_TX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>FAST_START_TX = 10<br>FAST_DEST_TX = 15<br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| 7-0<br>FAST_START_TX | Fast TSM TX "Jump-from" Point<br>During a Fast TX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. If Fast TX Warmups are enabled, this count will never be reached; its place in the TSM warmup will be replaced by the contents of FAST_DEST_TX[7:0], thereby executing the jump. |

## A.2.4.12 TSM_TIMING00 (TIMING00)

### A.2.4.12.1 Offset

| Register | Offset |
|---|---|
| TIMING00 | 30h |

### A.2.4.12.2 Function

This register contains the timing values to control the assertion and deassertion times for both TX and RX sequences for the BB_LDO_HF_EN TSM signal or signal group.

## A.2.4.12.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_HF_EN_RX_LO | | | | | | | | BB_LDO_HF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_HF_EN_TX_LO | | | | | | | | BB_LDO_HF_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.4.12.4 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_HF_EN_RX_LO | De-assertion time setting for BB_LDO_HF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_HF_EN_RX_HI | Assertion time setting for BB_LDO_HF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_HF_EN_TX_LO | De-assertion time setting for BB_LDO_HF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_HF_EN_TX_HI | Assertion time setting for BB_LDO_HF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_HF_EN signal or group will transition from LO to HI. |

# A.2.4.13 TSM_TIMING01 (TIMING01)

## A.2.4.13.1 Offset

| Register | Offset |
|---|---|
| TIMING01 | 34h |

### A.2.4.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_ADCDAC_EN_RX_LO | | | | | | | | BB_LDO_ADCDAC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_ADCDAC_EN_TX_LO | | | | | | | | BB_LDO_ADCDAC_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_ADCDAC_EN_RX_LO | De-assertion time setting for BB_LDO_ADCDAC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_ADCDAC_EN_RX_HI | Assertion time setting for BB_LDO_ADCDAC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_ADCDAC_EN_TX_LO | De-assertion time setting for BB_LDO_ADCDAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_ADCDAC_EN_TX_HI | Assertion time setting for BB_LDO_ADCDAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_ADCDAC_EN signal or group will transition from LO to HI. |

## A.2.4.14 TSM_TIMING02 (TIMING02)

### A.2.4.14.1 Offset

| Register | Offset |
|---|---|
| TIMING02 | 38h |

### A.2.4.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_BBA_EN_RX_LO | | | | | | | | BB_LDO_BBA_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.14.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_BBA_EN_RX_LO | De-assertion time setting for BB_LDO_BBA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_BBA_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_BBA_EN_RX_HI | Assertion time setting for BB_LDO_BBA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_BBA_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.15 TSM_TIMING03 (TIMING03)

### A.2.4.15.1 Offset

| Register | Offset |
|---|---|
| TIMING03 | 3Ch |

### A.2.4.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_PD_EN_RX_LO | | | | | | | | BB_LDO_PD_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_PD_EN_TX_LO | | | | | | | | BB_LDO_PD_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.15.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_PD_EN_RX_LO | De-assertion time setting for BB_LDO_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_PD_EN_RX_HI | Assertion time setting for BB_LDO_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_PD_EN_TX_LO | De-assertion time setting for BB_LDO_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_PD_EN_TX_HI | Assertion time setting for BB_LDO_PD_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_PD_EN signal or group will transition from LO to HI. |

## A.2.4.16 TSM_TIMING04 (TIMING04)

### A.2.4.16.1 Offset

| Register | Offset |
|---|---|
| TIMING04 | 40h |

## A.2.4.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_FDBK_EN_RX_LO | | | | | | | | BB_LDO_FDBK_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_FDBK_EN_TX_LO | | | | | | | | BB_LDO_FDBK_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.4.16.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_FDBK_EN_RX_LO | De-assertion time setting for BB_LDO_FDBK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_FDBK_EN_RX_HI | Assertion time setting for BB_LDO_FDBK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_FDBK_EN_TX_LO | De-assertion time setting for BB_LDO_FDBK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_FDBK_EN_TX_HI | Assertion time setting for BB_LDO_FDBK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_EN signal or group will transition from LO to HI. |

# A.2.4.17 TSM_TIMING05 (TIMING05)

## A.2.4.17.1 Offset

| Register | Offset |
|---|---|
| TIMING05 | 44h |

### A.2.4.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VCOLO_EN_RX_LO | | | | | | | | BB_LDO_VCOLO_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VCOLO_EN_TX_LO | | | | | | | | BB_LDO_VCOLO_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.17.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VCOLO_EN_RX_HI | Assertion time setting for BB_LDO_VCOLO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VCOLO_EN_TX_LO | De-assertion time setting for BB_LDO_VCOLO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_EN signal or group will transition from LO to HI. |

## A.2.4.18 TSM_TIMING06 (TIMING06)

### A.2.4.18.1 Offset

| Register | Offset |
|---|---|
| TIMING06 | 48h |

## A.2.4.18.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VTREF_EN_RX_LO | | | | | | | | BB_LDO_VTREF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BB_LDO_VTREF_EN_TX_LO | | | | | | | | BB_LDO_VTREF_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.4.18.3 Fields

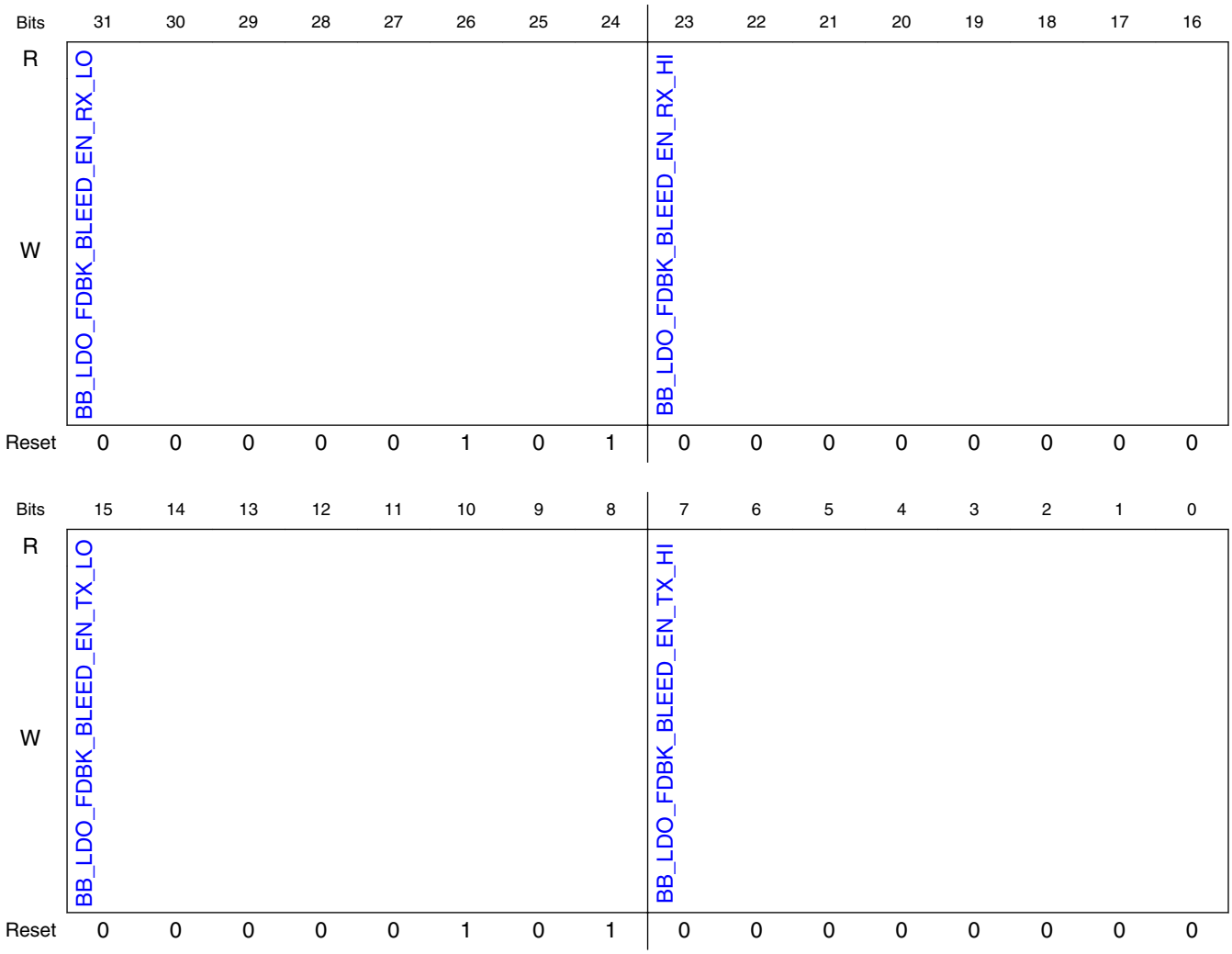
| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VTREF_EN_RX_LO | De-assertion time setting for BB_LDO_VTREF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VTREF_EN_RX_HI | Assertion time setting for BB_LDO_VTREF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VTREF_EN_TX_LO | De-assertion time setting for BB_LDO_VTREF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VTREF_EN_TX_HI | Assertion time setting for BB_LDO_VTREF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VTREF_EN signal or group will transition from LO to HI. |

# A.2.4.19 TSM_TIMING07 (TIMING07)

## A.2.4.19.1 Offset

| Register | Offset |
|---|---|
| TIMING07 | 4Ch |

## A.2.4.19.2 Diagram




## A.2.4.19.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_FDBK_BLEED_EN_RX_LO | De-assertion time setting for BB_LDO_FDBK_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_FDBK_BLEED_EN_RX_HI | Assertion time setting for BB_LDO_FDBK_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for BB_LDO_FDBK_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

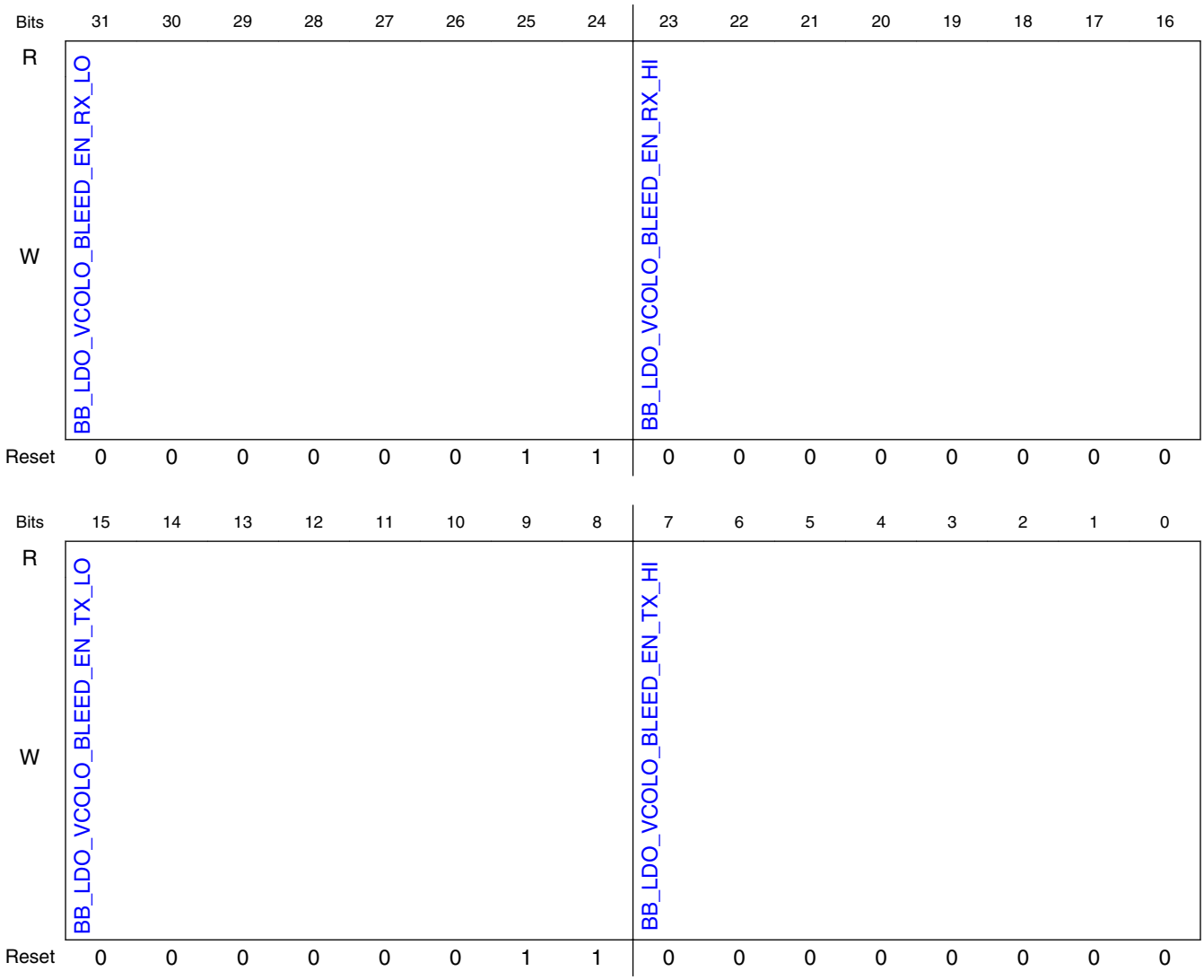
| Field | Function |
|---|---|
| BB_LDO_FDBK_BLEED_EN_TX_LO | |
| 7-0<br>BB_LDO_FDBK_BLEED_EN_TX_HI | Assertion time setting for BB_LDO_FDBK_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_FDBK_BLEED_EN signal or group will transition from LO to HI. |

## A.2.4.20 TSM_TIMING08 (TIMING08)

### A.2.4.20.1 Offset

| Register | Offset |
|---|---|
| TIMING08 | 50h |

## A.2.4.20.2 Diagram




## A.2.4.20.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_BLEED_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_LDO_VCOLO_BLEED_EN_RX_HI | Assertion time setting for BB_LDO_VCOLO_BLEED_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for BB_LDO_VCOLO_BLEED_EN (TX) |

*Table continues on the next page...*

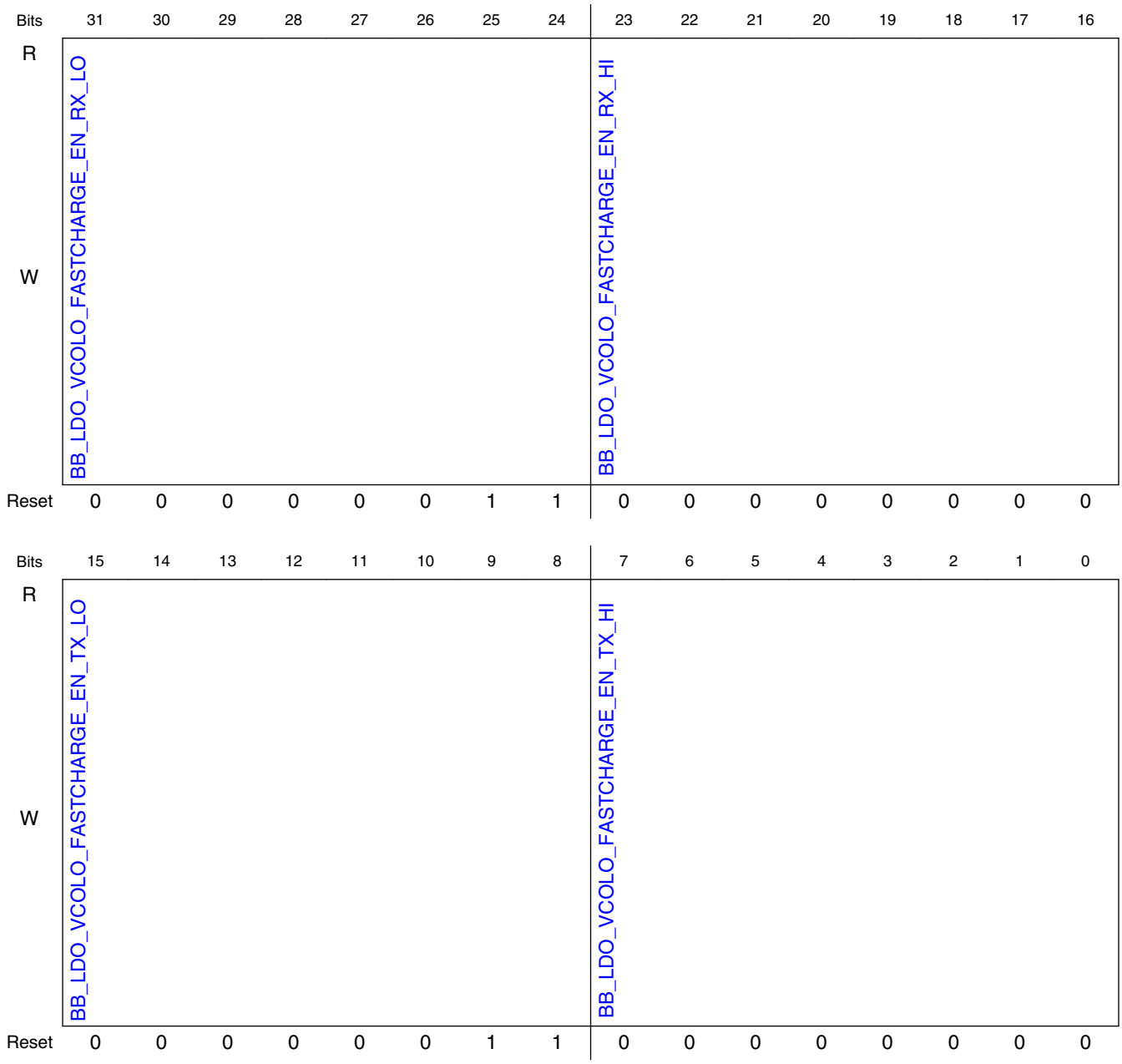
| Field | Function |
|---|---|
| BB_LDO_VCOLO_BLEED_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_BLEED_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_BLEED_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_BLEED_EN signal or group will transition from LO to HI. |

## A.2.4.21 TSM_TIMING09 (TIMING09)

### A.2.4.21.1 Offset

| Register | Offset |
|---|---|
| TIMING09 | 54h |

## A.2.4.21.2 Diagram




## A.2.4.21.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_LDO_VCOLO_FASTCHARGE_EN_RX_LO | De-assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (RX) |

*Table continues on the next page...*

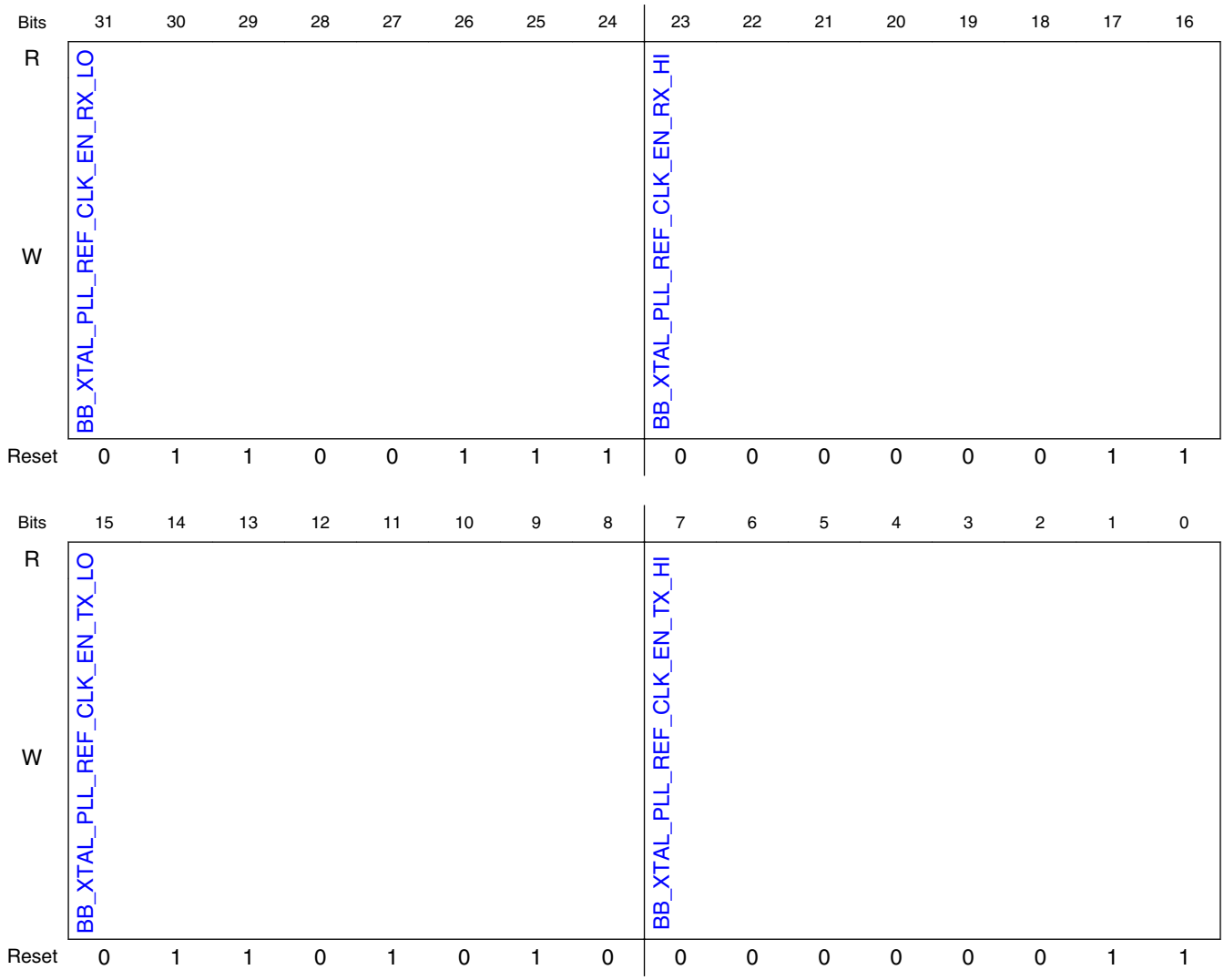
| Field | Function |
|---|---|
| BB_LDO_VCOLO_FASTCHARGE_EN_RX_HI | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from LO to HI. |
| 15-8<br>BB_LDO_VCOLO_FASTCHARGE_EN_TX_LO | De-assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_LDO_VCOLO_FASTCHARGE_EN_TX_HI | Assertion time setting for BB_LDO_VCOLO_FASTCHARGE_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_LDO_VCOLO_FASTCHARGE_EN signal or group will transition from LO to HI. |

## A.2.4.22 TSM_TIMING10 (TIMING10)

### A.2.4.22.1 Offset

| Register | Offset |
|---|---|
| TIMING10 | 58h |

## A.2.4.22.2 Diagram




## A.2.4.22.3 Fields

| Field | Function |
|---|---|
| 31-24<br>BB_XTAL_PLL_REF_CLK_EN_RX_LO | De-assertion time setting for BB_XTAL_PLL_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from HI to LO. |
| 23-16<br>BB_XTAL_PLL_REF_CLK_EN_RX_HI | Assertion time setting for BB_XTAL_PLL_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for BB_XTAL_PLL_REF_CLK_EN (TX) |

*Table continues on the next page...*

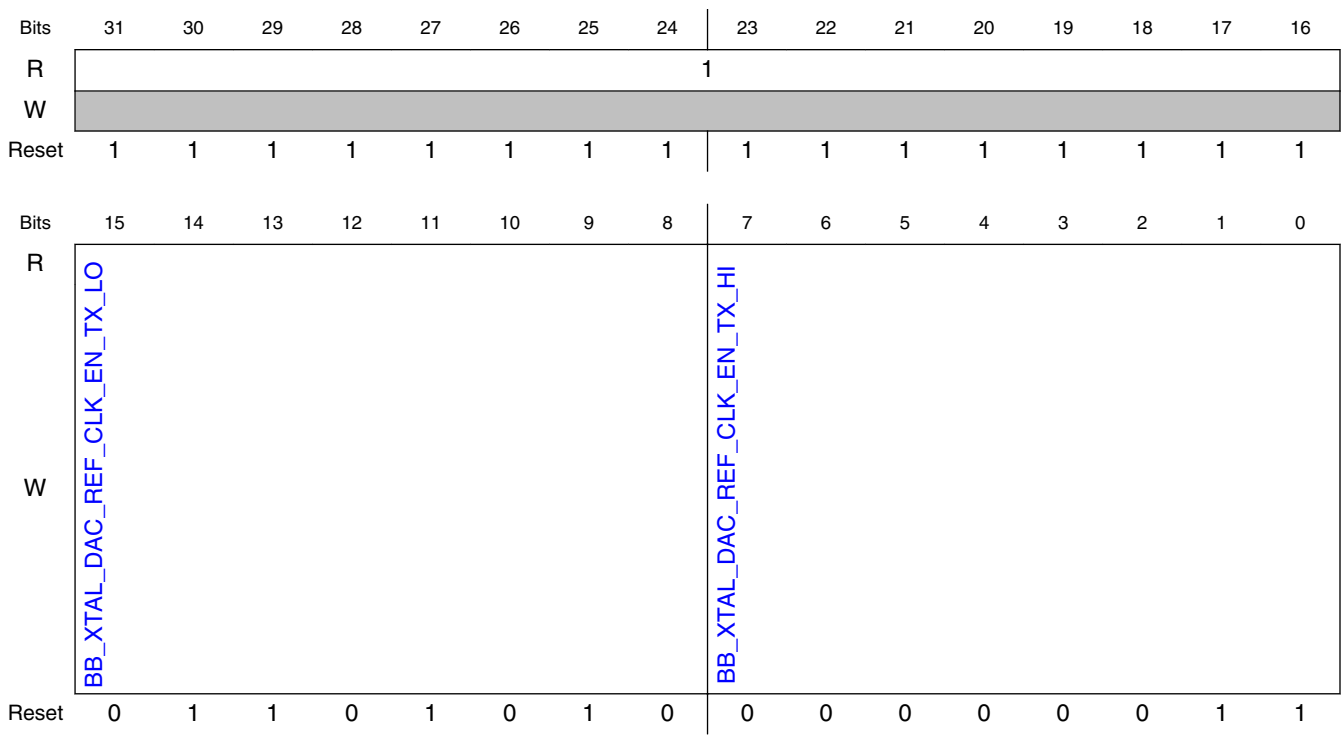
| Field | Function |
|---|---|
| BB_XTAL_PLL_REF_CLK_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_XTAL_PLL_REF_CLK_EN_TX_HI | Assertion time setting for BB_XTAL_PLL_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_PLL_REF_CLK_EN signal or group will transition from LO to HI. |

## A.2.4.23 TSM_TIMING11 (TIMING11)

### A.2.4.23.1 Offset

| Register | Offset |
|---|---|
| TIMING11 | 5Ch |

### A.2.4.23.2 Diagram



### A.2.4.23.3 Fields

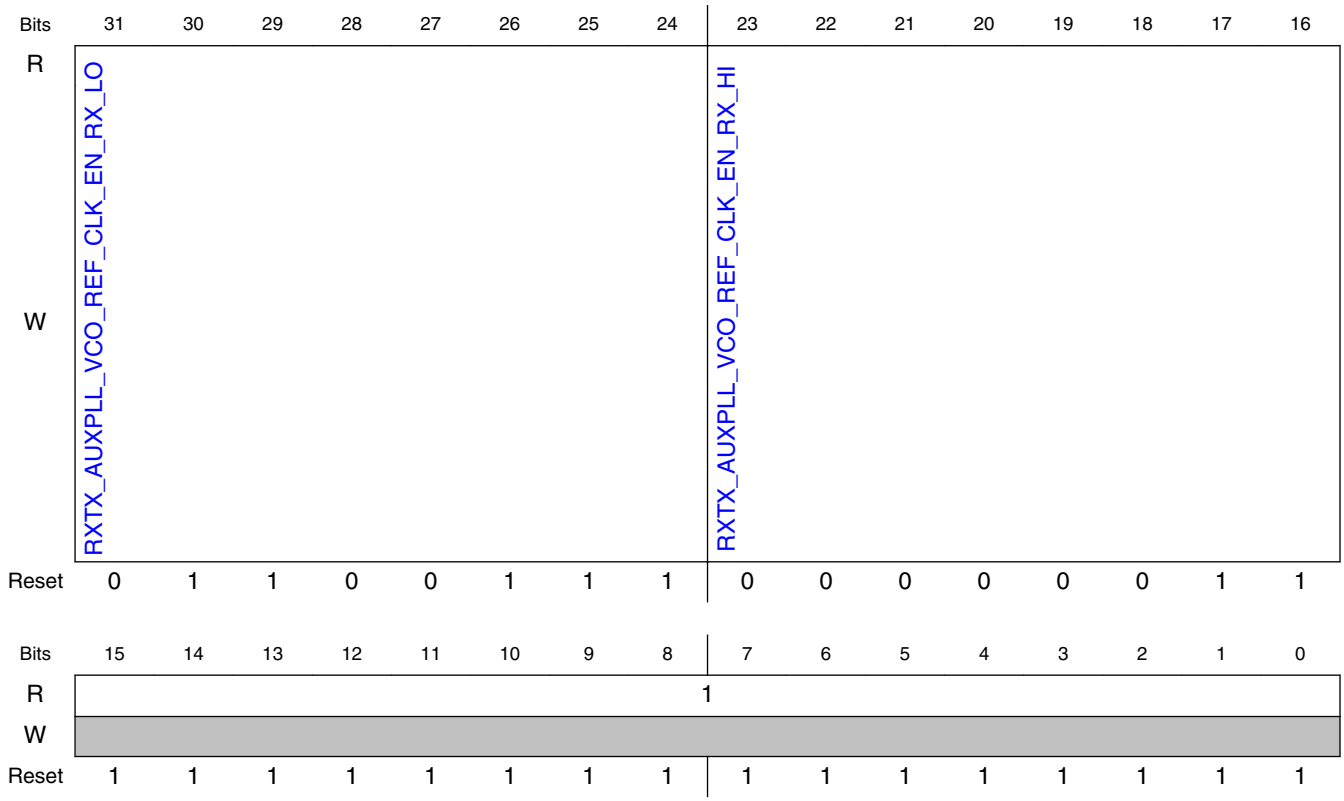
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>BB_XTAL_DAC_REF_CLK_EN_TX_LO | De-assertion time setting for BB_XTAL_DAC_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_DAC_REF_CLK_EN signal or group will transition from HI to LO. |
| 7-0<br>BB_XTAL_DAC_REF_CLK_EN_TX_HI | Assertion time setting for BB_XTAL_DAC_REF_CLK_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the BB_XTAL_DAC_REF_CLK_EN signal or group will transition from LO to HI. |

## A.2.4.24 TSM_TIMING12 (TIMING12)

### A.2.4.24.1 Offset

| Register | Offset |
|---|---|
| TIMING12 | 60h |

## A.2.4.24.2 Diagram




## A.2.4.24.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_VCO_REF_CLK_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_VCO_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_VCO_REF_CLK_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_VCO_REF_CLK_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_VCO_REF_CLK_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_VCO_REF_CLK_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.25 TSM_TIMING13 (TIMING13)

### A.2.4.25.1 Offset

| Register | Offset |
|---|---|
| TIMING13 | 64h |

### A.2.4.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_LOOP_IS_OPEN_RX_LO | | | | | | | | PLL_LOOP_IS_OPEN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_LOOP_IS_OPEN_TX_LO | | | | | | | | PLL_LOOP_IS_OPEN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.25.3 Fields

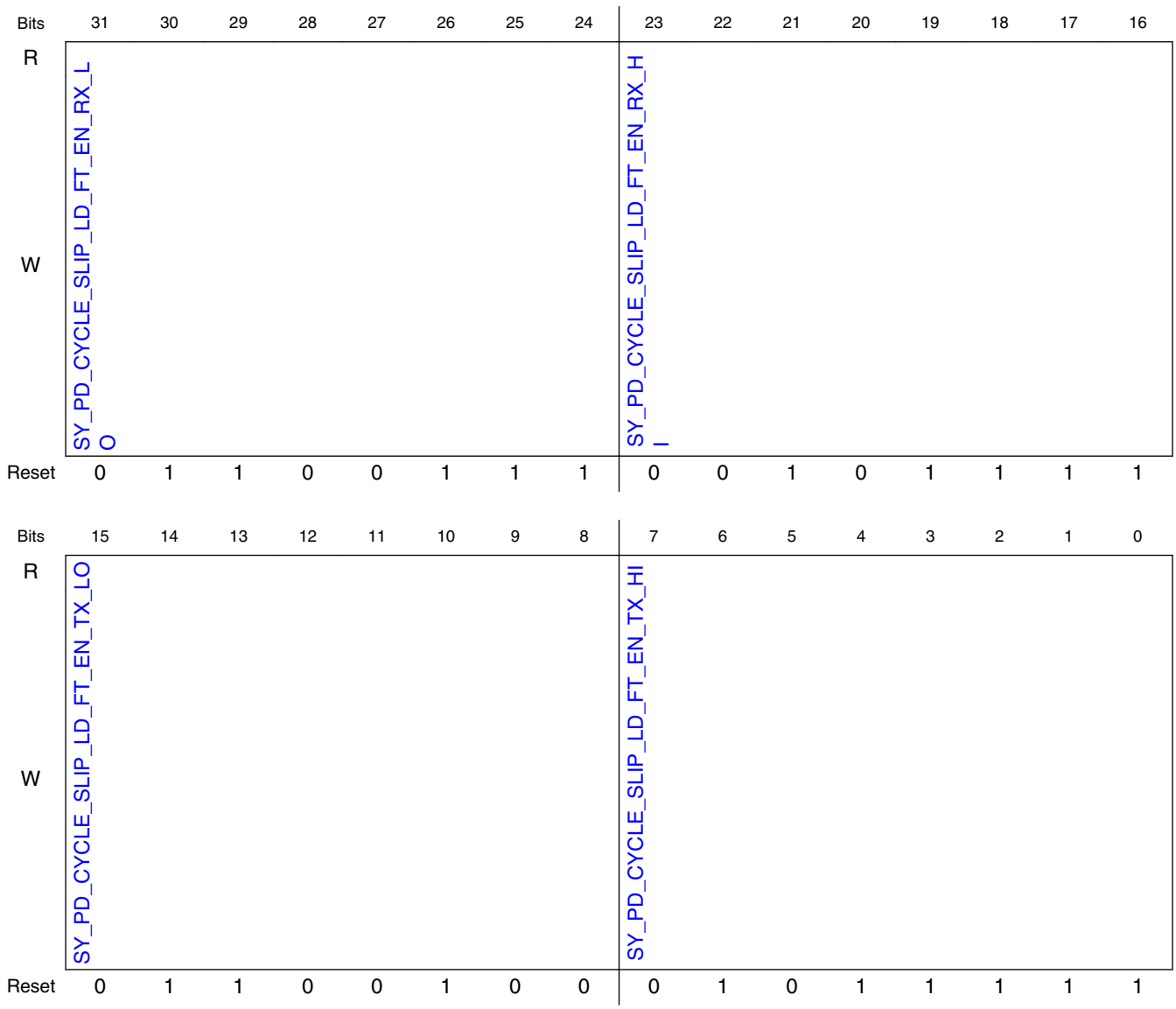
| Field | Function |
|---|---|
| 31-24<br>PLL_LOOP_IS_OPEN_RX_LO | De-assertion time setting for PLL_LOOP_IS_OPEN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from HI to LO. |
| 23-16<br>PLL_LOOP_IS_OPEN_RX_HI | Assertion time setting for PLL_LOOP_IS_OPEN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from LO to HI. |
| 15-8<br>PLL_LOOP_IS_OPEN_TX_LO | De-assertion time setting for PLL_LOOP_IS_OPEN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from HI to LO. |
| 7-0<br>PLL_LOOP_IS_OPEN_TX_HI | Assertion time setting for PLL_LOOP_IS_OPEN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_LOOP_IS_OPEN signal or group will transition from LO to HI. |

## A.2.4.26 TSM_TIMING14 (TIMING14)

## A.2.4.26.1 Offset

| Register | Offset |
|---|---|
| TIMING14 | 68h |

## A.2.4.26.2 Diagram




## A.2.4.26.3 Fields

| Field | Function |
|---|---|
| 31-24 | De-assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (RX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_CYCLE_SLIP_LD_FT_EN_RX_LO | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_PD_CYCLE_SLIP_LD_FT_EN_RX_HI | Assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_PD_CYCLE_SLIP_LD_FT_EN_TX_LO | De-assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_PD_CYCLE_SLIP_LD_FT_EN_TX_HI | Assertion time setting for SY_PD_CYCLE_SLIP_LD_FT_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_CYCLE_SLIP_LD_FT_EN signal or group will transition from LO to HI. |

# A.2.4.27 TSM_TIMING15 (TIMING15)

## A.2.4.27.1 Offset

| Register | Offset |
|---|---|
| TIMING15 | 6Ch |

## A.2.4.27.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_VCO_EN_RX_LO | | | | | | | | SY_VCO_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_VCO_EN_TX_LO | | | | | | | | SY_VCO_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### A.2.4.27.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_VCO_EN_RX_LO | De-assertion time setting for SY_VCO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_VCO_EN_RX_HI | Assertion time setting for SY_VCO_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_VCO_EN_TX_LO | De-assertion time setting for SY_VCO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_VCO_EN_TX_HI | Assertion time setting for SY_VCO_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_VCO_EN signal or group will transition from LO to HI. |

## A.2.4.28 TSM_TIMING16 (TIMING16)

### A.2.4.28.1 Offset

| Register | Offset |
|---|---|
| TIMING16 | 70h |

### A.2.4.28.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_RX_BUF_EN_RX_LO | | | | | | | | SY_LO_RX_BUF_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.28.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_LO_RX_BUF_EN_RX_LO | De-assertion time setting for SY_LO_RX_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_BUF_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_LO_RX_BUF_EN_RX_HI | Assertion time setting for SY_LO_RX_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_BUF_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.29 TSM_TIMING17 (TIMING17)

### A.2.4.29.1 Offset

| Register | Offset |
|---|---|
| TIMING17 | 74h |

### A.2.4.29.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_TX_BUF_EN_TX_LO | | | | | | | | SY_LO_TX_BUF_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

### A.2.4.29.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8 | De-assertion time setting for SY_LO_TX_BUF_EN (TX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_LO_TX_BUF_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_BUF_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_LO_TX_BUF_EN_TX_HI | Assertion time setting for SY_LO_TX_BUF_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_BUF_EN signal or group will transition from LO to HI. |

# A.2.4.30 TSM_TIMING18 (TIMING18)

## A.2.4.30.1 Offset

| Register | Offset |
|---|---|
| TIMING18 | 78h |

## A.2.4.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_EN_RX_LO | | | | | | | | SY_DIVN_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_EN_TX_LO | | | | | | | | SY_DIVN_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

## A.2.4.30.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_DIVN_EN_RX_LO | De-assertion time setting for SY_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_DIVN_EN_RX_HI | Assertion time setting for SY_DIVN_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_DIVN_EN_TX_LO | De-assertion time setting for SY_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

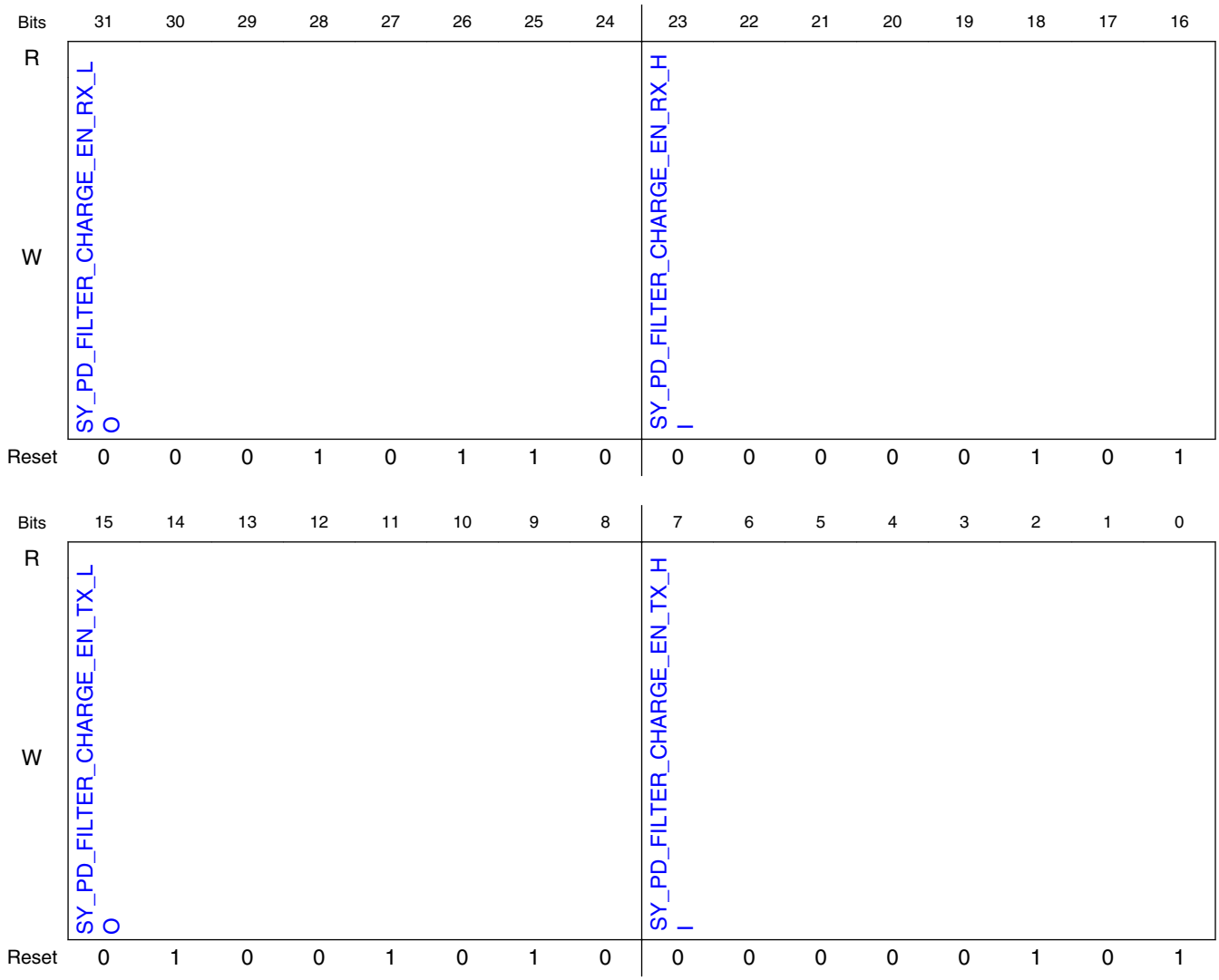
| Field | Function |
|---|---|
| 7-0<br>SY_DIVN_EN_TX_HI | Assertion time setting for SY_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_EN signal or group will transition from LO to HI. |

## A.2.4.31 TSM_TIMING19 (TIMING19)

### A.2.4.31.1 Offset

| Register | Offset |
|---|---|
| TIMING19 | 7Ch |

## A.2.4.31.2 Diagram




## A.2.4.31.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_PD_FILTER_CHARGE_EN_RX_LO | De-assertion time setting for SY_PD_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_PD_FILTER_CHARGE_EN_RX_HI | Assertion time setting for SY_PD_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for SY_PD_FILTER_CHARGE_EN (TX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_FILTER_CHARGE_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from HI to LO. |
| 7-0 | Assertion time setting for SY_PD_FILTER_CHARGE_EN (TX) |
| SY_PD_FILTER_CHARGE_EN_TX_HI | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_FILTER_CHARGE_EN signal or group will transition from LO to HI. |

## A.2.4.32 TSM_TIMING20 (TIMING20)

### A.2.4.32.1 Offset

| Register | Offset |
|---|---|
| TIMING20 | 80h |

### A.2.4.32.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_PD_EN_RX_LO | | | | | | | | SY_PD_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_PD_EN_TX_LO | | | | | | | | SY_PD_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

### A.2.4.32.3 Fields

| Field | Function |
|---|---|
| 31-24 | De-assertion time setting for SY_PD_EN (RX) |
| SY_PD_EN_RX_LO | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for SY_PD_EN (RX) |
| SY_PD_EN_RX_HI | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for SY_PD_EN (TX) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from HI to LO. |
| 7-0 | Assertion time setting for SY_PD_EN (TX) |
| SY_PD_EN_TX_HI | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_PD_EN signal or group will transition from LO to HI. |

# A.2.4.33 TSM_TIMING21 (TIMING21)

## A.2.4.33.1 Offset

| Register | Offset |
|---|---|
| TIMING21 | 84h |

## A.2.4.33.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_DIVN_EN_RX_LO | | | | | | | | SY_LO_DIVN_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_DIVN_EN_TX_LO | | | | | | | | SY_LO_DIVN_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

## A.2.4.33.3 Fields

| Field | Function |
|---|---|
| 31-24 | De-assertion time setting for SY_LO_DIVN_EN (RX) |
| SY_LO_DIVN_EN_RX_LO | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from HI to LO. |
| 23-16 | Assertion time setting for SY_LO_DIVN_EN (RX) |
| SY_LO_DIVN_EN_RX_HI | This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from LO to HI. |
| 15-8 | De-assertion time setting for SY_LO_DIVN_EN (TX) |
| SY_LO_DIVN_EN_TX_LO | This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 7-0<br>SY_LO_DIVN_EN_TX_HI | Assertion time setting for SY_LO_DIVN_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_DIVN_EN signal or group will transition from LO to HI. |

## A.2.4.34 TSM_TIMING22 (TIMING22)

### A.2.4.34.1 Offset

| Register | Offset |
|---|---|
| TIMING22 | 88h |

### A.2.4.34.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_LO_RX_EN_RX_LO | | | | | | | | SY_LO_RX_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.34.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_LO_RX_EN_RX_LO | De-assertion time setting for SY_LO_RX_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_LO_RX_EN_RX_HI | Assertion time setting for SY_LO_RX_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_LO_RX_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.35 TSM_TIMING23 (TIMING23)

## A.2.4.35.1 Offset

| Register | Offset |
|---|---|
| TIMING23 | 8Ch |

## A.2.4.35.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SY_LO_TX_EN_TX_LO | | | | | | | | SY_LO_TX_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

## A.2.4.35.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>SY_LO_TX_EN_TX_LO | De-assertion time setting for SY_LO_TX_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_LO_TX_EN_TX_HI | Assertion time setting for SY_LO_TX_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_LO_TX_EN signal or group will transition from LO to HI. |

# A.2.4.36 TSM_TIMING24 (TIMING24)

### A.2.4.36.1 Offset

| Register | Offset |
|---|---|
| TIMING24 | 90h |

### A.2.4.36.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_CAL_EN_RX_LO | | | | | | | | SY_DIVN_CAL_EN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SY_DIVN_CAL_EN_TX_LO | | | | | | | | SY_DIVN_CAL_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.36.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SY_DIVN_CAL_EN_RX_LO | De-assertion time setting for SY_DIVN_CAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from HI to LO. |
| 23-16<br>SY_DIVN_CAL_EN_RX_HI | Assertion time setting for SY_DIVN_CAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from LO to HI. |
| 15-8<br>SY_DIVN_CAL_EN_TX_LO | De-assertion time setting for SY_DIVN_CAL_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from HI to LO. |
| 7-0<br>SY_DIVN_CAL_EN_TX_HI | Assertion time setting for SY_DIVN_CAL_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SY_DIVN_CAL_EN signal or group will transition from LO to HI. |

## A.2.4.37 TSM_TIMING25 (TIMING25)

### A.2.4.37.1 Offset

| Register | Offset |
|---|---|
| TIMING25 | 94h |

### A.2.4.37.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_LNA_MIXER_EN_RX_LO | | | | | | | | RX_LNA_MIXER_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.37.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_LNA_MIXER_EN_RX_LO | De-assertion time setting for RX_LNA_MIXER_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_LNA_MIXER_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_LNA_MIXER_EN_RX_HI | Assertion time setting for RX_LNA_MIXER_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_LNA_MIXER_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.38 TSM_TIMING26 (TIMING26)

### A.2.4.38.1 Offset

| Register | Offset |
|---|---|
| TIMING26 | 98h |

### A.2.4.38.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_PA_EN_TX_LO | | | | | | | | TX_PA_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

### A.2.4.38.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_PA_EN_TX_LO | De-assertion time setting for TX_PA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_PA_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_PA_EN_TX_HI | Assertion time setting for TX_PA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_PA_EN signal or group will transition from LO to HI. |

## A.2.4.39 TSM_TIMING27 (TIMING27)

### A.2.4.39.1 Offset

| Register | Offset |
|---|---|
| TIMING27 | 9Ch |

## A.2.4.39.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_ADC_I_Q_EN_RX_LO | | | | | | | | RX_ADC_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.39.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_ADC_I_Q_EN_RX_LO | De-assertion time setting for RX_ADC_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_ADC_I_Q_EN_RX_HI | Assertion time setting for RX_ADC_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.40 TSM_TIMING28 (TIMING28)

## A.2.4.40.1 Offset

| Register | Offset |
|---|---|
| TIMING28 | A0h |

### A.2.4.40.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_ADC_RESET_EN_RX_LO | | | | | | | | RX_ADC_RESET_EN_RX_HI | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.40.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_ADC_RESET_EN_RX_LO | De-assertion time setting for RX_ADC_RESET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_RESET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_ADC_RESET_EN_RX_HI | Assertion time setting for RX_ADC_RESET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_ADC_RESET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.41 TSM_TIMING29 (TIMING29)

### A.2.4.41.1 Offset

| Register | Offset |
|---|---|
| TIMING29 | A4h |

## A.2.4.41.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_I_Q_EN_RX_LO | | | | | | | | RX_BBA_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.41.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_I_Q_EN_RX_LO | De-assertion time setting for RX_BBA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_I_Q_EN_RX_HI | Assertion time setting for RX_BBA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.42 TSM_TIMING30 (TIMING30)

## A.2.4.42.1 Offset

| Register | Offset |
|---|---|
| TIMING30 | A8h |

### A.2.4.42.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_PDET_EN_RX_LO | | | | | | | | RX_BBA_PDET_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.42.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_PDET_EN_RX_LO | De-assertion time setting for RX_BBA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_PDET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_PDET_EN_RX_HI | Assertion time setting for RX_BBA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_PDET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.43 TSM_TIMING31 (TIMING31)

### A.2.4.43.1 Offset

| Register | Offset |
|---|---|
| TIMING31 | ACh |

## A.2.4.43.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_BBA_TZA_DCOC_EN_RX_LO | | | | | | | | RX_BBA_TZA_DCOC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.43.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_BBA_TZA_DCOC_EN_RX_LO | De-assertion time setting for RX_BBA_TZA_DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_TZA_DCOC_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_BBA_TZA_DCOC_EN_RX_HI | Assertion time setting for RX_BBA_TZA_DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_BBA_TZA_DCOC_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.44 TSM_TIMING32 (TIMING32)

## A.2.4.44.1 Offset

| Register | Offset |
|---|---|
| TIMING32 | B0h |

### A.2.4.44.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_TZA_I_Q_EN_RX_LO | | | | | | | | RX_TZA_I_Q_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.44.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_TZA_I_Q_EN_RX_LO | De-assertion time setting for RX_TZA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_I_Q_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_TZA_I_Q_EN_RX_HI | Assertion time setting for RX_TZA_I_Q_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_I_Q_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.45 TSM_TIMING33 (TIMING33)

### A.2.4.45.1 Offset

| Register | Offset |
|---|---|
| TIMING33 | B4h |

## A.2.4.45.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_TZA_PDET_EN_RX_LO | | | | | | | | RX_TZA_PDET_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.45.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_TZA_PDET_EN_RX_LO | De-assertion time setting for RX_TZA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_PDET_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_TZA_PDET_EN_RX_HI | Assertion time setting for RX_TZA_PDET_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_TZA_PDET_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.46 TSM_TIMING34 (TIMING34)

## A.2.4.46.1 Offset

| Register | Offset |
|---|---|
| TIMING34 | B8h |

## A.2.4.46.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_DIG_EN_RX_LO | | | | | | | | PLL_DIG_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PLL_DIG_EN_TX_LO | | | | | | | | PLL_DIG_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

## A.2.4.46.3 Fields

| Field | Function |
|---|---|
| 31-24<br>PLL_DIG_EN_RX_LO | De-assertion time setting for PLL_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from HI to LO. |
| 23-16<br>PLL_DIG_EN_RX_HI | Assertion time setting for PLL_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from LO to HI. |
| 15-8<br>PLL_DIG_EN_TX_LO | De-assertion time setting for PLL_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from HI to LO. |
| 7-0<br>PLL_DIG_EN_TX_HI | Assertion time setting for PLL_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the PLL_DIG_EN signal or group will transition from LO to HI. |

# A.2.4.47 TSM_TIMING35 (TIMING35)

## A.2.4.47.1 Offset

| Register | Offset |
|---|---|
| TIMING35 | BCh |

## A.2.4.47.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_DIG_EN_TX_LO | | | | | | | | TX_DIG_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

## A.2.4.47.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_DIG_EN_TX_LO | De-assertion time setting for TX_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_DIG_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_DIG_EN_TX_HI | Assertion time setting for TX_DIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_DIG_EN signal or group will transition from LO to HI. |

# A.2.4.48 TSM_TIMING36 (TIMING36)

## A.2.4.48.1 Offset

| Register | Offset |
|---|---|
| TIMING36 | C0h |

### A.2.4.48.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_DIG_EN_RX_LO | | | | | | | | RX_DIG_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.48.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_DIG_EN_RX_LO | De-assertion time setting for RX_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_DIG_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_DIG_EN_RX_HI | Assertion time setting for RX_DIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_DIG_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.49 TSM_TIMING37 (TIMING37)

### A.2.4.49.1 Offset

| Register | Offset |
|---|---|
| TIMING37 | C4h |

## A.2.4.49.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_INIT_RX_LO | | | | | | | | RX_INIT_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.49.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_INIT_RX_LO | De-assertion time setting for RX_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_INIT signal or group will transition from HI to LO. |
| 23-16<br>RX_INIT_RX_HI | Assertion time setting for RX_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_INIT signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.50 TSM_TIMING38 (TIMING38)

## A.2.4.50.1 Offset

| Register | Offset |
|---|---|
| TIMING38 | C8h |

### A.2.4.50.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SIGMA_DELTA_EN_RX_LO | | | | | | | | SIGMA_DELTA_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SIGMA_DELTA_EN_TX_LO | | | | | | | | SIGMA_DELTA_EN_TX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.4.50.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SIGMA_DELTA_EN_RX_LO | De-assertion time setting for SIGMA_DELTA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from HI to LO. |
| 23-16<br>SIGMA_DELTA_EN_RX_HI | Assertion time setting for SIGMA_DELTA_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from LO to HI. |
| 15-8<br>SIGMA_DELTA_EN_TX_LO | De-assertion time setting for SIGMA_DELTA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from HI to LO. |
| 7-0<br>SIGMA_DELTA_EN_TX_HI | Assertion time setting for SIGMA_DELTA_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SIGMA_DELTA_EN signal or group will transition from LO to HI. |

## A.2.4.51 TSM_TIMING39 (TIMING39)

### A.2.4.51.1 Offset

| Register | Offset |
|---|---|
| TIMING39 | CCh |

### A.2.4.51.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_PHY_EN_RX_LO | | | | | | | | RX_PHY_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.51.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RX_PHY_EN_RX_LO | De-assertion time setting for RX_PHY_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_PHY_EN signal or group will transition from HI to LO. |
| 23-16<br>RX_PHY_EN_RX_HI | Assertion time setting for RX_PHY_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RX_PHY_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.52 TSM_TIMING40 (TIMING40)

### A.2.4.52.1 Offset

| Register | Offset |
|---|---|
| TIMING40 | D0h |

### A.2.4.52.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCOC_EN_RX_LO | | | | | | | | DCOC_EN_RX_HI | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.52.3 Fields

| Field | Function |
|---|---|
| 31-24<br>DCOC_EN_RX_LO | De-assertion time setting for DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_EN signal or group will transition from HI to LO. |
| 23-16<br>DCOC_EN_RX_HI | Assertion time setting for DCOC_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.53 TSM_TIMING41 (TIMING41)

### A.2.4.53.1 Offset

| Register | Offset |
|---|---|
| TIMING41 | D4h |

## A.2.4.53.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCOC_INIT_RX_LO | | | | | | | | DCOC_INIT_RX_HI | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.53.3 Fields

| Field | Function |
|---|---|
| 31-24<br>DCOC_INIT_RX_LO | De-assertion time setting for DCOC_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_INIT signal or group will transition from HI to LO. |
| 23-16<br>DCOC_INIT_RX_HI | Assertion time setting for DCOC_INIT (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the DCOC_INIT signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.54 TSM_TIMING42 (TIMING42)

## A.2.4.54.1 Offset

| Register | Offset |
|---|---|
| TIMING42 | D8h |

### A.2.4.54.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAR_ADC_TRIG_EN_RX_LO | | | | | | | | SAR_ADC_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAR_ADC_TRIG_EN_TX_LO | | | | | | | | SAR_ADC_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.54.3 Fields

| Field | Function |
|---|---|
| 31-24<br>SAR_ADC_TRIG_EN_RX_LO | De-assertion time setting for SAR_ADC_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>SAR_ADC_TRIG_EN_RX_HI | Assertion time setting for SAR_ADC_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>SAR_ADC_TRIG_EN_TX_LO | De-assertion time setting for SAR_ADC_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>SAR_ADC_TRIG_EN_TX_HI | Assertion time setting for SAR_ADC_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the SAR_ADC_TRIG_EN signal or group will transition from LO to HI. |

## A.2.4.55 TSM_TIMING43 (TIMING43)

### A.2.4.55.1 Offset

| Register | Offset |
|---|---|
| TIMING43 | DCh |

## A.2.4.55.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE0_EN_RX_LO | | | | | | | | TSM_SPARE0_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE0_EN_TX_LO | | | | | | | | TSM_SPARE0_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.55.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE0_EN_RX_LO | De-assertion time setting for TSM_SPARE0_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE0_EN_RX_HI | Assertion time setting for TSM_SPARE0_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE0_EN_TX_LO | De-assertion time setting for TSM_SPARE0_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE0_EN_TX_HI | Assertion time setting for TSM_SPARE0_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE0_EN signal or group will transition from LO to HI. |

# A.2.4.56 TSM_TIMING44 (TIMING44)

## A.2.4.56.1 Offset

| Register | Offset |
|---|---|
| TIMING44 | E0h |

## A.2.4.56.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE1_EN_RX_LO | | | | | | | | TSM_SPARE1_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE1_EN_TX_LO | | | | | | | | TSM_SPARE1_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.56.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE1_EN_RX_LO | De-assertion time setting for TSM_SPARE1_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE1_EN_RX_HI | Assertion time setting for TSM_SPARE1_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE1_EN_TX_LO | De-assertion time setting for TSM_SPARE1_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE1_EN_TX_HI | Assertion time setting for TSM_SPARE1_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE1_EN signal or group will transition from LO to HI. |

# A.2.4.57 TSM_TIMING45 (TIMING45)

## A.2.4.57.1 Offset

| Register | Offset |
|---|---|
| TIMING45 | E4h |

## A.2.4.57.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE2_EN_RX_LO | | | | | | | | TSM_SPARE2_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE2_EN_TX_LO | | | | | | | | TSM_SPARE2_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.57.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE2_EN_RX_LO | De-assertion time setting for TSM_SPARE2_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE2_EN_RX_HI | Assertion time setting for TSM_SPARE2_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE2_EN_TX_LO | De-assertion time setting for TSM_SPARE2_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE2_EN_TX_HI | Assertion time setting for TSM_SPARE2_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE2_EN signal or group will transition from LO to HI. |

# A.2.4.58 TSM_TIMING46 (TIMING46)

## A.2.4.58.1 Offset

| Register | Offset |
|---|---|
| TIMING46 | E8h |

### A.2.4.58.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE3_EN_RX_LO | | | | | | | | TSM_SPARE3_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSM_SPARE3_EN_TX_LO | | | | | | | | TSM_SPARE3_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.58.3 Fields

| Field | Function |
|---|---|
| 31-24<br>TSM_SPARE3_EN_RX_LO | De-assertion time setting for TSM_SPARE3_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from HI to LO. |
| 23-16<br>TSM_SPARE3_EN_RX_HI | Assertion time setting for TSM_SPARE3_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from LO to HI. |
| 15-8<br>TSM_SPARE3_EN_TX_LO | De-assertion time setting for TSM_SPARE3_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from HI to LO. |
| 7-0<br>TSM_SPARE3_EN_TX_HI | Assertion time setting for TSM_SPARE3_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TSM_SPARE3_EN signal or group will transition from LO to HI. |

## A.2.4.59 TSM_TIMING47 (TIMING47)

### A.2.4.59.1 Offset

| Register | Offset |
|---|---|
| TIMING47 | ECh |

## A.2.4.59.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO0_TRIG_EN_RX_LO | | | | | | | | GPIO0_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO0_TRIG_EN_TX_LO | | | | | | | | GPIO0_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.59.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO0_TRIG_EN_RX_LO | De-assertion time setting for GPIO0_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO0_TRIG_EN_RX_HI | Assertion time setting for GPIO0_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO0_TRIG_EN_TX_LO | De-assertion time setting for GPIO0_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO0_TRIG_EN_TX_HI | Assertion time setting for GPIO0_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO0_TRIG_EN signal or group will transition from LO to HI. |

# A.2.4.60 TSM_TIMING48 (TIMING48)

## A.2.4.60.1 Offset

| Register | Offset |
|---|---|
| TIMING48 | F0h |

### A.2.4.60.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO1_TRIG_EN_RX_LO | | | | | | | | GPIO1_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO1_TRIG_EN_TX_LO | | | | | | | | GPIO1_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.60.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO1_TRIG_EN_RX_LO | De-assertion time setting for GPIO1_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO1_TRIG_EN_RX_HI | Assertion time setting for GPIO1_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO1_TRIG_EN_TX_LO | De-assertion time setting for GPIO1_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO1_TRIG_EN_TX_HI | Assertion time setting for GPIO1_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO1_TRIG_EN signal or group will transition from LO to HI. |

## A.2.4.61 TSM_TIMING49 (TIMING49)

### A.2.4.61.1 Offset

| Register | Offset |
|---|---|
| TIMING49 | F4h |

## A.2.4.61.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO2_TRIG_EN_RX_LO | | | | | | | | GPIO2_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO2_TRIG_EN_TX_LO | | | | | | | | GPIO2_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.2.4.61.3 Fields

| Field | Function |
|---|---|
| 31-24<br>GPIO2_TRIG_EN_RX_LO | De-assertion time setting for GPIO2_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO2_TRIG_EN_RX_HI | Assertion time setting for GPIO2_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO2_TRIG_EN_TX_LO | De-assertion time setting for GPIO2_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO2_TRIG_EN_TX_HI | Assertion time setting for GPIO2_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO2_TRIG_EN signal or group will transition from LO to HI. |

# A.2.4.62 TSM_TIMING50 (TIMING50)

## A.2.4.62.1 Offset

| Register | Offset |
|---|---|
| TIMING50 | F8h |

### A.2.4.62.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO3_TRIG_EN_RX_LO | | | | | | | | GPIO3_TRIG_EN_RX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GPIO3_TRIG_EN_TX_LO | | | | | | | | GPIO3_TRIG_EN_TX_HI | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.2.4.62.3 Fields

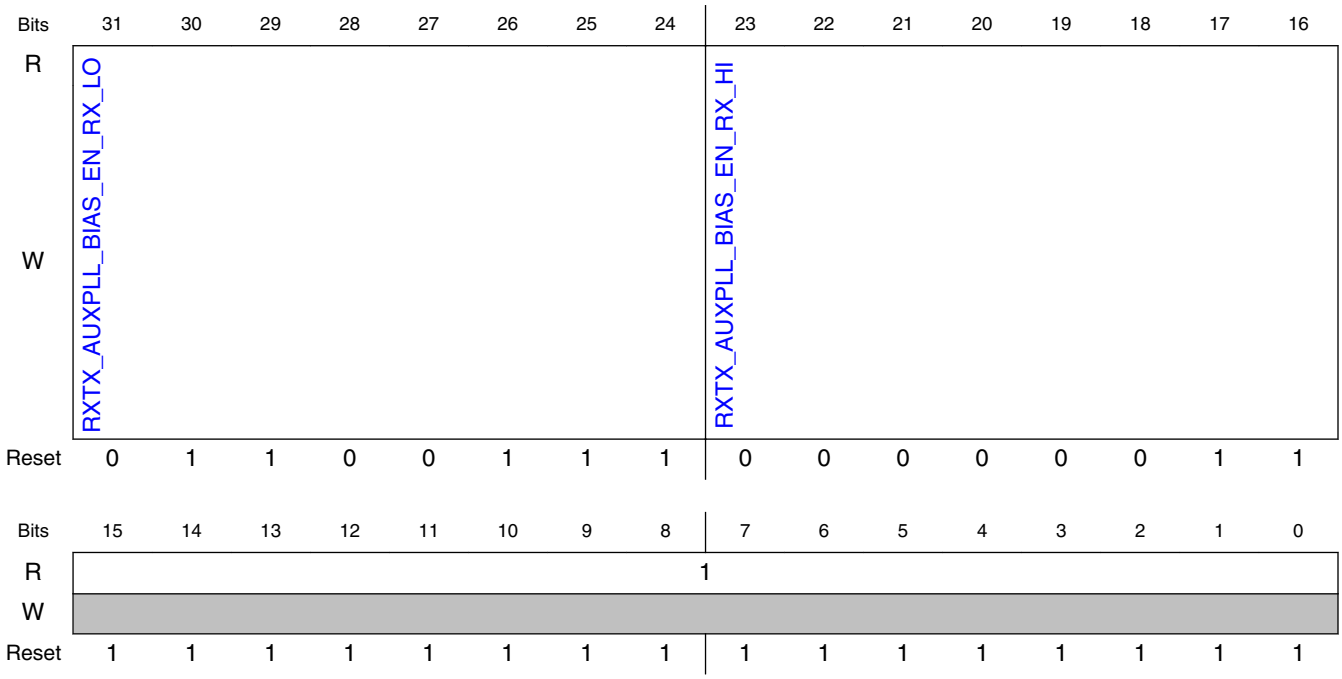
| Field | Function |
|---|---|
| 31-24<br>GPIO3_TRIG_EN_RX_LO | De-assertion time setting for GPIO3_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from HI to LO. |
| 23-16<br>GPIO3_TRIG_EN_RX_HI | Assertion time setting for GPIO3_TRIG_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from LO to HI. |
| 15-8<br>GPIO3_TRIG_EN_TX_LO | De-assertion time setting for GPIO3_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from HI to LO. |
| 7-0<br>GPIO3_TRIG_EN_TX_HI | Assertion time setting for GPIO3_TRIG_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the GPIO3_TRIG_EN signal or group will transition from LO to HI. |

## A.2.4.63 TSM_TIMING51 (TIMING51)

### A.2.4.63.1 Offset

| Register | Offset |
|---|---|
| TIMING51 | FCh |

## A.2.4.63.2 Diagram




## A.2.4.63.3 Fields

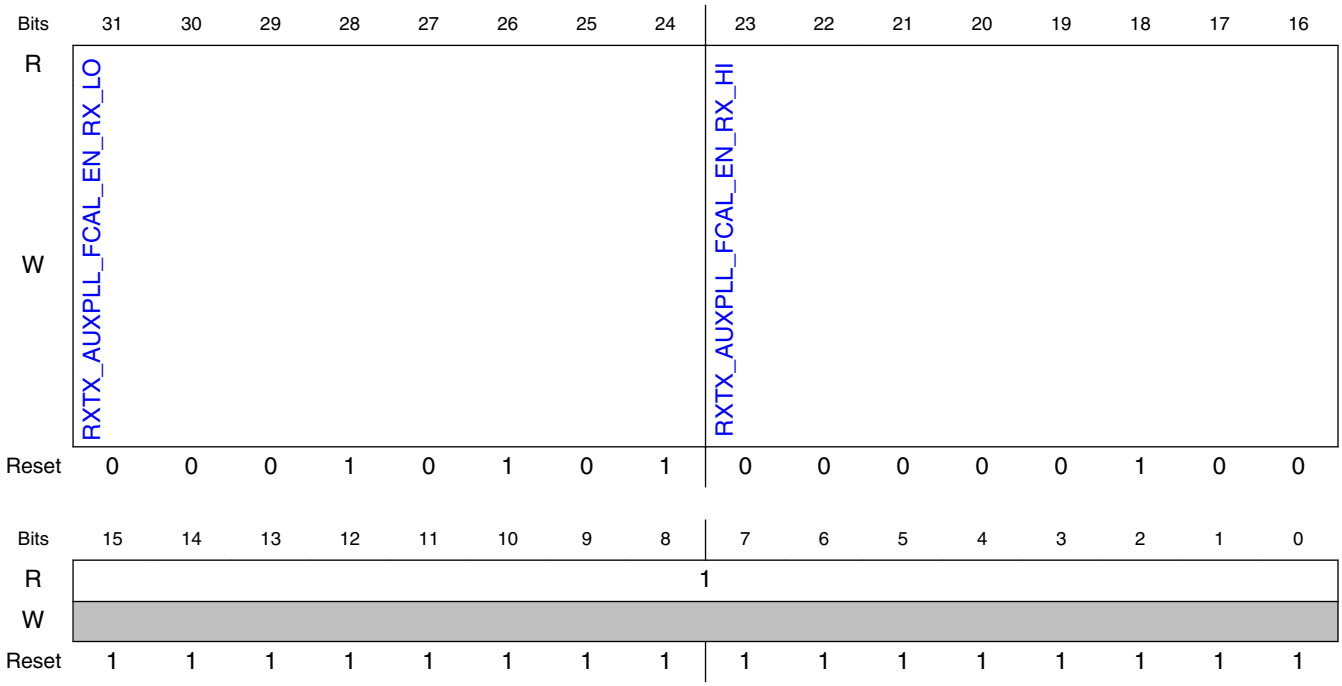
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_BIAS_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_BIAS_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_BIAS_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_BIAS_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_BIAS_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_BIAS_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.64 TSM_TIMING52 (TIMING52)

## A.2.4.64.1 Offset

| Register | Offset |
|---|---|
| TIMING52 | 100h |

### A.2.4.64.2 Diagram




### A.2.4.64.3 Fields

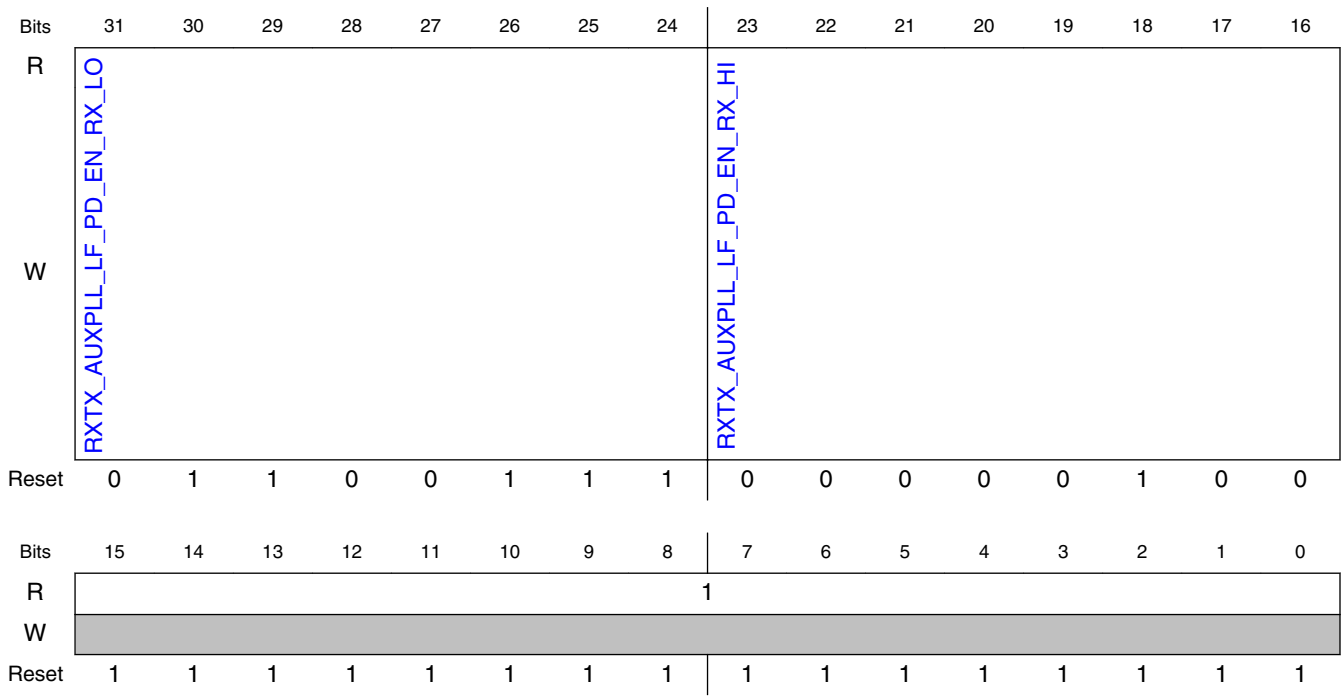
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_FCAL_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_FCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_FCAL_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_FCAL_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_FCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_FCAL_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.65 TSM_TIMING53 (TIMING53)

## A.2.4.65.1 Offset

| Register | Offset |
|---|---|
| TIMING53 | 104h |

## A.2.4.65.2 Diagram




## A.2.4.65.3 Fields

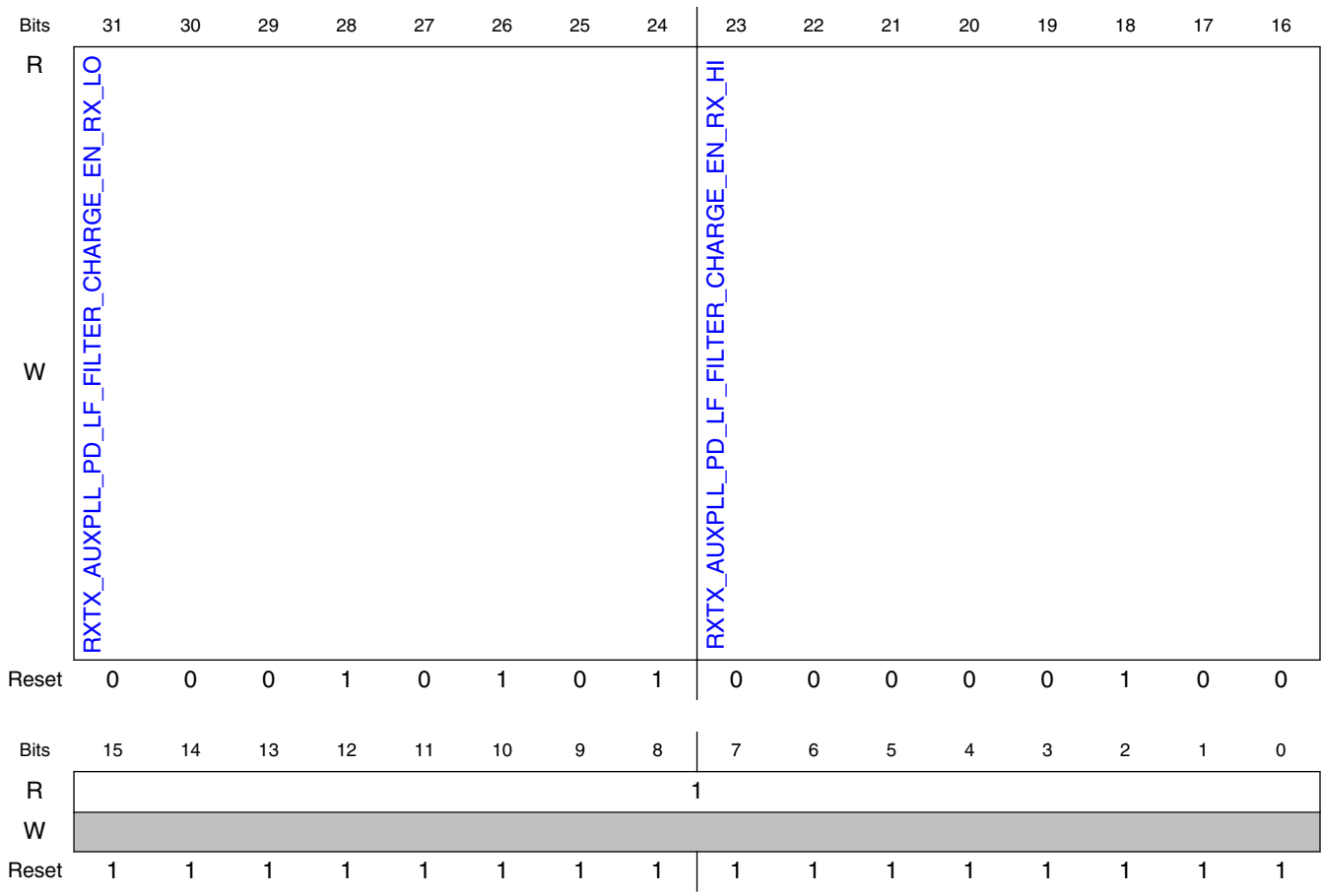
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_LF_PD_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_LF_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_LF_PD_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_LF_PD_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_LF_PD_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_LF_PD_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.66 TSM_TIMING54 (TIMING54)

## A.2.4.66.1 Offset

| Register | Offset |
|---|---|
| TIMING54 | 108h |

## A.2.4.66.2 Diagram




## A.2.4.66.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN signal or group will transition from HI to LO. |

*Table continues on the next page...*

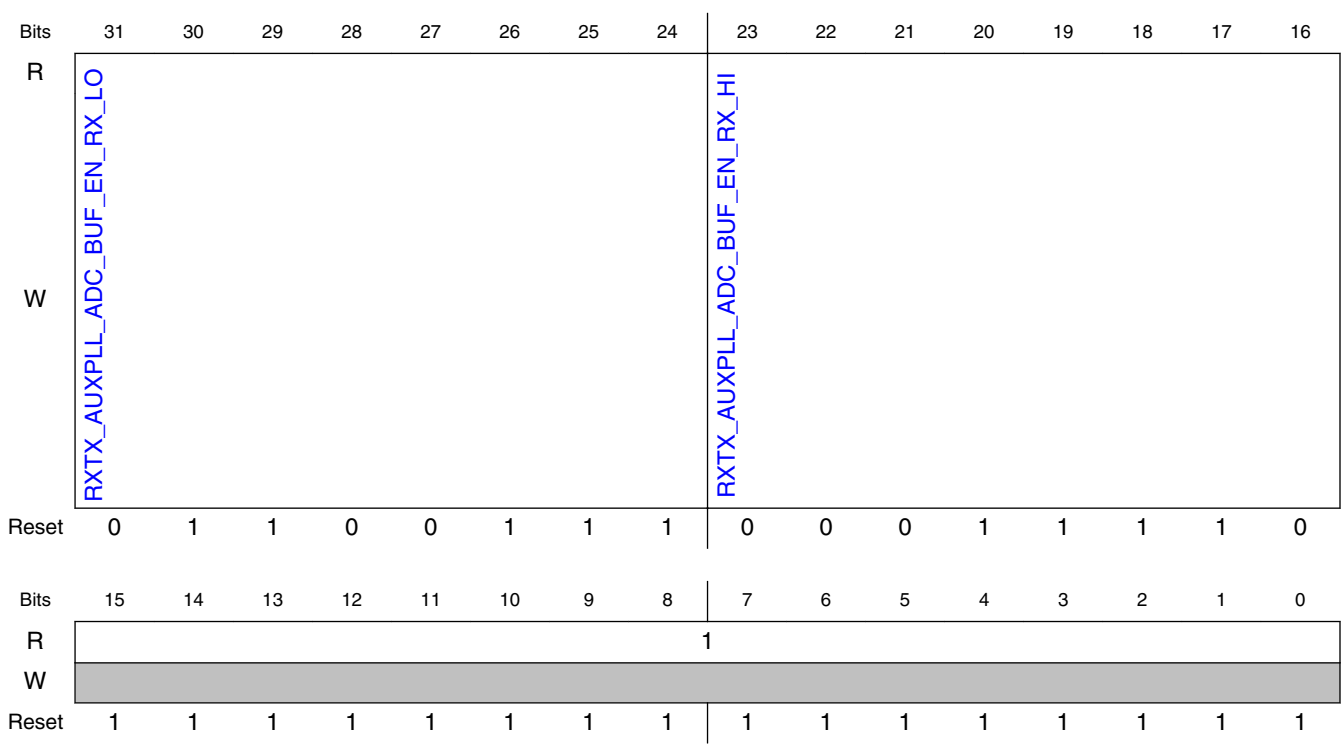
| Field | Function |
|---|---|
| 23-16<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.67 TSM_TIMING55 (TIMING55)

### A.2.4.67.1 Offset

| Register | Offset |
|---|---|
| TIMING55 | 10Ch |

### A.2.4.67.2 Diagram



### A.2.4.67.3 Fields

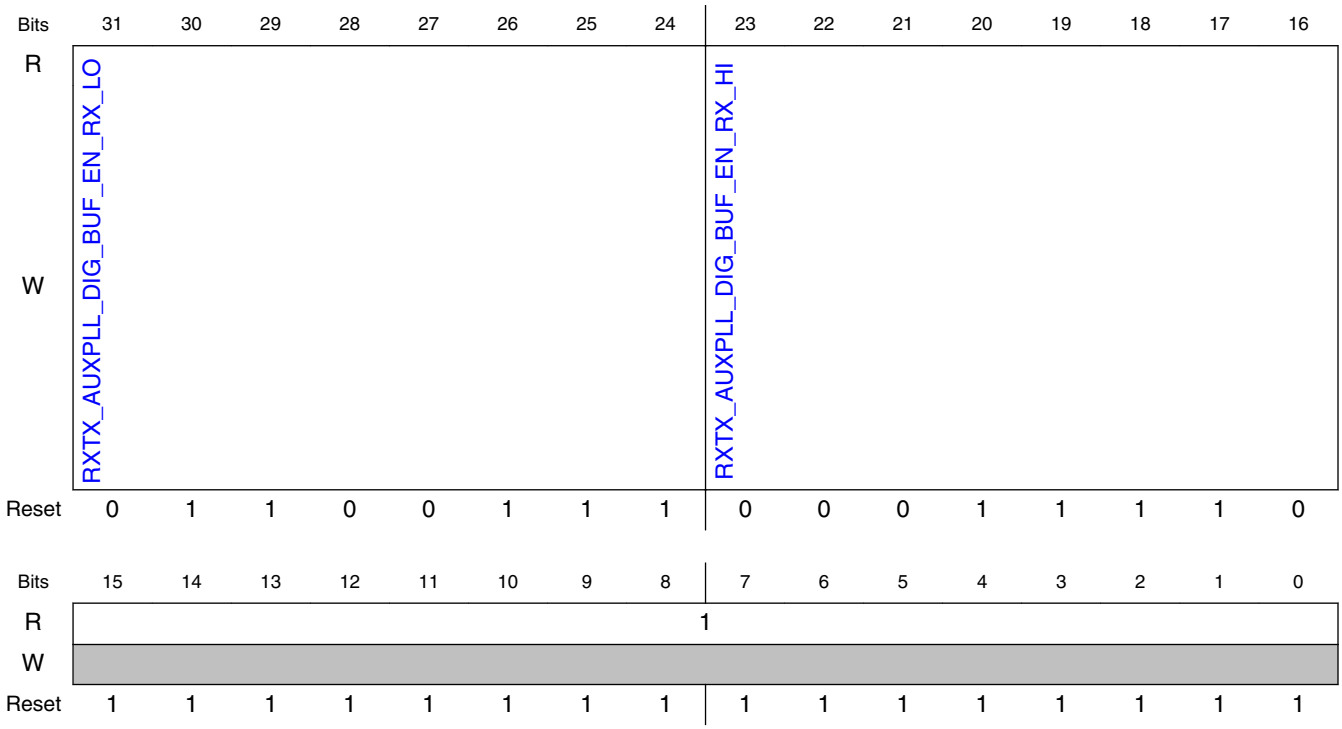
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_ADC_BUF_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_ADC_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_ADC_BUF_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_ADC_BUF_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_ADC_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_ADC_BUF_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.68 TSM_TIMING56 (TIMING56)

### A.2.4.68.1 Offset

| Register | Offset |
|---|---|
| TIMING56 | 110h |

## A.2.4.68.2 Diagram




## A.2.4.68.3 Fields

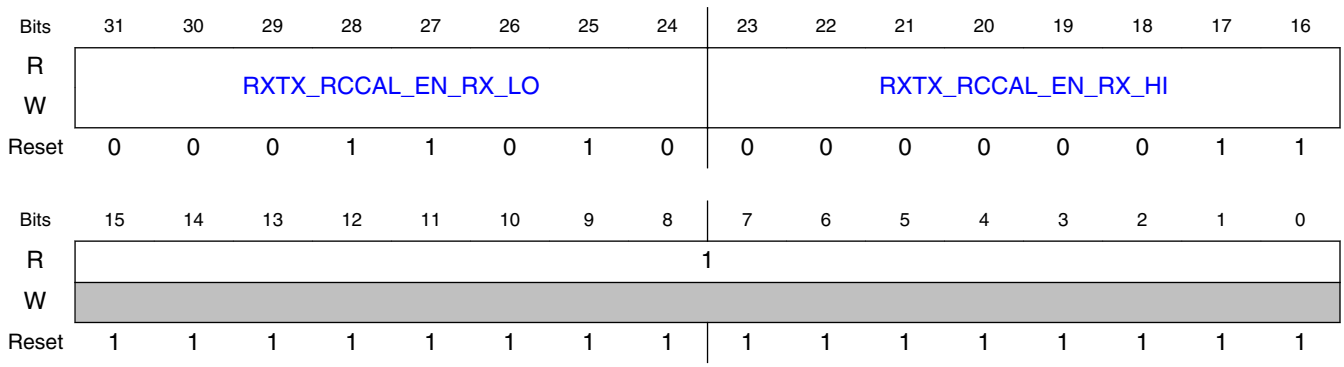
| Field | Function |
|---|---|
| 31-24<br>RXTX_AUXPLL_DIG_BUF_EN_RX_LO | De-assertion time setting for RXTX_AUXPLL_DIG_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_DIG_BUF_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_AUXPLL_DIG_BUF_EN_RX_HI | Assertion time setting for RXTX_AUXPLL_DIG_BUF_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_AUXPLL_DIG_BUF_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

# A.2.4.69 TSM_TIMING57 (TIMING57)

### A.2.4.69.1 Offset

| Register | Offset |
|---|---|
| TIMING57 | 114h |

### A.2.4.69.2 Diagram




### A.2.4.69.3 Fields

| Field | Function |
|---|---|
| 31-24<br>RXTX_RCCAL_EN_RX_LO | De-assertion time setting for RXTX_RCCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_RCCAL_EN signal or group will transition from HI to LO. |
| 23-16<br>RXTX_RCCAL_EN_RX_HI | Assertion time setting for RXTX_RCCAL_EN (RX)<br>This field sets the point during a TSM RX sequence (the tsm_count[7:0] value) at which the RXTX_RCCAL_EN signal or group will transition from LO to HI. |
| 15-0<br>— | Reserved |

## A.2.4.70 TSM_TIMING58 (TIMING58)

### A.2.4.70.1 Offset

| Register | Offset |
|---|---|
| TIMING58 | 118h |

## A.2.4.70.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX_HPM_DAC_EN_TX_LO | | | | | | | | TX_HPM_DAC_EN_TX_HI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

## A.2.4.70.3 Fields

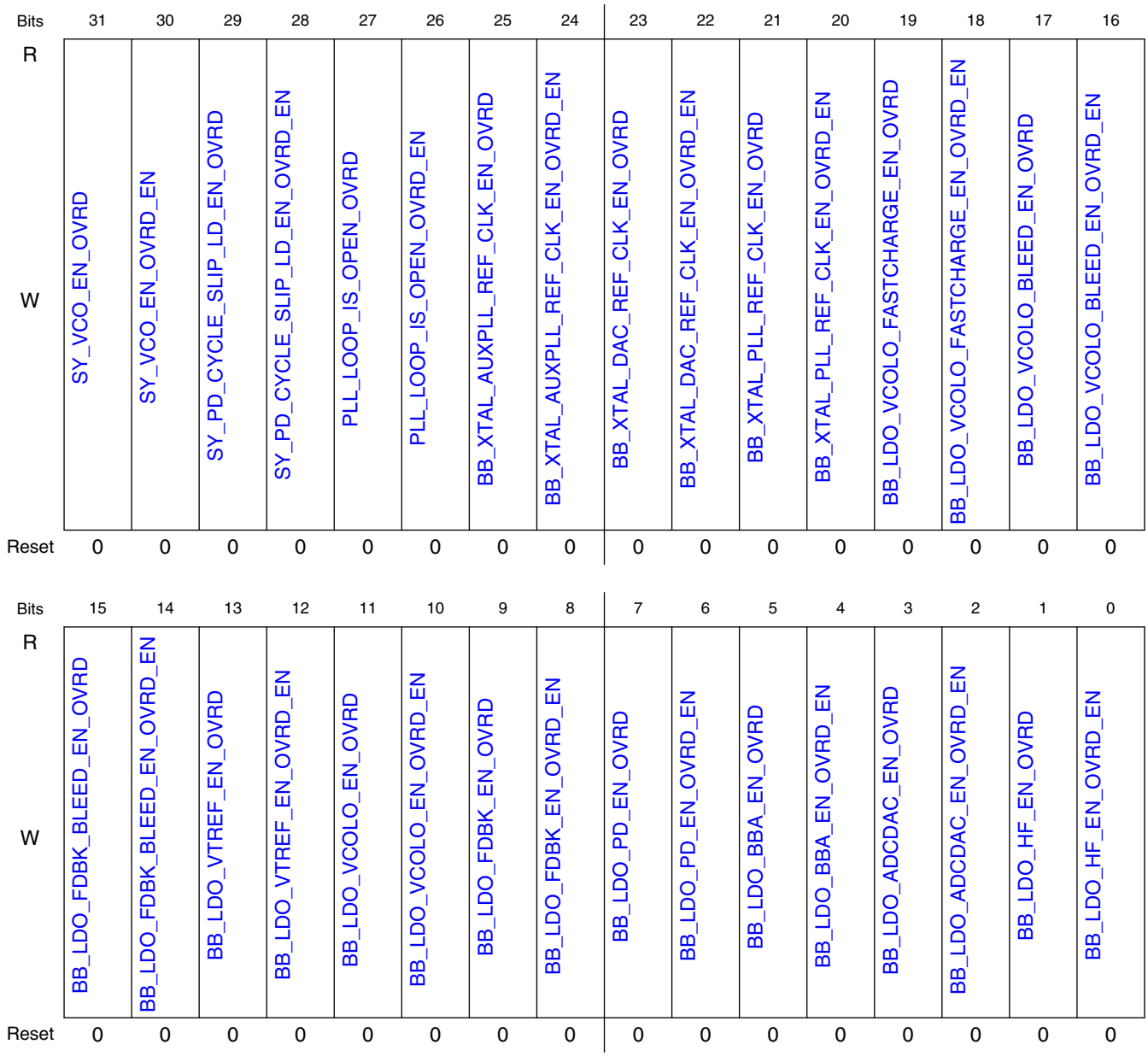
| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TX_HPM_DAC_EN_TX_LO | De-assertion time setting for TX_HPM_DAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_HPM_DAC_EN signal or group will transition from HI to LO. |
| 7-0<br>TX_HPM_DAC_EN_TX_HI | Assertion time setting for TX_HPM_DAC_EN (TX)<br>This field sets the point during a TSM TX sequence (the tsm_count[7:0] value) at which the TX_HPM_DAC_EN signal or group will transition from LO to HI. |

# A.2.4.71 TSM OVERRIDE REGISTER 0 (OVRD0)

## A.2.4.71.1 Offset

| Register | Offset |
|---|---|
| OVRD0 | 11Ch |

## A.2.4.71.2 Diagram




## A.2.4.71.3 Fields

| Field | Function |
|---|---|
| 31<br>SY_VCO_EN_OVRD | Override value for SY_VCO_EN<br>When SY_VCO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_vco_en". This bit is ignored when SY_VCO_EN_OVRD_EN==0. |
| 30 | Override control for SY_VCO_EN<br>0b - Normal operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_VCO_EN_OVRD_EN | 1b - Use the state of SY_VCO_EN_OVRD to override the signal "sy_vco_en". |
| 29<br>SY_PD_CYCLE_SLIP_LD_EN_OVRD | Override value for SY_PD_CYCLE_SLIP_LD_EN<br>When SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_cycle_slip_ld_en". This bit is ignored when SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN==0. |
| 28<br>SY_PD_CYCLE_SLIP_LD_EN_OVRD_EN | Override control for SY_PD_CYCLE_SLIP_LD_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_CYCLE_SLIP_LD_EN_OVRD to override the signal "sy_pd_cycle_slip_ld_en". |
| 27<br>PLL_LOOP_IS_OPEN_OVRD | Override value for PLL_LOOP_IS_OPEN<br>When PLL_LOOP_IS_OPEN_OVRD_EN=1, this value overrides the mission mode state of the signal "pll_loop_is_open". This bit is ignored when PLL_LOOP_IS_OPEN_OVRD_EN==0. |
| 26<br>PLL_LOOP_IS_OPEN_OVRD_EN | Override control for PLL_LOOP_IS_OPEN<br>0b - Normal operation.<br>1b - Use the state of PLL_LOOP_IS_OPEN_OVRD to override the signal "pll_loop_is_open". |
| 25<br>BB_XTAL_AUXPLL_REF_CLK_EN_OVRD | Override value for BB_XTAL_AUXPLL_REF_CLK_EN<br>When BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_auxpll_ref_clk_en". This bit is ignored when BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN==0. |
| 24<br>BB_XTAL_AUXPLL_REF_CLK_EN_OVRD_EN | Override control for BB_XTAL_AUXPLL_REF_CLK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_XTAL_AUXPLL_REF_CLK_EN_OVRD to override the signal "bb_xtal_auxpll_ref_clk_en". |
| 23<br>BB_XTAL_DAC_REF_CLK_EN_OVRD | Override value for BB_XTAL_DAC_REF_CLK_EN<br>When BB_XTAL_DAC_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_dac_ref_clk_en". This bit is ignored when BB_XTAL_DAC_REF_CLK_EN_OVRD_EN==0. |
| 22<br>BB_XTAL_DAC_REF_CLK_EN_OVRD_EN | Override control for BB_XTAL_DAC_REF_CLK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_XTAL_DAC_REF_CLK_EN_OVRD to override the signal "bb_xtal_dac_ref_clk_en". |
| 21<br>BB_XTAL_PLL_REF_CLK_EN_OVRD | Override value for BB_XTAL_PLL_REF_CLK_EN<br>When BB_XTAL_PLL_REF_CLK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_xtal_pll_ref_clk_en". This bit is ignored when BB_XTAL_PLL_REF_CLK_EN_OVRD_EN==0. |
| 20<br>BB_XTAL_PLL_REF_CLK_EN_OVRD_EN | Override control for BB_XTAL_PLL_REF_CLK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_XTAL_PLL_REF_CLK_EN_OVRD to override the signal "bb_xtal_pll_ref_clk_en". |
| 19<br>BB_LDO_VCOLO_FASTCHARGE_EN_OVRD | Override value for BB_LDO_VCOLO_FASTCHARGE_EN<br>When BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_fastcharge_en". This bit is ignored when BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN==0. |
| 18 | Override control for BB_LDO_VCOLO_FASTCHARGE_EN<br>0b - Normal operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BB_LDO_VCOLO_FASTCHARGE_EN_OVRD_EN | 1b - Use the state of BB_LDO_VCOLO_FASTCHARGE_EN_OVRD to override the signal "bb_ldo_vcolo_fastcharge_en". |
| 17<br>BB_LDO_VCOLO_BLEED_EN_OVRD | Override value for BB_LDO_VCOLO_BLEED_EN<br>When BB_LDO_VCOLO_BLEED_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_bleed_en". This bit is ignored when BB_LDO_VCOLO_BLEED_EN_OVRD_EN==0. |
| 16<br>BB_LDO_VCOLO_BLEED_EN_OVRD_EN | Override control for BB_LDO_VCOLO_BLEED_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VCOLO_BLEED_EN_OVRD to override the signal "bb_ldo_vcolo_bleed_en". |
| 15<br>BB_LDO_FDBK_BLEED_EN_OVRD | Override value for BB_LDO_FDBK_BLEED_EN<br>When BB_LDO_FDBK_BLEED_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_fdbk_bleed_en". This bit is ignored when BB_LDO_FDBK_BLEED_EN_OVRD_EN==0. |
| 14<br>BB_LDO_FDBK_BLEED_EN_OVRD_EN | Override control for BB_LDO_FDBK_BLEED_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_FDBK_BLEED_EN_OVRD to override the signal "bb_ldo_fdbk_bleed_en". |
| 13<br>BB_LDO_VTREF_EN_OVRD | Override value for BB_LDO_VTREF_EN<br>When BB_LDO_VTREF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vtref_en". This bit is ignored when BB_LDO_VTREF_EN_OVRD_EN==0. |
| 12<br>BB_LDO_VTREF_EN_OVRD_EN | Override control for BB_LDO_VTREF_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VTREF_EN_OVRD to override the signal "bb_ldo_vtref_en". |
| 11<br>BB_LDO_VCOLO_EN_OVRD | Override value for BB_LDO_VCOLO_EN<br>When BB_LDO_VCOLO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_vcolo_en". This bit is ignored when BB_LDO_VCOLO_EN_OVRD_EN==0. |
| 10<br>BB_LDO_VCOLO_EN_OVRD_EN | Override control for BB_LDO_VCOLO_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_VCOLO_EN_OVRD to override the signal "bb_ldo_vcolo_en". |
| 9<br>BB_LDO_FDBK_EN_OVRD | Override value for BB_LDO_FDBK_EN<br>When BB_LDO_FDBK_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_fdbk_en". This bit is ignored when BB_LDO_FDBK_EN_OVRD_EN==0. |
| 8<br>BB_LDO_FDBK_EN_OVRD_EN | Override control for BB_LDO_FDBK_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_FDBK_EN_OVRD to override the signal "bb_ldo_fdbk_en". |
| 7<br>BB_LDO_PD_EN_OVRD | Override value for BB_LDO_PD_EN<br>When BB_LDO_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_pd_en". This bit is ignored when BB_LDO_PD_EN_OVRD_EN==0. |
| 6<br>BB_LDO_PD_EN_OVRD_EN | Override control for BB_LDO_PD_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_PD_EN_OVRD to override the signal "bb_ldo_pd_en". |

*Table continues on the next page...*

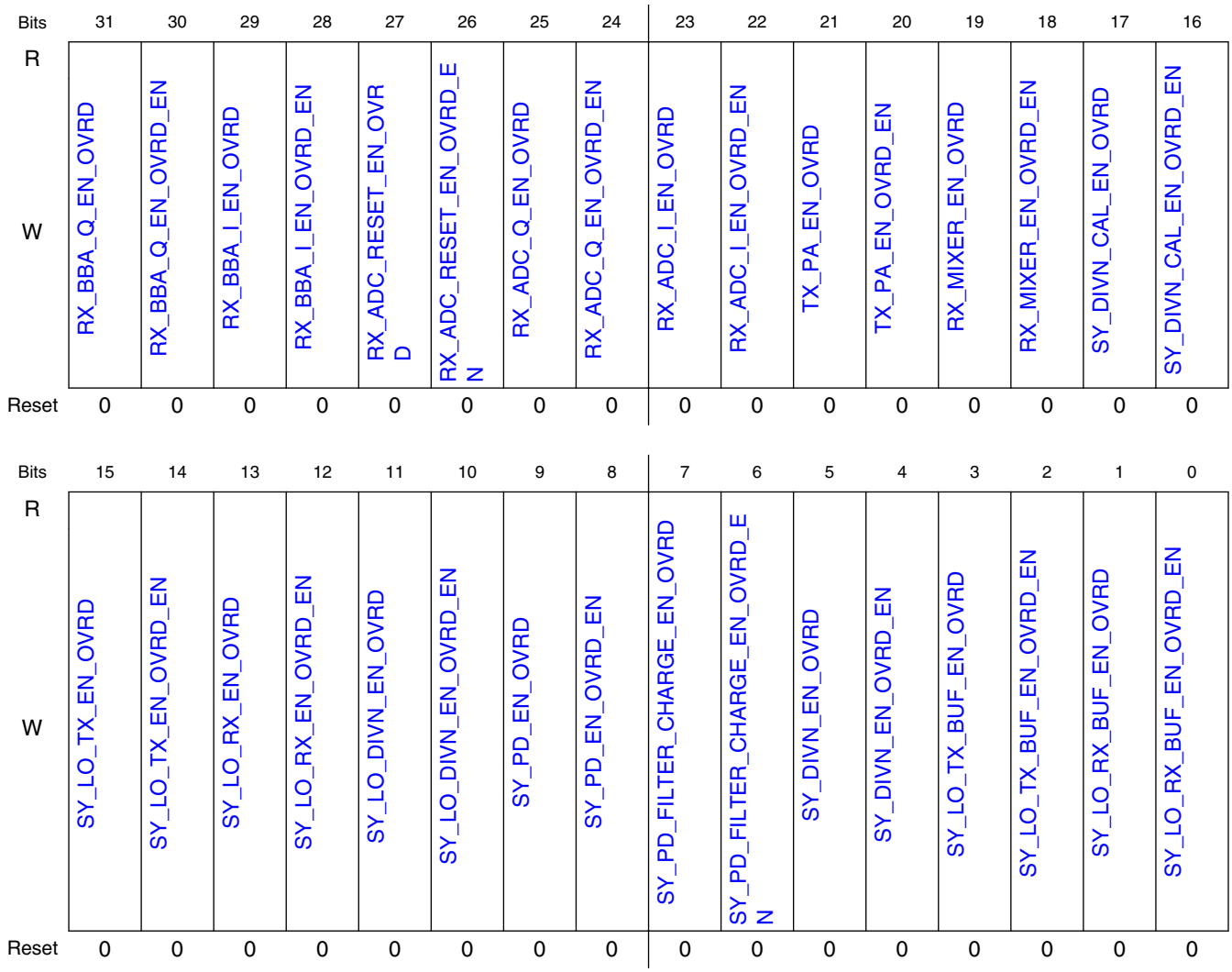
| Field | Function |
|---|---|
| 5<br>BB_LDO_BBA_EN_OVRD | Override value for BB_LDO_BBA_EN<br>When BB_LDO_BBA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_bba_en". This bit is ignored when BB_LDO_BBA_EN_OVRD_EN==0. |
| 4<br>BB_LDO_BBA_EN_OVRD_EN | Override control for BB_LDO_BBA_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_BBA_EN_OVRD to override the signal "bb_ldo_bba_en". |
| 3<br>BB_LDO_ADCDAC_EN_OVRD | Override value for BB_LDO_ADCDAC_EN<br>When BB_LDO_ADCDAC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_adcdac_en". This bit is ignored when BB_LDO_ADCDAC_EN_OVRD_EN==0. |
| 2<br>BB_LDO_ADCDAC_EN_OVRD_EN | Override control for BB_LDO_ADCDAC_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_ADCDAC_EN_OVRD to override the signal "bb_ldo_adcdac_en". |
| 1<br>BB_LDO_HF_EN_OVRD | Override value for BB_LDO_HF_EN<br>When BB_LDO_HF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "bb_ldo_hf_en". This bit is ignored when BB_LDO_HF_EN_OVRD_EN==0. |
| 0<br>BB_LDO_HF_EN_OVRD_EN | Override control for BB_LDO_HF_EN<br>0b - Normal operation.<br>1b - Use the state of BB_LDO_HF_EN_OVRD to override the signal "bb_ldo_hf_en". |

## A.2.4.72 TSM OVERRIDE REGISTER 1 (OVRD1)

### A.2.4.72.1 Offset

| Register | Offset |
|---|---|
| OVRD1 | 120h |

## A.2.4.72.2 Diagram




## A.2.4.72.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_BBA_Q_EN_OVRD | Override value for RX_BBA_Q_EN<br>When RX_BBA_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_q_en". This bit is ignored when RX_BBA_Q_EN_OVRD_EN==0. |
| 30<br>RX_BBA_Q_EN_OVRD_EN | Override control for RX_BBA_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_BBA_Q_EN_OVRD to override the signal "rx_bba_q_en". |
| 29<br>RX_BBA_I_EN_OVRD | Override value for RX_BBA_I_EN<br>When RX_BBA_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_i_en". This bit is ignored when RX_BBA_I_EN_OVRD_EN==0. |
| 28 | Override control for RX_BBA_I_EN |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RX_BBA_I_EN_OVRD_EN | 0b - Normal operation.<br>1b - Use the state of RX_BBA_I_EN_OVRD to override the signal "rx_bba_i_en". |
| 27<br>RX_ADC_RESET_EN_OVRD | Override value for RX_ADC_RESET_EN<br>When RX_ADC_RESET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_reset_en". This bit is ignored when RX_ADC_RESET_EN_OVRD_EN==0. |
| 26<br>RX_ADC_RESET_EN_OVRD_EN | Override control for RX_ADC_RESET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_RESET_EN_OVRD to override the signal "rx_adc_reset_en". |
| 25<br>RX_ADC_Q_EN_OVRD | Override value for RX_ADC_Q_EN<br>When RX_ADC_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_q_en". This bit is ignored when RX_ADC_Q_EN_OVRD_EN==0. |
| 24<br>RX_ADC_Q_EN_OVRD_EN | Override control for RX_ADC_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_Q_EN_OVRD to override the signal "rx_adc_q_en". |
| 23<br>RX_ADC_I_EN_OVRD | Override value for RX_ADC_I_EN<br>When RX_ADC_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_adc_i_en". This bit is ignored when RX_ADC_I_EN_OVRD_EN==0. |
| 22<br>RX_ADC_I_EN_OVRD_EN | Override control for RX_ADC_I_EN<br>0b - Normal operation.<br>1b - Use the state of RX_ADC_I_EN_OVRD to override the signal "rx_adc_i_en". |
| 21<br>TX_PA_EN_OVRD | Override value for TX_PA_EN<br>When TX_PA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_pa_en". This bit is ignored when TX_PA_EN_OVRD_EN==0. |
| 20<br>TX_PA_EN_OVRD_EN | Override control for TX_PA_EN<br>0b - Normal operation.<br>1b - Use the state of TX_PA_EN_OVRD to override the signal "tx_pa_en". |
| 19<br>RX_MIXER_EN_OVRD | Override value for RX_MIXER_EN<br>When RX_MIXER_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_mixer_en". This bit is ignored when RX_MIXER_EN_OVRD_EN==0. |
| 18<br>RX_MIXER_EN_OVRD_EN | Override control for RX_MIXER_EN<br>0b - Normal operation.<br>1b - Use the state of RX_MIXER_EN_OVRD to override the signal "rx_mixer_en". |
| 17<br>SY_DIVN_CAL_EN_OVRD | Override value for SY_DIVN_CAL_EN<br>When SY_DIVN_CAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_divn_cal_en". This bit is ignored when SY_DIVN_CAL_EN_OVRD_EN==0. |
| 16<br>SY_DIVN_CAL_EN_OVRD_EN | Override control for SY_DIVN_CAL_EN<br>0b - Normal operation.<br>1b - Use the state of SY_DIVN_CAL_EN_OVRD to override the signal "sy_divn_cal_en". |
| 15<br>SY_LO_TX_EN_OVRD | Override value for SY_LO_TX_EN<br>When SY_LO_TX_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_tx_en". This bit is ignored when SY_LO_TX_EN_OVRD_EN==0. |
| 14<br>SY_LO_TX_EN_OVRD_EN | Override control for SY_LO_TX_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_TX_EN_OVRD to override the signal "sy_lo_tx_en". |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13<br>SY_LO_RX_EN_OVRD | Override value for SY_LO_RX_EN<br>When SY_LO_RX_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_rx_en". This bit is ignored when SY_LO_RX_EN_OVRD_EN==0. |
| 12<br>SY_LO_RX_EN_OVRD_EN | Override control for SY_LO_RX_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_RX_EN_OVRD to override the signal "sy_lo_rx_en". |
| 11<br>SY_LO_DIVN_EN_OVRD | Override value for SY_LO_DIVN_EN<br>When SY_LO_DIVN_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_divn_en". This bit is ignored when SY_LO_DIVN_EN_OVRD_EN==0. |
| 10<br>SY_LO_DIVN_EN_OVRD_EN | Override control for SY_LO_DIVN_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_DIVN_EN_OVRD to override the signal "sy_lo_divn_en". |
| 9<br>SY_PD_EN_OVRD | Override value for SY_PD_EN<br>When SY_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_en". This bit is ignored when SY_PD_EN_OVRD_EN==0. |
| 8<br>SY_PD_EN_OVRD_EN | Override control for SY_PD_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_EN_OVRD to override the signal "sy_pd_en". |
| 7<br>SY_PD_FILTER_CHARGE_EN_OVRD | Override value for SY_PD_FILTER_CHARGE_EN<br>When SY_PD_FILTER_CHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_pd_filter_charge_en". This bit is ignored when SY_PD_FILTER_CHARGE_EN_OVRD_EN==0. |
| 6<br>SY_PD_FILTER_CHARGE_EN_OVRD_EN | Override control for SY_PD_FILTER_CHARGE_EN<br>0b - Normal operation.<br>1b - Use the state of SY_PD_FILTER_CHARGE_EN_OVRD to override the signal "sy_pd_filter_charge_en". |
| 5<br>SY_DIVN_EN_OVRD | Override value for SY_DIVN_EN<br>When SY_DIVN_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_divn_en". This bit is ignored when SY_DIVN_EN_OVRD_EN==0. |
| 4<br>SY_DIVN_EN_OVRD_EN | Override control for SY_DIVN_EN<br>0b - Normal operation.<br>1b - Use the state of SY_DIVN_EN_OVRD to override the signal "sy_divn_en". |
| 3<br>SY_LO_TX_BUF_EN_OVRD | Override value for SY_LO_TX_BUF_EN<br>When SY_LO_TX_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_tx_buf_en". This bit is ignored when SY_LO_TX_BUF_EN_OVRD_EN==0. |
| 2<br>SY_LO_TX_BUF_EN_OVRD_EN | Override control for SY_LO_TX_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_TX_BUF_EN_OVRD to override the signal "sy_lo_tx_buf_en". |
| 1<br>SY_LO_RX_BUF_EN_OVRD | Override value for SY_LO_RX_BUF_EN<br>When SY_LO_RX_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sy_lo_rx_buf_en". This bit is ignored when SY_LO_RX_BUF_EN_OVRD_EN==0. |
| 0<br>SY_LO_RX_BUF_EN_OVRD_EN | Override control for SY_LO_RX_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of SY_LO_RX_BUF_EN_OVRD to override the signal "sy_lo_rx_buf_en". |

# A.2.4.73 TSM OVERRIDE REGISTER 2 (OVRD2)

## A.2.4.73.1 Offset

| Register | Offset |
|---|---|
| OVRD2 | 124h |

## A.2.4.73.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FREQ_TARG_LD_EN_OVRD | FREQ_TARG_LD_EN_OVRD_EN | DCOC_INIT_OVRD | DCOC_INIT_OVRD_EN | DCOC_EN_OVRD | DCOC_EN_OVRD_EN | RX_PHY_EN_OVRD | RX_PHY_EN_OVRD_EN | SIGMA_DELTA_EN_OVRD | SIGMA_DELTA_EN_OVRD_EN | RX_INIT_OVRD | RX_INIT_OVRD_EN | RX_DIG_EN_OVRD | RX_DIG_EN_OVRD_EN | TX_DIG_EN_OVRD | TX_DIG_EN_OVRD_EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PLL_DIG_EN_OVRD | PLL_DIG_EN_OVRD_EN | RX_TZA_DCOC_EN_OVRD | RX_TZA_DCOC_EN_OVRD_EN | RX_TZA_PDET_EN_OVRD | RX_TZA_PDET_EN_OVRD_EN | RX_TZA_Q_EN_OVRD | RX_TZA_Q_EN_OVRD_EN | RX_TZA_I_EN_OVRD | RX_TZA_I_EN_OVRD_EN | RX_LNA_EN_OVRD | RX_LNA_EN_OVRD_EN | RX_BBA_DCOC_EN_OVRD | RX_BBA_DCOC_EN_OVRD_EN | RX_BBA_PDET_EN_OVRD | RX_BBA_PDET_EN_OVRD_EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.4.73.3 Fields

| Field | Function |
|---|---|
| 31 | Override value for FREQ_TARG_LD_EN |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FREQ_TARG_LD_EN_OVRD | When FREQ_TARG_LD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "freq_targ_ld_en". This bit is ignored when FREQ_TARG_LD_EN_OVRD_EN==0. |
| 30<br>FREQ_TARG_LD_EN_OVRD_EN | Override control for FREQ_TARG_LD_EN<br>0b - Normal operation.<br>1b - Use the state of FREQ_TARG_LD_EN_OVRD to override the signal "freq_targ_ld_en". |
| 29<br>DCOC_INIT_OVRD | Override value for DCOC_INIT<br>When DCOC_INIT_OVRD_EN=1, this value overrides the mission mode state of the signal "dcoc_init". This bit is ignored when DCOC_INIT_OVRD_EN==0. |
| 28<br>DCOC_INIT_OVRD_EN | Override control for DCOC_INIT<br>0b - Normal operation.<br>1b - Use the state of DCOC_INIT_OVRD to override the signal "dcoc_init". |
| 27<br>DCOC_EN_OVRD | Override value for DCOC_EN<br>When DCOC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "dcoc_en". This bit is ignored when DCOC_EN_OVRD_EN==0. |
| 26<br>DCOC_EN_OVRD_EN | Override control for DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of DCOC_EN_OVRD to override the signal "dcoc_en". |
| 25<br>RX_PHY_EN_OVRD | Override value for RX_PHY_EN<br>When RX_PHY_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_phy_en". This bit is ignored when RX_PHY_EN_OVRD_EN==0. |
| 24<br>RX_PHY_EN_OVRD_EN | Override control for RX_PHY_EN<br>0b - Normal operation.<br>1b - Use the state of RX_PHY_EN_OVRD to override the signal "rx_phy_en". |
| 23<br>SIGMA_DELTA_EN_OVRD | Override value for SIGMA_DELTA_EN<br>When SIGMA_DELTA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "sigma_delta_en". This bit is ignored when SIGMA_DELTA_EN_OVRD_EN==0. |
| 22<br>SIGMA_DELTA_EN_OVRD_EN | Override control for SIGMA_DELTA_EN<br>0b - Normal operation.<br>1b - Use the state of SIGMA_DELTA_EN_OVRD to override the signal "sigma_delta_en". |
| 21<br>RX_INIT_OVRD | Override value for RX_INIT<br>When RX_INIT_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_init". This bit is ignored when RX_INIT_OVRD_EN==0. |
| 20<br>RX_INIT_OVRD_EN | Override control for RX_INIT<br>0b - Normal operation.<br>1b - Use the state of RX_INIT_OVRD to override the signal "rx_init". |
| 19<br>RX_DIG_EN_OVRD | Override value for RX_DIG_EN<br>When RX_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_dig_en". This bit is ignored when RX_DIG_EN_OVRD_EN==0. |
| 18<br>RX_DIG_EN_OVRD_EN | Override control for RX_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of RX_DIG_EN_OVRD to override the signal "rx_dig_en". |
| 17<br>TX_DIG_EN_OVRD | Override value for TX_DIG_EN<br>When TX_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_dig_en". This bit is ignored when TX_DIG_EN_OVRD_EN==0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 16<br>TX_DIG_EN_OVRD_EN | Override control for TX_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of TX_DIG_EN_OVRD to override the signal "tx_dig_en". |
| 15<br>PLL_DIG_EN_OVRD | Override value for PLL_DIG_EN<br>When PLL_DIG_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "pll_dig_en". This bit is ignored when PLL_DIG_EN_OVRD_EN==0. |
| 14<br>PLL_DIG_EN_OVRD_EN | Override control for PLL_DIG_EN<br>0b - Normal operation.<br>1b - Use the state of PLL_DIG_EN_OVRD to override the signal "pll_dig_en". |
| 13<br>RX_TZA_DCOC_EN_OVRD | Override control for RX_TZA_DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_DCOC_EN_OVRD to override the signal "rx_tza_dcoc_en". |
| 12<br>RX_TZA_DCOC_EN_OVRD_EN | Override control for RX_TZA_DCOC_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_DCOC_EN_OVRD to override the signal "rx_tza_dcoc_en". |
| 11<br>RX_TZA_PDET_EN_OVRD | Override value for RX_TZA_PDET_EN<br>When RX_TZA_PDET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_pdet_en". This bit is ignored when RX_TZA_PDET_EN_OVRD_EN==0. |
| 10<br>RX_TZA_PDET_EN_OVRD_EN | Override control for RX_TZA_PDET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_PDET_EN_OVRD to override the signal "rx_tza_pdet_en". |
| 9<br>RX_TZA_Q_EN_OVRD | Override value for RX_TZA_Q_EN<br>When RX_TZA_Q_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_q_en". This bit is ignored when RX_TZA_Q_EN_OVRD_EN==0. |
| 8<br>RX_TZA_Q_EN_OVRD_EN | Override control for RX_TZA_Q_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_Q_EN_OVRD to override the signal "rx_tza_q_en". |
| 7<br>RX_TZA_I_EN_OVRD | Override value for RX_TZA_I_EN<br>When RX_TZA_I_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_tza_i_en". This bit is ignored when RX_TZA_I_EN_OVRD_EN==0. |
| 6<br>RX_TZA_I_EN_OVRD_EN | Override control for RX_TZA_I_EN<br>0b - Normal operation.<br>1b - Use the state of RX_TZA_I_EN_OVRD to override the signal "rx_tza_i_en". |
| 5<br>RX_LNA_EN_OVRD | Override value for RX_LNA_EN<br>When RX_LNA_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_lna_en". This bit is ignored when RX_LNA_EN_OVRD_EN==0. |
| 4<br>RX_LNA_EN_OVRD_EN | Override control for RX_LNA_EN<br>0b - Normal operation.<br>1b - Use the state of RX_LNA_EN_OVRD to override the signal "rx_lna_en". |
| 3<br>RX_BBA_DCOC_EN_OVRD | Override value for RX_BBA_DCOC_EN<br>When RX_BBA_DCOC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_dcoc_en". This bit is ignored when RX_BBA_DCOC_EN_OVRD_EN==0. |
| 2 | Override control for RX_BBA_DCOC_EN<br>0b - Normal operation. |

*Table continues on the next page...*

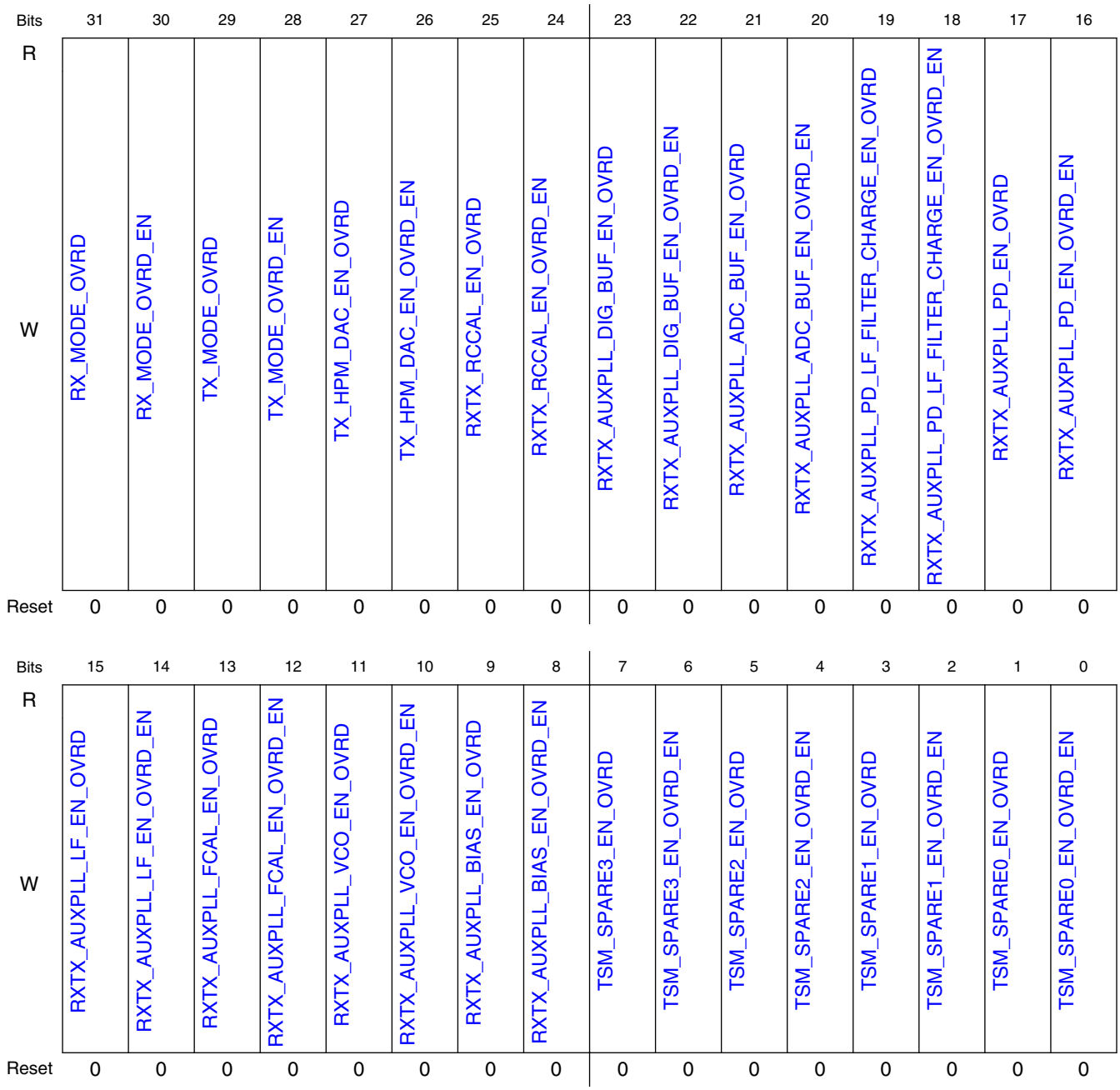
| Field | Function |
|---|---|
| RX_BBA_DCOC_EN_OVRD_EN | 1b - Use the state of RX_BBA_DCOC_EN_OVRD to override the signal "rx_bba_dcoc_en". |
| 1<br>RX_BBA_PDET_EN_OVRD | Override value for RX_BBA_PDET_EN<br>When RX_BBA_PDET_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_bba_pdet_en". This bit is ignored when RX_BBA_PDET_EN_OVRD_EN==0. |
| 0<br>RX_BBA_PDET_EN_OVRD_EN | Override control for RX_BBA_PDET_EN<br>0b - Normal operation.<br>1b - Use the state of RX_BBA_PDET_EN_OVRD to override the signal "rx_bba_pdet_en". |

## A.2.4.74 TSM OVERRIDE REGISTER 3 (OVRD3)

### A.2.4.74.1 Offset

| Register | Offset |
|---|---|
| OVRD3 | 128h |

## A.2.4.74.2 Diagram




## A.2.4.74.3 Fields

| Field | Function |
|---|---|
| 31<br>RX_MODE_OVRD | Override value for RX_MODE<br>When RX_MODE_OVRD_EN=1, this value overrides the mission mode state of the signal "rx_mode". This bit is ignored when RX_MODE_OVRD_EN==0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 30<br>RX_MODE_OVRD_EN | Override control for RX_MODE<br>0b - Normal operation.<br>1b - Use the state of RX_MODE_OVRD to override the signal "rx_mode". |
| 29<br>TX_MODE_OVRD | Override value for TX_MODE<br>When TX_MODE_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_mode". This bit is ignored when TX_MODE_OVRD_EN==0. |
| 28<br>TX_MODE_OVRD_EN | Override control for TX_MODE<br>0b - Normal operation.<br>1b - Use the state of TX_MODE_OVRD to override the signal "tx_mode". |
| 27<br>TX_HPM_DAC_EN_OVRD | Override value for TX_HPM_DAC_EN<br>When TX_HPM_DAC_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tx_hpm_dac_en". This bit is ignored when TX_HPM_DAC_EN_OVRD_EN==0. |
| 26<br>TX_HPM_DAC_EN_OVRD_EN | Override control for TX_HPM_DAC_EN<br>0b - Normal operation.<br>1b - Use the state of TX_HPM_DAC_EN_OVRD to override the signal "tx_hpm_dac_en". |
| 25<br>RXTX_RCCAL_EN_OVRD | Override value for RXTX_RCCAL_EN<br>When RXTX_RCCAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_rccal_en". This bit is ignored when RXTX_RCCAL_EN_OVRD_EN==0. |
| 24<br>RXTX_RCCAL_EN_OVRD_EN | Override control for RXTX_RCCAL_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_RCCAL_EN_OVRD to override the signal "rxtx_rccal_en". |
| 23<br>RXTX_AUXPLL_DIG_BUF_EN_OVRD | Override value for RXTX_AUXPLL_DIG_BUF_EN<br>When RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_dig_buf_en". This bit is ignored when RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN==0. |
| 22<br>RXTX_AUXPLL_DIG_BUF_EN_OVRD_EN | Override control for RXTX_AUXPLL_DIG_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_DIG_BUF_EN_OVRD to override the signal "rxtx_auxpll_dig_buf_en". |
| 21<br>RXTX_AUXPLL_ADC_BUF_EN_OVRD | Override value for RXTX_AUXPLL_ADC_BUF_EN<br>When RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_adc_buf_en". This bit is ignored when RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN==0. |
| 20<br>RXTX_AUXPLL_ADC_BUF_EN_OVRD_EN | Override control for RXTX_AUXPLL_ADC_BUF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_ADC_BUF_EN_OVRD to override the signal "rxtx_auxpll_adc_buf_en". |
| 19<br>RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD | Override value for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN<br>When RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_pd_lf_filter_charge_en". This bit is ignored when RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN==0. |
| 18 | Override control for RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD to override the signal "rxtx_auxpll_pd_lf_filter_charge_en". |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RXTX_AUXPLL_PD_LF_FILTER_CHARGE_EN_OVRD_EN | |
| 17<br>RXTX_AUXPLL_PD_EN_OVRD | Override value for RXTX_AUXPLL_PD_EN<br>When RXTX_AUXPLL_PD_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_pd_en". This bit is ignored when RXTX_AUXPLL_PD_EN_OVRD_EN==0. |
| 16<br>RXTX_AUXPLL_PD_EN_OVRD_EN | Override control for RXTX_AUXPLL_PD_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_PD_EN_OVRD to override the signal "rxtx_auxpll_pd_en". |
| 15<br>RXTX_AUXPLL_LF_EN_OVRD | Override value for RXTX_AUXPLL_LF_EN<br>When RXTX_AUXPLL_LF_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_lf_en". This bit is ignored when RXTX_AUXPLL_LF_EN_OVRD_EN==0. |
| 14<br>RXTX_AUXPLL_LF_EN_OVRD_EN | Override control for RXTX_AUXPLL_LF_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_LF_EN_OVRD to override the signal "rxtx_auxpll_lf_en". |
| 13<br>RXTX_AUXPLL_FCAL_EN_OVRD | Override value for RXTX_AUXPLL_FCAL_EN<br>When RXTX_AUXPLL_FCAL_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_fcal_en". This bit is ignored when RXTX_AUXPLL_FCAL_EN_OVRD_EN==0. |
| 12<br>RXTX_AUXPLL_FCAL_EN_OVRD_EN | Override control for RXTX_AUXPLL_FCAL_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_FCAL_EN_OVRD to override the signal "rxtx_auxpll_fcal_en". |
| 11<br>RXTX_AUXPLL_VCO_EN_OVRD | Override value for RXTX_AUXPLL_VCO_EN<br>When RXTX_AUXPLL_VCO_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_vco_en". This bit is ignored when RXTX_AUXPLL_VCO_EN_OVRD_EN==0. |
| 10<br>RXTX_AUXPLL_VCO_EN_OVRD_EN | Override control for RXTX_AUXPLL_VCO_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_VCO_EN_OVRD to override the signal "rxtx_auxpll_vco_en". |
| 9<br>RXTX_AUXPLL_BIAS_EN_OVRD | Override value for RXTX_AUXPLL_BIAS_EN<br>When RXTX_AUXPLL_BIAS_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "rxtx_auxpll_bias_en". This bit is ignored when RXTX_AUXPLL_BIAS_EN_OVRD_EN==0. |
| 8<br>RXTX_AUXPLL_BIAS_EN_OVRD_EN | Override control for RXTX_AUXPLL_BIAS_EN<br>0b - Normal operation.<br>1b - Use the state of RXTX_AUXPLL_BIAS_EN_OVRD to override the signal "rxtx_auxpll_bias_en". |
| 7<br>TSM_SPARE3_EN_OVRD | Override value for TSM_SPARE3_EN<br>When TSM_SPARE3_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare3_en". This bit is ignored when TSM_SPARE3_EN_OVRD_EN==0. |
| 6 | Override control for TSM_SPARE3_EN<br>0b - Normal operation. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TSM_SPARE3_EN_OVRD_EN | 1b - Use the state of TSM_SPARE3_EN_OVRD to override the signal "tsm_spare3_en". |
| 5<br>TSM_SPARE2_EN_OVRD | Override value for TSM_SPARE2_EN<br>When TSM_SPARE2_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare2_en". This bit is ignored when TSM_SPARE2_EN_OVRD_EN==0. |
| 4<br>TSM_SPARE2_EN_OVRD_EN | Override control for TSM_SPARE2_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE2_EN_OVRD to override the signal "tsm_spare2_en". |
| 3<br>TSM_SPARE1_EN_OVRD | Override value for TSM_SPARE1_EN<br>When TSM_SPARE1_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare1_en". This bit is ignored when TSM_SPARE1_EN_OVRD_EN==0. |
| 2<br>TSM_SPARE1_EN_OVRD_EN | Override control for TSM_SPARE1_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE1_EN_OVRD to override the signal "tsm_spare1_en". |
| 1<br>TSM_SPARE0_EN_OVRD | Override value for TSM_SPARE0_EN<br>When TSM_SPARE0_EN_OVRD_EN=1, this value overrides the mission mode state of the signal "tsm_spare0_en". This bit is ignored when TSM_SPARE0_EN_OVRD_EN==0. |
| 0<br>TSM_SPARE0_EN_OVRD_EN | Override control for TSM_SPARE0_EN<br>0b - Normal operation.<br>1b - Use the state of TSM_SPARE0_EN_OVRD to override the signal "tsm_spare0_en". |

## A.2.5 XCVR_WOR register descriptions

### A.2.5.1 XCVR_WOR_ADDR Memory map

XCVR_WOR base address: 4103_04C0h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | WAKE-ON-RADIO CONTROL REGISTER (WOR_CTRL) | 32 | RW | 0007_0000h |
| 4h | WAKE-ON-RADIO TIMEOUT REGISTER (WOR_TIMEOUT) | 32 | RW | 00FF_0000h |
| 8h | WAKE-ON-RADIO TIMESTAMP 1 (TIMESTAMP1) | 32 | RO | 0000_0000h |
| Ch | WAKE-ON-RADIO TIMESTAMP 2 (TIMESTAMP2) | 32 | RO | 0000_0000h |
| 10h | WAKE-ON-RADIO TIMESTAMP 3 (TIMESTAMP3) | 32 | RO | 0000_0000h |
| 14h | WAKE-ON-RADIO STATUS REGISTER (WOR_STATUS) | 32 | RO | 0000_0000h |
| 18h | WINDOW-WIDENING CONTROL REGISTER (WW_CTRL) | 32 | RW | 0000_0002h |
| 1Ch | FREQUENCY HOP CONTROL REGISTER (HOP_CTRL) | 32 | RW | 0030_0000h |
| 20h | SLOT 0 DESCRIPTOR (LSB) (SLOT0_DESC0) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

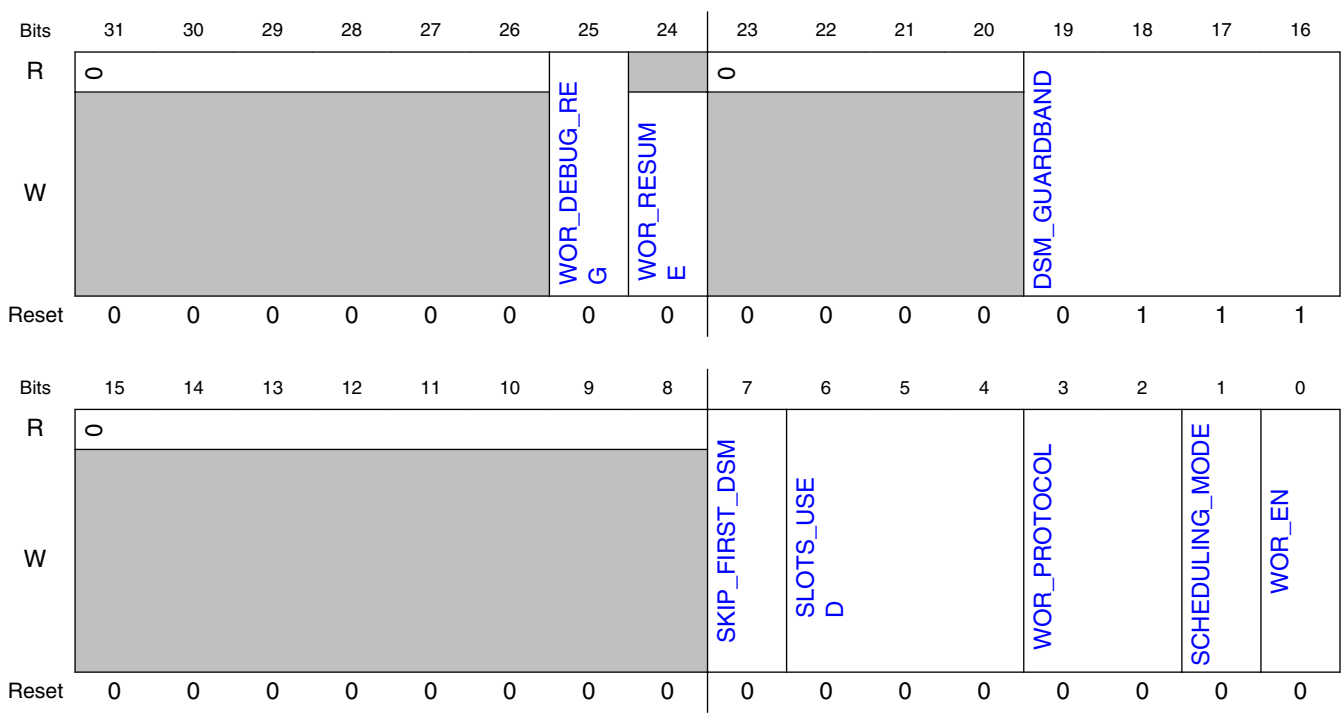
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 24h | SLOT 0 DESCRIPTOR (MSB) (SLOT0_DESC1) | 32 | RW | 0000_0000h |
| 28h | SLOT 1 DESCRIPTOR (LSB) (SLOT1_DESC0) | 32 | RW | 0000_0000h |
| 2Ch | SLOT 1 DESCRIPTOR (MSB) (SLOT1_DESC1) | 32 | RW | 0000_0000h |
| 30h | SLOT 2 DESCRIPTOR (LSB) (SLOT2_DESC0) | 32 | RW | 0000_0000h |
| 34h | SLOT 2 DESCRIPTOR (MSB) (SLOT2_DESC1) | 32 | RW | 0000_0000h |
| 38h | SLOT 3 DESCRIPTOR (LSB) (SLOT3_DESC0) | 32 | RW | 0000_0000h |
| 3Ch | SLOT 3 DESCRIPTOR (MSB) (SLOT3_DESC1) | 32 | RW | 0000_0000h |

## A.2.5.2 WAKE-ON-RADIO CONTROL REGISTER (WOR_CTRL)

### A.2.5.2.1 Offset

| Register | Offset |
|---|---|
| WOR_CTRL | 0h |

### A.2.5.2.2 Diagram



### A.2.5.2.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>WOR_DEBUG_REG | WAKE-ON-RADIO Debug Register Enable<br>WOR_DEBUG_REG alters the readback values of several read-only registers, for debug purposes. The registers which have a different readback value when WOR_DEBUG_REG=1 are:<br>• TIMESTAMP1<br>• TIMESTAMP2<br>• TIMESTAMP3<br>See the description of those registers, which indicates their nominal, and debug-mode readback values. |
| 24<br>WOR_RESUME | WAKE-ON-RADIO Resume<br>This bit causes the WOR state machine to advance from its WAIT_RESUME state. The state machine will enter WAIT_RESUME on any slot in which an MCU wakeup occurs (based on descriptor WAKE_ON_COMPLETE and/or WAKE_ON_TIMEOUT fields). After waking up (interrupting) the MCU, WOR will hold in WAIT_RESUME, giving software whatever time it needs to complete all required activity. Software then sets WOR_RESUME=1, causing the state machine to break out of WAIT_RESUME, advance to the next slot, and enter DSM. This bit is write-only. Writes of '0' have no effect. Readback value is always 0. Setting WOR_RESUME has no effect if WOR state machine is not in WAIT_RESUME.<br>1: Causes WOR state machine to advance out of WAIT_RESUME state, and begin the next slot<br>0: No effect. |
| 23-20<br>— | Reserved |
| 19-16<br>DSM_GUARDBAND | WAKE-ON-RADIO DSM Guardband<br>Determines the number of 32.768KHz clock cycles of guardband used by the WOR state machine in computing duration of the DSM for any slot. Increasing this setting by 1 LSB causes WOR to shorten the DSM duration by 1 32.768KHz clock cycle. This guardband takes into account synchronization, and other, delays within the RSIM, and ensures that DSM will wakeup will never occur "too late" with respect to the slot descriptor's START_TIME field. The current default setting is 7. Values below 7 should not be used. |
| 15-8<br>— | Reserved |
| 7<br>SKIP_FIRST_DSM | WAKE-ON-RADIO Skip DSM On First Slot<br>Setting this bit causes the WOR State Machine to skip the DSM entry/exit cycle for the first SLOT (SLOT 0) encountered after setting SLOTS_USED > 0, regardless of the setting of the NO_DSM field of the slot's descriptor. Subsequent slots, including SLOT 0 when it is re-encountered, will enter/exit DSM based on the descriptor NO_DSM. The SKIP_FIRST_DSM feature is useful, especially in Constant Interval Scheduling mode, when the desire is to begin RF operations immediately (at the start of the first EVENT_INTERVAL), as programmed in the descriptor START_TIME, without waiting out a DSM cycle<br>1: WOR skips DSM entry/exit on the first SLOT 0 after setting SLOTS_USED > 0, regardless of descriptor NO_DSM<br>0: WOR always obeys descriptor NO_DSM<br>**Note:** SKIP_FIRST_DSM should not be used if the SLOT 0 descriptor FUNCTION is set to "NO_RF". |
| 6-4<br>SLOTS_USED | WAKE-ON-RADIO Number Of Slots Used |

*Table continues on the next page...*

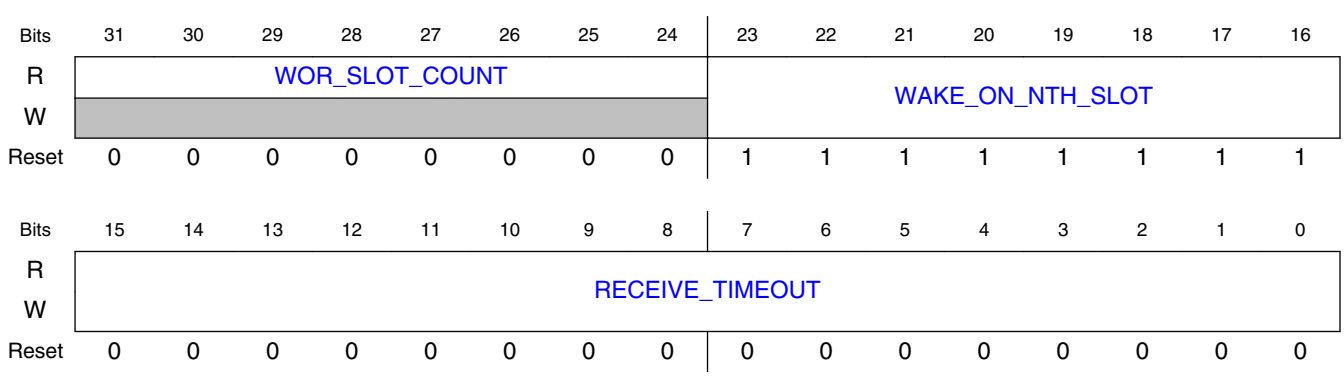
| Field | Function |
|---|---|
| | Determines the number of timeslots used to schedule future DSM and RF operations, for either scheduling mode. Up to 4 timeslots can be enabled. Increasing the number of timeslots used decreases average power consumption. Valid values are 0 ≤ SLOTS_USED ≤ 4. The SLOTS_USED register is the mechanism by which the WOR state machine is activated and de-activated. Setting SLOTS_USED to 0 disables the WOR state machine and puts the WOR functionality in an IDLE state. A subsequent write of a non-zero value to SL0TS_USED activates the state machine, initializing to SLOT 0, and starts a new WOR "session". |
| 3-2<br>WOR_PROTOCOL | WAKE-ON-RADIO Protocol Selector<br>Assigns the Wake-On-Radio function to one of the radio's protocol engines.<br>0: Not assigned<br>1: Assigned to 802.15.4<br>2: Assigned to GENERIC_FSK<br>3: Reserved |
| 1<br>SCHEDULING_MODE | WAKE-ON-RADIO Scheduling Mode<br>Determines the scheduling mode used by the WOR HW to organize future DSM and RF operations.<br>1: Constant-Interval Scheduling mode.<br>0: Timeslot Scheduling mode. |
| 0<br>WOR_EN | WAKE-ON-RADIO Enable<br>Wake-On-Radio Master Enable. In addition to enabling the block function, this bit gates the clock to the WOR module. This bit should be set, and remain set, whenever WOR is in use.<br>1: Wake-On-Radio is enabled.<br>0: Wake-On-Radio is disabled, and clocks to the module are gated off. |

## A.2.5.3 WAKE-ON-RADIO TIMEOUT REGISTER (WOR_TIMEOUT)

### A.2.5.3.1 Offset

| Register | Offset |
|---|---|
| WOR_TIMEOUT | 4h |

### A.2.5.3.2 Diagram



#### A.2.5.3.3 Fields

| Field | Function |
|---|---|
| 31-24<br>WOR_SLOT_COUNT | WAKE-ON-RADIO Absolute Slot Count<br>This read-only register indicates the number of slots that have elapsed since SLOTS_USED was set to a non-zero value to activate the WOR state machine. WOR_SLOT_COUNT can be considered an Absolute Slot Number. WOR_SLOT_COUNT counts slots up to a maximum of 255. If the number of elapsed slots exceeds 255, WOR_SLOT_COUNT will remain at 255. WOR_SLOT_COUNT is "sticky" in the sense that it will hold its value after the WOR session has ended by setting SLOTS_USED=0; WOR_SLOT_COUNT will clear to 0 when a new WOR session is started by taking SLOTS_USED from 0 to any non-zero value. |
| 23-16<br>WAKE_ON_NTH_SLOT | WAKE-ON-RADIO Force Wake On Nth Slot<br>If the WAKE_ON_COMPLETE bit is not set in any descriptor, WOR will not awaken the MCU even after a number of good packets are received. Additionally, if WAKE_ON_TIMEOUT is also not set in any descriptor, WOR will not wake up the MCU under any circumstances, which would not be desirable. This register provides a backup method to awaken the MCU, after *N+1* slots have elapsed, where *N* = WAKE_ON_NTH_SLOT[7:0], regardless of whether or not any good packets have been received, and regardless of the setting of the WAKE_ON_* descriptor bits. Applies to Constant-Interval Mode only. (For Timeslot mode, at least one slot *must* have a WAKE_ON_* bit set in its descriptor). |
| 15-0<br>RECEIVE_TIMEOUT | WAKE-ON-RADIO Receive Timeout<br>For any RX slot, determines how long RX stays on after the descriptor's START_TIME is reached, in microseconds. A timeout is declared if a good packet is not received before the expiration of this time period (START_TIME + RECEIVE_TIMEOUT + WINDOW_WIDENING). Applies to TIMESLOT mode only. |

### A.2.5.4 WAKE-ON-RADIO TIMESTAMP 1 (TIMESTAMP1)

#### A.2.5.4.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP1 | 8h |

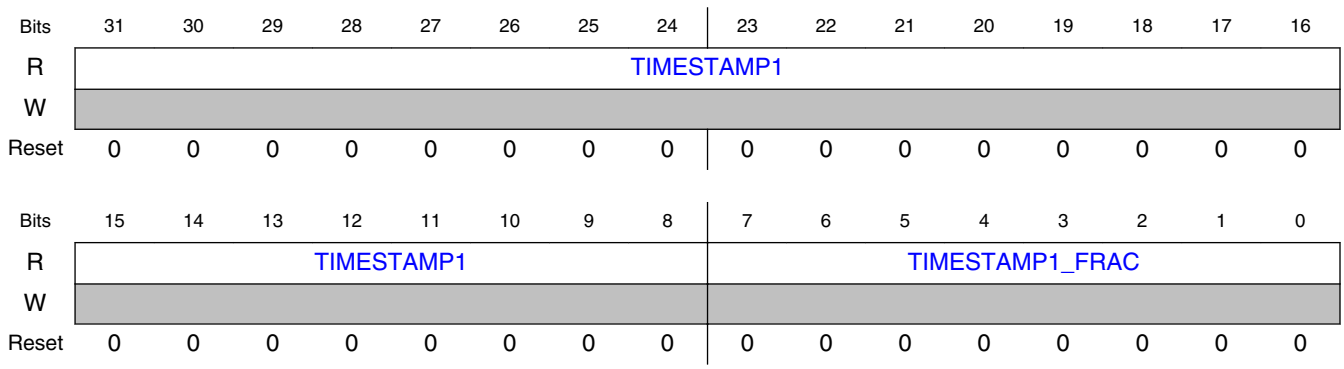
#### A.2.5.4.2 Function

Timestamp captured for a packet received on SLOT 1. During SLOT 1, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP1 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 1.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WOR_ADDBACK[27:0], which is the number of microseconds that the WOR state machine "added back" to the assigned Link Layer's Event Timer, to correct for the amount of time that Link Layer was in DSM (Event Timer frozen). The WOR_ADDBACK readback value applies to the most recently-completed DSM cycle.

### A.2.5.4.3 Diagram




### A.2.5.4.4 Fields

| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP1 | WAKE-ON-RADIO TIMESTAMP1<br>Timestamp captured for a packet received on SLOT 1. During SLOT 1, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP1 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 1. |
| 7-0<br>TIMESTAMP1_FRAC | WAKE-ON-RADIO TIMESTAMP 1 Fractional<br>TIMESTAMP1_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP1_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional component has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP1_FRAC[3:0], shall always read back '0000'. |

## A.2.5.5 WAKE-ON-RADIO TIMESTAMP 2 (TIMESTAMP2)

### A.2.5.5.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP2 | Ch |

### A.2.5.5.2 Function

Timestamp captured for a packet received on SLOT 2. During SLOT 2, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP2 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 2.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WW_DSM_ADJUST[19:0], which is the accumulated DSM component of the Window Widening adjustment value that is used to offset the start and end of receive operations during the timeslot.

### A.2.5.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP2 | | | | | | | | TIMESTAMP2_FRAC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.5.5.4 Fields

| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP2 | WAKE-ON-RADIO TIMESTAMP2 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Timestamp captured for a packet received on SLOT 2. During SLOT 2, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP2 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 2. |
| 7-0<br>TIMESTAMP2_FRAC | WAKE-ON-RADIO TIMESTAMP 2 Fractional<br>TIMESTAMP2_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP2_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP2_FRAC[3:0], shall always read back '0000'. |

## A.2.5.6 WAKE-ON-RADIO TIMESTAMP 3 (TIMESTAMP3)

### A.2.5.6.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP3 | 10h |

### A.2.5.6.2 Function

Timestamp captured for a packet received on SLOT 3. During SLOT 3, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP3 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 3.

**Note:** Timestamp for SLOT 0 is captured into the respective Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space.

**Note:** If WOR_DEBUG=1, this register instead reads back WW_RUN_ADJUST[19:0], which is the accumulated "run time" component of the Window Widening adjustment value that is used to offset the start and end of receive operations during the timeslot.

### A.2.5.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP3 | | | | | | | | TIMESTAMP3_FRAC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.5.6.4 Fields

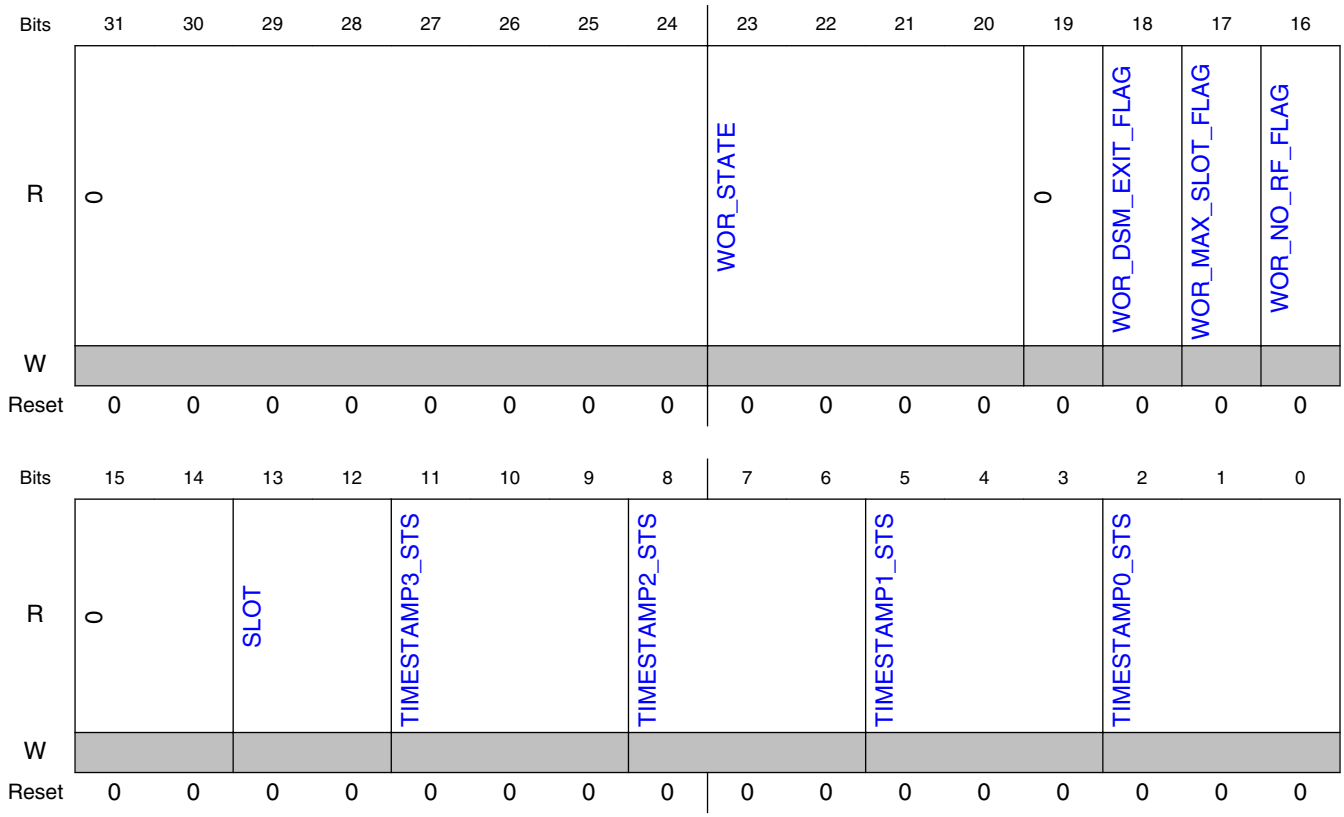
| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP3 | WAKE-ON-RADIO TIMESTAMP3<br>Timestamp captured for a packet received on SLOT 3. During SLOT 3, any valid Access Address detection (for protocols other than 802.15.4), or SFD detection (for 802.15.4) will result in the contents of the respective Link Layer's Event Timer being captured into this register at the instant of detection (there is no offset applied). TIMESTAMP3 remains "sticky" until a new AA or SFD detection occurs, during either the same, or a subsequent, SLOT 3. |
| 7-0<br>TIMESTAMP3_FRAC | WAKE-ON-RADIO TIMESTAMP 3 Fractional<br>TIMESTAMP3_FRAC[7:4] applies to 802.15.4 only. For other protocols, this field can be ignored. For 802.15.4, the captured timestamp represents a 28-bit number with 24 Integer bits and 4 Fractional bits. TIMESTAMP3_FRAC[7:4] is the fractional component, which appears to the right of the decimal point. The fractional has been provided to allow for microsecond accuracy in capturing packet reception instants. The lower 4 bits, TIMESTAMP3_FRAC[3:0], shall always read back '0000'. |

## A.2.5.7 WAKE-ON-RADIO STATUS REGISTER (WOR_STATUS)

### A.2.5.7.1 Offset

| Register | Offset |
|---|---|
| WOR_STATUS | 14h |

## A.2.5.7.2 Diagram




## A.2.5.7.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20<br>WOR_STATE | WAKE-ON-RADIO Current State<br>This read-only register reflects the instantaneous state of the Wake-On-Radio state vector. A listing of the Wake-On-Radio states, along with the associated state vector, is below:<br>• DISABLED (0)<br>• COMP_DURATION1 (1)<br>• COMP_DURATION2 (2)<br>• SLEEP_REQ (3)<br>• IN_DSM (4)<br>• COMP_ADDBACK (5)<br>• COMP_WW (6)<br>• WAIT_SEQ_START (7)<br>• WAIT_SEQ_END (8)<br>• WAIT_RESUME (9) |
| 19<br>— | Reserved |
| 18 | WAKE-ON-RADIO Early DSM Exit Flag |

*Table continues on the next page...*

| Field | Function |
|---|---|
| WOR_DSM_EXIT_FLAG | This read-only flag indicates that, in the current slot, an event external to the radio has forced an early exit from DSM, before the WOR-computed DSM duration has expired. The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to an "DSM Early Exit wakeup", as described by the conditions above. The WOR_IRQ will become set, and the MCU awakened, even if the slot descriptor's WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT bits are both set to 0. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "DSM Early Exit wakeup", software must set WOR_RESUME to advance to the next slot. |
| 17<br>WOR_MAX_SLOT_FLAG | WAKE-ON-RADIO Maximum Slot Count Reached Flag<br>This read-only flag indicates that, in the current slot, the following conditions have all been met:<br>1. SCHEDULING_MODE is set to 1 (Constant Interval)<br>2. WOR_SLOT_COUNT[7:0] (absolute slot number) has matched WAKE_ON_NTH_SLOT[7:0]<br><br>The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to a "Maximum Slot Reached wakeup", as described by the conditions above. The WOR_IRQ will become set, and the MCU awakened, even if the slot descriptor's WAKE_ON_COMPLETE and WAKE_ON_TIMEOUT bits are both set to 0. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "Maximum Slot Reached wakeup", software must set WOR_RESUME to advance to the next slot |
| 16<br>WOR_NO_RF_FLAG | WAKE-ON-RADIO NO_RF Slot Flag<br>This read-only flag indicates that, in the current slot, the following conditions have all been met:<br>1. The slot's descriptor FUNCTION is set to "NO_RF"<br>2. A DSM cycle has just completed (exited) during the slot<br>3. The slot's descriptor WAKE_ON_COMPLETE bit was set (or the WOR_SLOT_COUNT matched WAKE_ON_NTH_SLOT), requiring an MCU wakeup.<br><br>The WOR_IRQ (WAKE_IRQ) interrupt inside the assigned Link Layer will become set under these conditions, and this flag indicates that the reason for WOR_IRQ was due to a "NO_RF wakeup", as described by the conditions above. The WOR state machine is holding in the WAIT_RESUME state, and once software has completed all activities pursuant to the "NO_RF wakeup", software must set WOR_RESUME to advance to the next slot |
| 15-14<br>— | Reserved |
| 13-12<br>SLOT | WAKE-ON-RADIO Current Slot<br>This read-only register reflects the instantaneous state of the WOR's SLOT counter, indicating which SLOT is currently in service by the WOR state machine. Software can determine which of the 4 descriptors is currently in effect based on this register. |
| 11-9<br>TIMESTAMP3_STS | WAKE-ON-RADIO Timestamp 3 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP3 register contents:<br><br>**TIMESTAMP3_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 3 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP3 register.<br>**TIMESTAMP3_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 3 RX operation<br>**TIMESTAMP3_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 3 RX operation |
| 8-6<br>TIMESTAMP2_STS | WAKE-ON-RADIO Timestamp 2 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP2 register contents: |

*Table continues on the next page...*

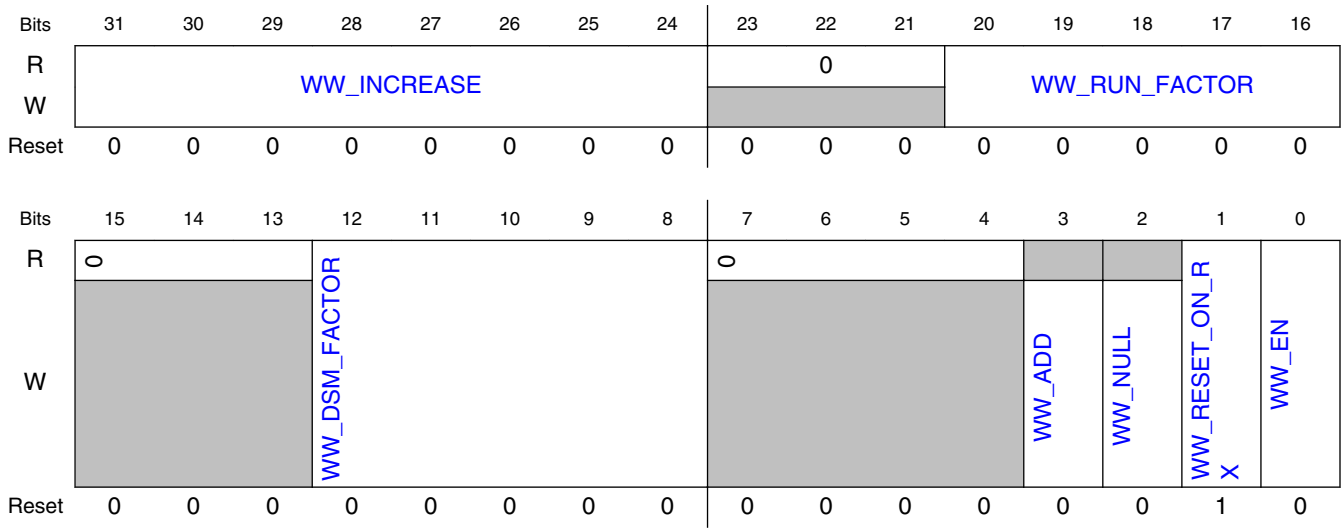
| Field | Function |
|---|---|
| | **TIMESTAMP2_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 2 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP2 register.<br>**TIMESTAMP2_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 2 RX operation<br>**TIMESTAMP2_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 2 RX operation |
| 5-3<br>TIMESTAMP1_STS | WAKE-ON-RADIO Timestamp 1 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP1 register contents:<br>**TIMESTAMP1_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 1 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP1 register.<br>**TIMESTAMP1_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 1 RX operation<br>**TIMESTAMP1_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 1 RX operation |
| 2-0<br>TIMESTAMP0_STS | WAKE-ON-RADIO Timestamp 0 Status<br>This read-only register comprises 3 bits which convey information about the validity of the TIMESTAMP0 register contents:<br>**TIMESTAMP0_STS[0]** -- "AA/SFD Status". If set, this bit indicates at least 1 AA or SFD detection occured during the current, or most recent, SLOT 0 RX operation, and therefore a timestamp, valid or not, was captured into the TIMESTAMP0 register.<br>**TIMESTAMP0_STS[1]** -- "CRC Status". If set, this bit indicates a packet was received which passed CRC verification during the current, or most recent, SLOT 0 RX operation<br>**TIMESTAMP0_STS[2]** -- "Filter Status". If set, this bit indicates a packet was received which passed Packet Filtering rules during the current, or most recent, SLOT 0 RX operation<br>**Note:** TIMESTAMP0 register (timestamp for SLOT 0) is captured into the assigned Link Layer's TIMESTAMP register (external to WOR), and can be read from that Link Layer's address space. However TIMESTAMP0_STS[2:0], which applies to that register, resides in WOR (XCVR) address space. |

## A.2.5.8 WINDOW-WIDENING CONTROL REGISTER (WW_CTRL)

### A.2.5.8.1 Offset

| Register | Offset |
|---|---|
| WW_CTRL | 18h |

## A.2.5.8.2 Diagram




## A.2.5.8.3 Fields

| Field | Function |
|---|---|
| 31-24<br>WW_INCREASE | Window-widening Manual Increase Amount<br>This register allows software to increase (add to) the HW-computed window-widening adjustment. This incremental amount is added whenever a 1 is written to the WW_ADD register bit. The units of WW_INCREASE[7:0] are microseconds. It is an unsigned (positive only) addition. This is a one-time only addition (whever WW_ADD is set), and the window-widening adjustment value continues to accumulate after the increment. The capability to add to the HW-computed adjustment is intended for contingency or debug purposes only, as the HW-computed value should be accurate. |
| 23-21<br>— | Reserved |
| 20-16<br>WW_RUN_FACTOR | Window-widening Runtime Factor<br>Determines the rate of Window-widening that applies during RUNTIME (Reference Oscillator On). This setting should be based on the combined Sleep Clock Accuracy (SCA) of the Master + Slave. Each LSB is worth 64ppm of SCA drift. Valid values are 0 -> 16. The lower this factor, the greater the power savings. |
| 15-13<br>— | Reserved |
| 12-8<br>WW_DSM_FACTOR | Window-widening DSM Factor<br>WW_DSM_FACTOR[4:0]: Determines the rate of Window-widening that applies during DSM (Reference Oscillator Off). This setting should be based on the combined Sleep Clock Accuracy (SCA) of the Master + Slave. Each LSB is worth 64ppm of combined SCA drift. Valid values are 0 -> 16. The lower this factor, the greater the power savings. |
| 7-4<br>— | Reserved |
| 3<br>WW_ADD | Window-widening Add Command |

*Table continues on the next page...*

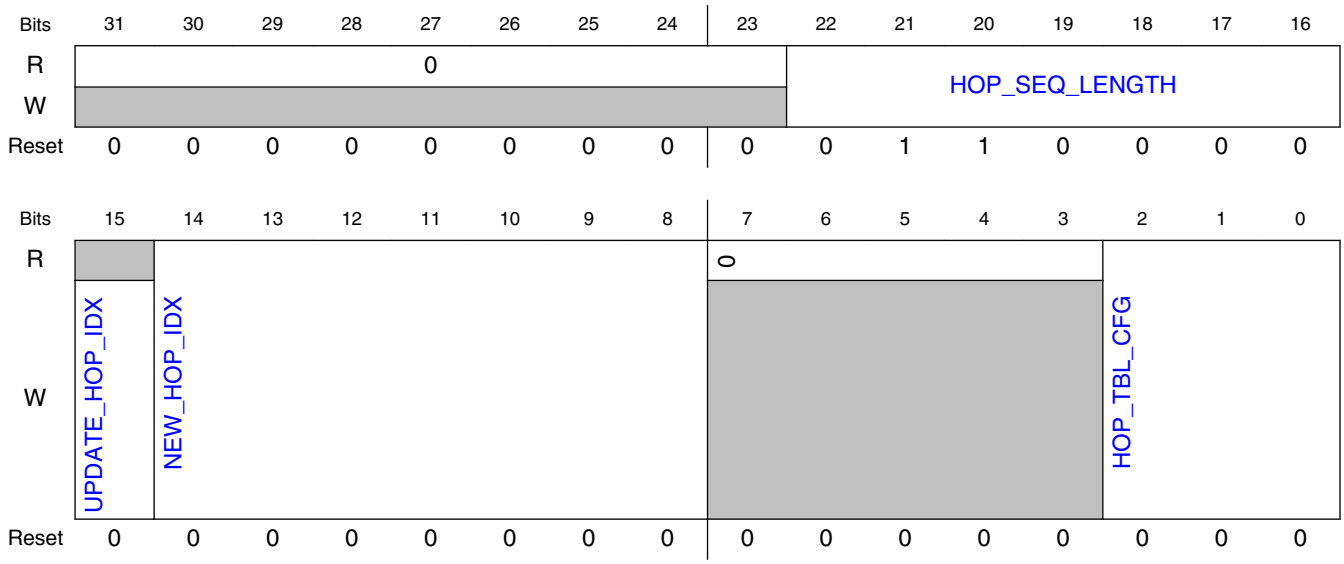
| Field | Function |
|---|---|
| | This bit allows software to increase (add to) the HW-computed window-widening adjustment. The incremental amount is determined by the WW_INCREASE[7:0] register. The capability to add to the HW-computed adjustment is intended for contingency or debug purposes only, as the HW-computed value should be accurate. This bit is write-only. Readback value is always 0.<br>1: Increase the HW-computed window-widening adjustment by amount WW_INCREASE[7:0].<br>0: No effect |
| 2<br>WW_NULL | Window-widening Null Command<br>This bit allows software to reset the accumulated window-widening adjustment to 0. This bit is write-only. Readback value is always 0.<br>1: Reset the accumulated window-widening adjustment to 0.<br>0: No effect. |
| 1<br>WW_RESET_ON_RX | Window-widening Reset on Received Good Packet<br>Enable a good packet reception to reset the accumulated window-widening adjustment to 0. A good packet is one that passes CRC verification, and passes packet filtering rules. Filtering rules vary by protocol.<br>1: Allow good packet reception to reset window-widening adjustment. This is the default.<br>0: Block good packet reception from resetting window-widening adjustment. |
| 0<br>WW_EN | Window-widening Enable<br>Master Enable for Window Widening. Typically, only slave devices employ window widening.<br>1: Window Widening is enabled.<br>0: Window Widening is disabled. |

## A.2.5.9 FREQUENCY HOP CONTROL REGISTER (HOP_CTRL)

### A.2.5.9.1 Offset

| Register | Offset |
|---|---|
| HOP_CTRL | 1Ch |

## A.2.5.9.2 Diagram




## A.2.5.9.3 Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22-16<br>HOP_SEQ_LENGTH | New Hop Table Index<br>The HOP_SEQ_LENGTH[6:0] register determines the number of entries in the WOR Hop Table, regardless of how the table is configured by HOP_TBL_CFG[2:0]. During WOR operation, once the Hop Table Index reaches HOP_SEQ_LENGTH - 1, the next Hop Table Index will wrap to 0. HOP_SEQ_LENGTH must not exceed the table size specified by HOP_TBL_CFG (48 or 96 entries). |
| 15<br>UPDATE_HOP_IDX | Update Hop Table Index<br>Software can jump to a new index (read pointer address) of the WOR Hop Table at any time by writing the desired index to the NEW_HOP_IDX[6:0] register, and setting the UPDATE_HOP_IDX bit to 1. This bit is write-only. Readback value is always 0.<br>1: Jump to new Hop Table Index pointed to by NEW_HOP_IDX[6:0]<br>0: No effect |
| 14-8<br>NEW_HOP_IDX | New Hop Table Index<br>Software can jump to a new index (read pointer address) of the WOR Hop Table at any time by writing the desired index to this register, and setting the UPDATE_HOP_IDX bit. |
| 7-3<br>— | Reserved |
| 2-0<br>HOP_TBL_CFG | Hop Table Configuration<br>For maximum flexibility in Frequency Hop scheduling, 4 hop table configurations are defined. The configurations affect:<br>1. The source of the RF channel information (Link Layer register or WOR Hop Table)<br>2. How the RF channel information is organized in the WOR Hop Table<br>3. How the hop frequency word is interpreted by the PLL Digital Block |

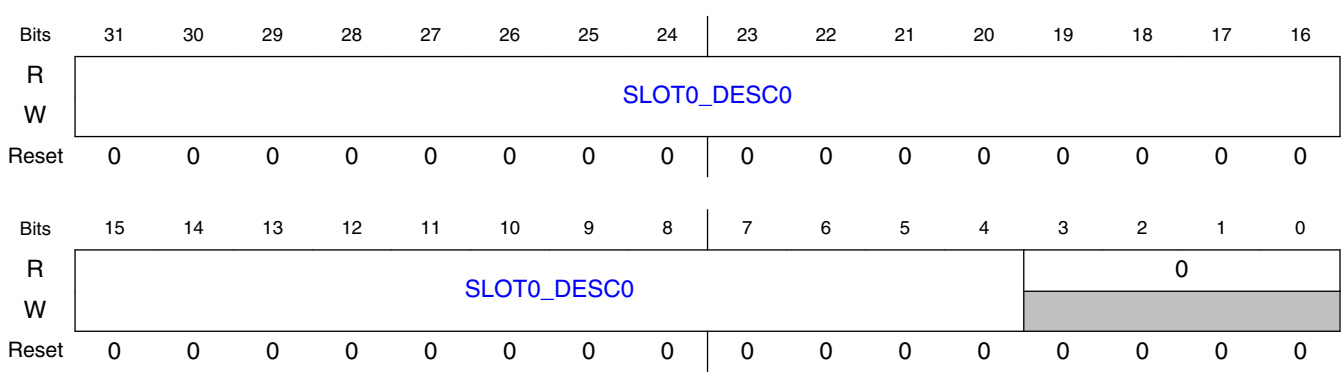
| Field | Function |
|---|---|
| | Source of RF channel information:<br>0: Link Layer CHANNEL_NUM register (legacy configuration). The Link Layer is the one assigned by WOR_PROTOCOL register.<br>1: WOR Hop Table<br>2: WOR Hop Table<br>3: WOR Hop Table<br>4-7: Reserved<br><br>Hop Table Organization:<br>0: N/A<br>1: Table size is 96 entries. Two 7-bit CHANNEL_NUM fields are packed into a 16-bit hop frequency word for RAM storage<br>2-3: Table size is 48 entries. Each 16-bit hop frequency word in RAM contains a specification for a single RF channel<br>4-7: Reserved<br><br>PLL Digital Interpretation of Hop Frequency Word:<br>0: Lower 7 bits of hop frequency word contain a 7-bit mapped channel (legacy format).<br>1: Lower 7 bits of hop frequency word contain a 7-bit mapped channel (legacy format).<br>2-3: See Chapter *PLL Digital*<br>4-7: Reserved |

# A.2.5.10 SLOT 0 DESCRIPTOR (LSB) (SLOT0_DESC0)

## A.2.5.10.1 Offset

| Register | Offset |
|---|---|
| SLOT0_DESC0 | 20h |

## A.2.5.10.2 Diagram



### A.2.5.10.3 Fields

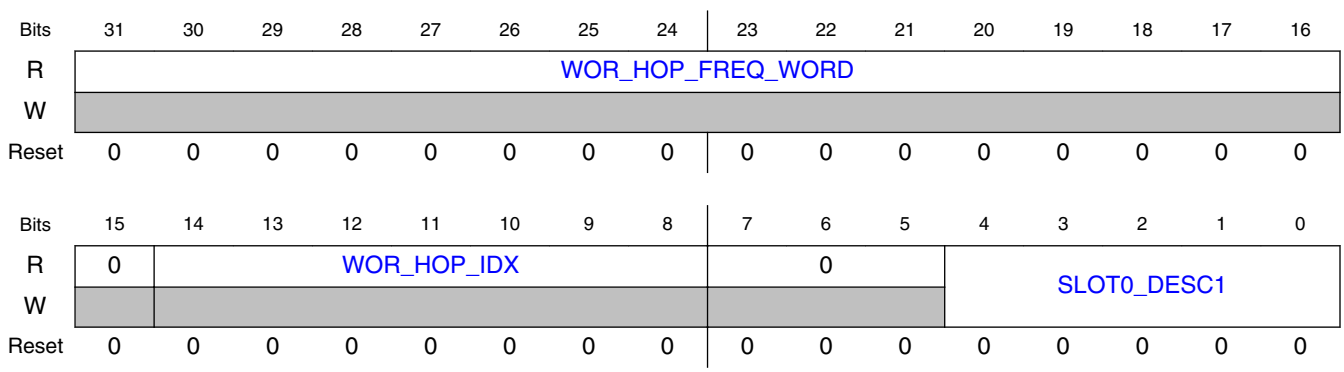
| Field | Function |
|---|---|
| 31-4<br>SLOT0_DESC0 | Slot 0 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 0 |
| 3-0<br>— | Reserved |

## A.2.5.11 SLOT 0 DESCRIPTOR (MSB) (SLOT0_DESC1)

### A.2.5.11.1 Offset

| Register | Offset |
|---|---|
| SLOT0_DESC1 | 24h |

### A.2.5.11.2 Diagram




### A.2.5.11.3 Fields

| Field | Function |
|---|---|
| 31-16<br>WOR_HOP_FREQ_WORD | Current Hop Frequency Word<br>This read-only register reflects the current Hop Frequency Word, formulated by WOR and transmitted to the PLL Digital Block |
| 15<br>— | Reserved |
| 14-8 | Current Hop Table Index |

*Table continues on the next page...*

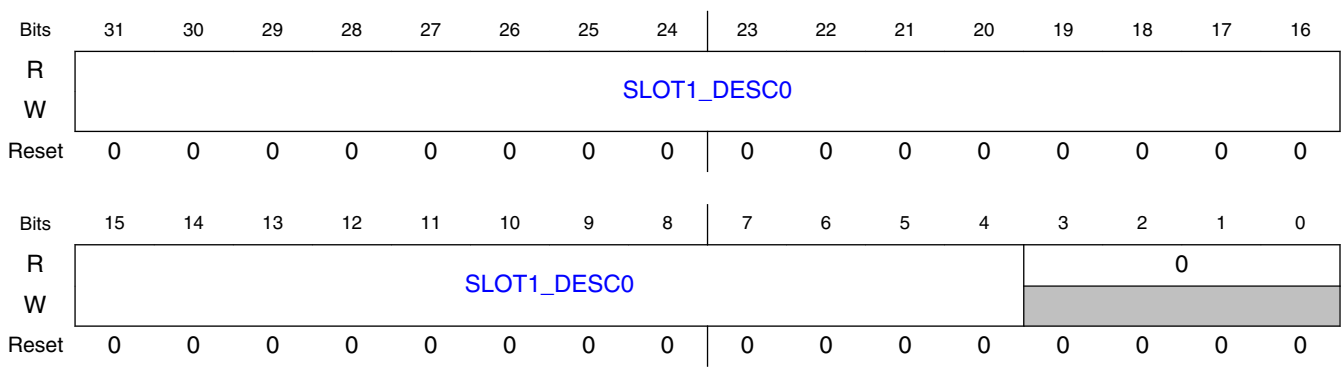
| Field | Function |
|---|---|
| WOR_HOP_IDX | This read-only register reflects the current Hop Table Index (read address pointer) into the WOR Hop Table |
| 7-5<br>— | Reserved |
| 4-0<br>SLOT0_DESC1 | Slot 0 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 0 |

## A.2.5.12 SLOT 1 DESCRIPTOR (LSB) (SLOT1_DESC0)

### A.2.5.12.1 Offset

| Register | Offset |
|---|---|
| SLOT1_DESC0 | 28h |

### A.2.5.12.2 Diagram




### A.2.5.12.3 Fields

| Field | Function |
|---|---|
| 31-4<br>SLOT1_DESC0 | Slot 1 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 1 |
| 3-0<br>— | Reserved |

# A.2.5.13 SLOT 1 DESCRIPTOR (MSB) (SLOT1_DESC1)

## A.2.5.13.1 Offset

| Register | Offset |
|---|---|
| SLOT1_DESC1 | 2Ch |

## A.2.5.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | SLOT1_DESC1 | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.5.13.3 Fields

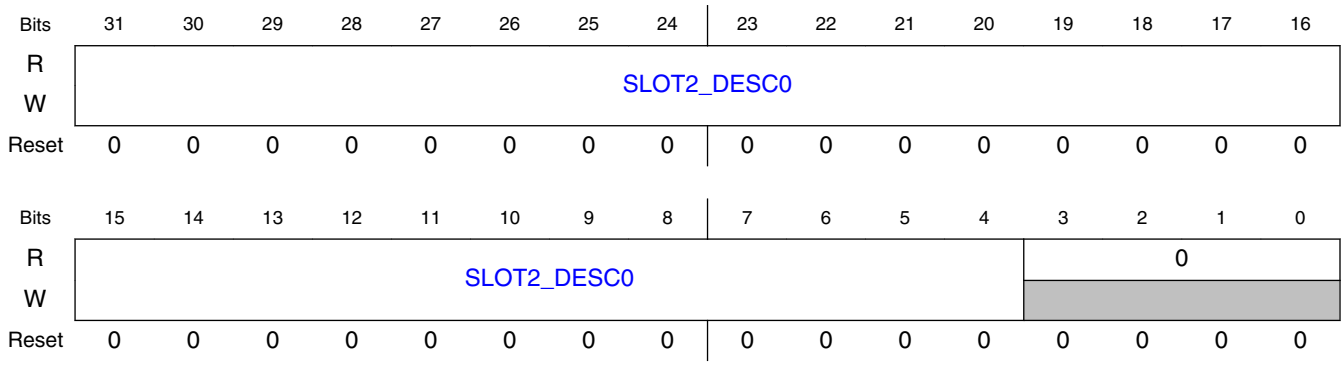
| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT1_DESC1 | Slot 1 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 1 |

# A.2.5.14 SLOT 2 DESCRIPTOR (LSB) (SLOT2_DESC0)

## A.2.5.14.1 Offset

| Register | Offset |
|---|---|
| SLOT2_DESC0 | 30h |

### A.2.5.14.2 Diagram




### A.2.5.14.3 Fields

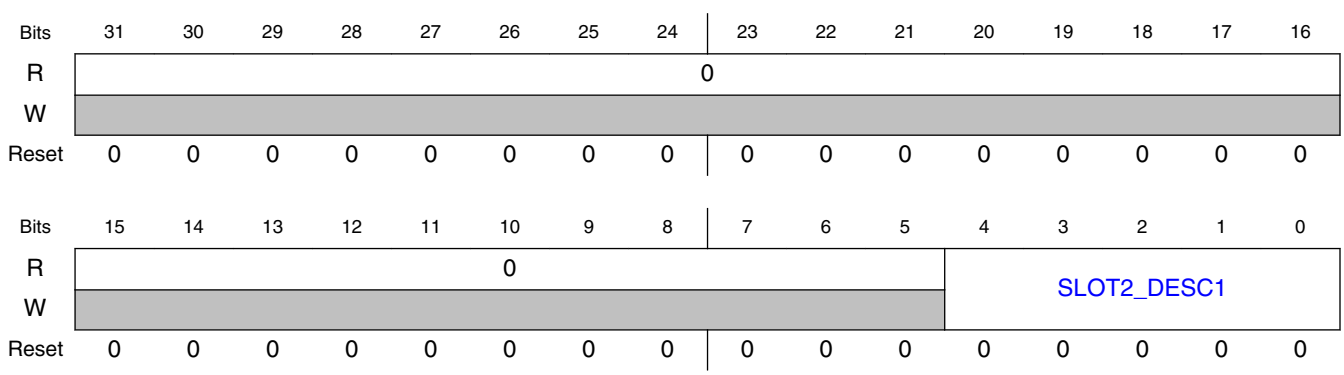
| Field | Function |
|---|---|
| 31-4<br>SLOT2_DESC0 | Slot 2 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 2 |
| 3-0<br>— | Reserved |

## A.2.5.15 SLOT 2 DESCRIPTOR (MSB) (SLOT2_DESC1)

### A.2.5.15.1 Offset

| Register | Offset |
|---|---|
| SLOT2_DESC1 | 34h |

### A.2.5.15.2 Diagram



### A.2.5.15.3 Fields

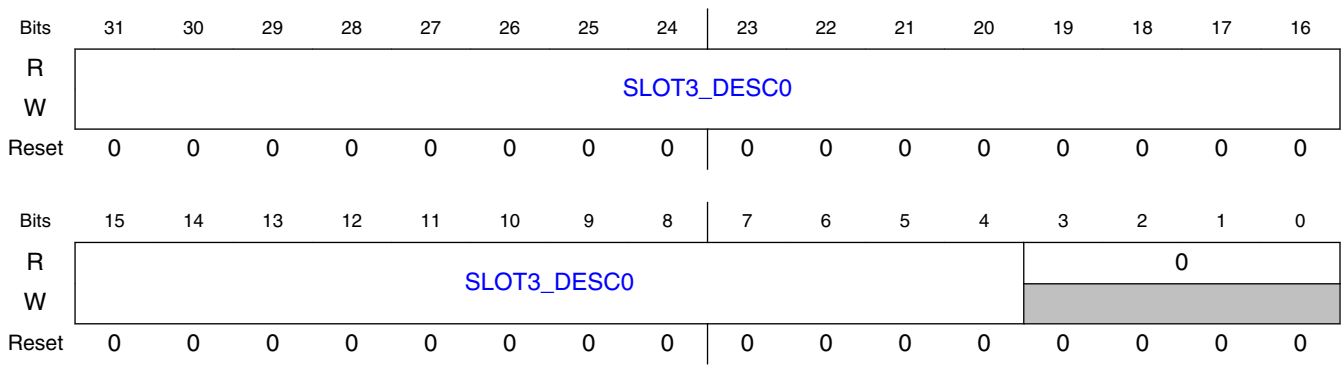
| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT2_DESC1 | Slot 2 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 2 |

## A.2.5.16 SLOT 3 DESCRIPTOR (LSB) (SLOT3_DESC0)

### A.2.5.16.1 Offset

| Register | Offset |
|---|---|
| SLOT3_DESC0 | 38h |

### A.2.5.16.2 Diagram




### A.2.5.16.3 Fields

| Field | Function |
|---|---|
| 31-4<br>SLOT3_DESC0 | Slot 3 Descriptor (LSB's)<br>This register represents the 28 LSB's of the WOR descriptor for Slot 3 |
| 3-0<br>— | Reserved |

# A.2.5.17 SLOT 3 DESCRIPTOR (MSB) (SLOT3_DESC1)

## A.2.5.17.1 Offset

| Register | Offset |
|---|---|
| SLOT3_DESC1 | 3Ch |

## A.2.5.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | SLOT3_DESC1 | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.2.5.17.3 Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>SLOT3_DESC1 | Slot 3 Descriptor (MSB's)<br>This register represents the 5 MSB's of the WOR descriptor for Slot 3 |

# A.2.6 XCVR_ANALOG register descriptions

## A.2.6.1 XCVR_ANALOG_ADDR Memory map

XCVR_ANA base address: 4103_0500h

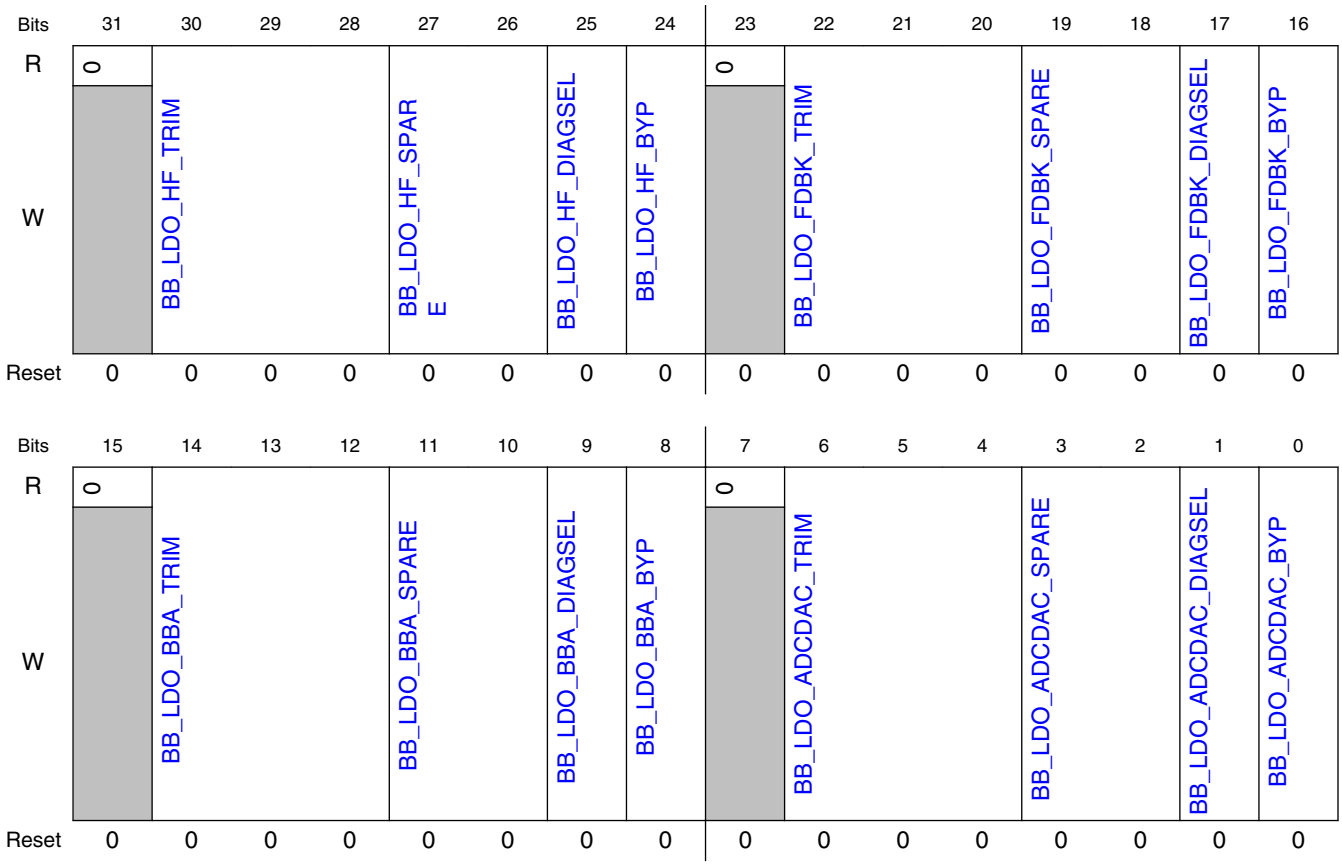
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | RF Analog Baseband LDO Control 1 (BB_LDO_1) | 32 | RW | 0000_0000h |
| 4h | RF Analog Baseband LDO Control 2 (BB_LDO_2) | 32 | RW | 0000_0000h |
| 8h | RF Analog ADC Control (RX_ADC) | 32 | RW | 0000_0140h |
| Ch | RF Analog BBA Control (RX_BBA) | 32 | RW | 0300_0003h |
| 10h | RF Analog LNA Control (RX_LNA) | 32 | RW | 0000_0000h |
| 14h | RF Analog TZA Control (RX_TZA) | 32 | RW | 0000_0003h |
| 18h | RF Analog Aux PLL Control (RX_AUXPLL) | 32 | RW | 0000_9002h |
| 1Ch | RF Analog Synthesizer Control 1 (SY_CTRL_1) | 32 | RW | 0000_0150h |
| 20h | RF Analog Synthesizer Control 2 (SY_CTRL_2) | 32 | RW | 0000_0014h |
| 24h | RF Analog TX HPM DAC and PA Control (TX_DAC_PA) | 32 | RW | 0002_0000h |
| 28h | RF Analog Balun TX Mode Control (BALUN_TX) | 32 | RW | 0072_4B6Dh |
| 2Ch | RF Analog Balun RX Mode Control (BALUN_RX) | 32 | RW | 0049_372Dh |
| 30h | RF Analog DFT Observation Register 1 (DFT_OBSV_1) | 32 | RO | 0000_0000h |
| 34h | RF Analog DFT Observation Register 2 (DFT_OBSV_2) | 32 | RW | 0000_0000h |
| 38h | RF Analog DFT Observation Register 3 (DFT_OBSV_3) | 32 | RW | 0000_FF00h |

## A.2.6.2 RF Analog Baseband LDO Control 1 (BB_LDO_1)

### A.2.6.2.1 Offset

| Register | Offset |
|---|---|
| BB_LDO_1 | 0h |

## A.2.6.2.2 Diagram




## A.2.6.2.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>BB_LDO_HF_TRIM | rmap_bb_ldo_hf_trim[2:0]<br>Trim settings for LDO.<br>000b - 1.20 V ( Default )<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 27-26<br>BB_LDO_HF_SPARE | rmap_bb_ldo_hf_spare[1:0]<br>Spare bits. Not used so far. |
| 25 | rmap_bb_ldo_hf_diagsel |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BB_LDO_HF_DIAGSEL | Diagsel for LDO.<br>0b - Diag disable<br>1b - Diag enable |
| 24<br>BB_LDO_HF_BYP | rmap_bb_ldo_hf_byp<br>Bypass signal for LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |
| 23<br>— | Reserved |
| 22-20<br>BB_LDO_FDBK_TRIM | rmap_bb_ldo_fdbk_trim[2:0]<br>Trim settings for LDO.(with 3.6/1.425V supply)<br>000b - 1.2/1.176 V ( Default )<br>001b - 1.138/1.115 V<br>010b - 1.085/1.066 V<br>011b - 1.04/1.025 V<br>100b - 1.28/1.25 V<br>101b - 1.4/1.35 V<br>110b - 1.55/1.4 V<br>111b - 1.78/1.4 V |
| 19-18<br>BB_LDO_FDBK_SPARE | rmap_bb_ldo_fdbk_spare[1:0]<br>Spare bits |
| 17<br>BB_LDO_FDBK_DIAGSEL | rmap_bb_ldo_fdbk_diagsel<br>Diagsel for DIVN/FCAL LDO.<br>0b - Diag disable<br>1b - Diag enable |
| 16<br>BB_LDO_FDBK_BYP | rmap_bb_ldo_fdbk_byp<br>Bypass signal for DIVN/FCAL LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |
| 15<br>— | Reserved |
| 14-12<br>BB_LDO_BBA_TRIM | rmap_bb_ldo_bba_trim[2:0]<br>Trim settings for LDO.<br>000b - 1.20 V ( Default )<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 11-10<br>BB_LDO_BBA_SPARE | rmap_bb_ldo_bba_spare[1:0]<br>Spare bits. Not used so far. |
| 9 | rmap_bb_ldo_bba_diagsel |

*Table continues on the next page...*

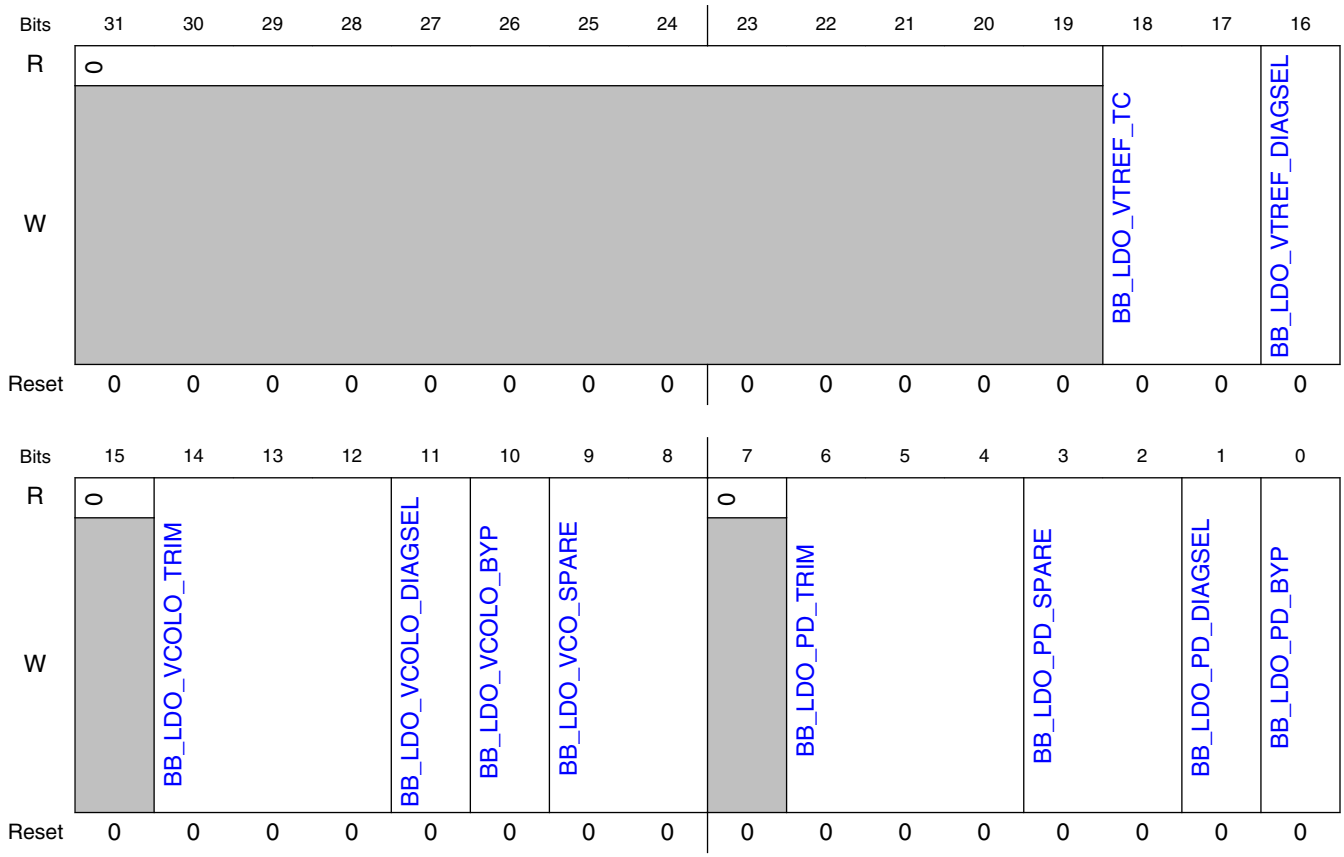
| Field | Function |
|---|---|
| BB_LDO_BBA_DIAGSEL | Diagsel for LDO.<br>0b - Diag disable<br>1b - Diag enable |
| 8<br>BB_LDO_BBA_BYP | rmap_bb_ldo_bba_byp<br>Bypass signal for LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |
| 7<br>— | Reserved |
| 6-4<br>BB_LDO_ADCDAC_TRIM | rmap_bb_ldo_adcdac_trim[2:0]<br>Trim settings for LDO.<br>000b - 1.20 V ( Default )<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 3-2<br>BB_LDO_ADCDAC_SPARE | rmap_bb_ldo_adcdac_spare[1:0]<br>Spare bits. Not used so far. |
| 1<br>BB_LDO_ADCDAC_DIAGSEL | rmap_bb_ldo_adcdac_diagsel<br>Diagsel for LDO.<br>0b - Diag disable<br>1b - Diag enable |
| 0<br>BB_LDO_ADCDAC_BYP | rmap_bb_ldo_adcdac_byp<br>Bypass signal for LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |

## A.2.6.3 RF Analog Baseband LDO Control 2 (BB_LDO_2)

### A.2.6.3.1 Offset

| Register | Offset |
|---|---|
| BB_LDO_2 | 4h |

## A.2.6.3.2 Diagram




## A.2.6.3.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-17<br>BB_LDO_VTREF_TC | rmap_bb_ldo_vtref_tc[1:0]<br>Changes Temperature coefficient of reference(vldo_vcolo,vldo_fdbk)<br>00b - 1.117/1.176 V<br>01b - 1.134/1.188 V<br>10b - 1.10/1.162 V<br>11b - 1.09/1.152 V |
| 16<br>BB_LDO_VTREF_DIAGSEL | rmap_bb_ldo_vtref_diagsel<br>Diagsel for Reference.<br>0b - Diag disable<br>1b - Diag enable |
| 15<br>— | Reserved |
| 14-12 | rmap_bb_ldo_vcolo_trim[2:0] |

*Table continues on the next page...*

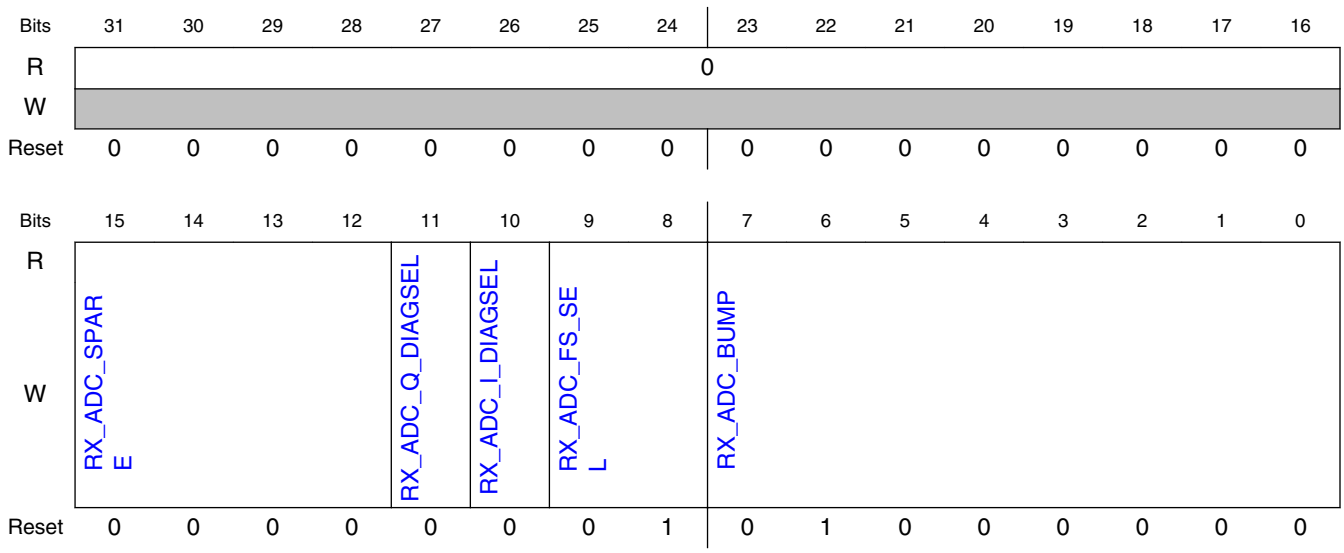
| Field | Function |
|---|---|
| BB_LDO_VCOLO_TRIM | Trim settings for LDO.(with 3.6/1.425V supply)<br>000b - 1.138/1.117 V ( Default )<br>001b - 1.076/1.058 V<br>010b - 1.027/1.012 V<br>011b - 0.98/0.97 V<br>100b - 1.22/1.19 V<br>101b - 1.33/1.3 V<br>110b - 1.5/1.4 V<br>111b - 1.82/1.4 V |
| 11<br>BB_LDO_VCOLO_DIAGSEL | rmap_bb_ldo_vcolo_diagsel<br>Diagsel for VCO/LO LDO.<br>0b - Diag disable<br>1b - Diag enable |
| 10<br>BB_LDO_VCOLO_BYP | rmap_bb_ldo_vcolo_byp<br>Bypass signal for VCO/LO LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |
| 9-8<br>BB_LDO_VCO_SPARE | rmap_bb_ldo_vco_spare[1:0]<br>Spare bits |
| 7<br>— | Reserved |
| 6-4<br>BB_LDO_PD_TRIM | rmap_bb_ldo_pd_trim[2:0]<br>Trim settings for LDO.<br>000b - 1.20 V ( Default )<br>001b - 1.25 V<br>010b - 1.28 V<br>011b - 1.33 V<br>100b - 1.40 V<br>101b - 1.44 V<br>110b - 1.50 V<br>111b - 1.66 V |
| 3-2<br>BB_LDO_PD_SPARE | rmap_bb_ldo_pd_spare[1:0]<br>Spare bits. Not used so far. |
| 1<br>BB_LDO_PD_DIAGSEL | rmap_bb_ldo_pd_diagsel<br>Diagsel for Reference.<br>0b - Diag disable<br>1b - Diag enable |
| 0<br>BB_LDO_PD_BYP | rmap_bb_ldo_pd_byp<br>Bypass signal for LDO. Vline comes out directly.<br>0b - Bypass disabled.<br>1b - Bypass enabled |

# A.2.6.4 RF Analog ADC Control (RX_ADC)

## A.2.6.4.1 Offset

| Register | Offset |
|---|---|
| RX_ADC | 8h |

## A.2.6.4.2 Diagram




## A.2.6.4.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>RX_ADC_SPARE | rmap_rx_adc_spare[3:0]<br>Not used |
| 11<br>RX_ADC_Q_DIAGSEL | rmap_rx_adc_q_diagsel<br>Enable diagnostics output for ADC Q |
| 10<br>RX_ADC_I_DIAGSEL | rmap_rx_adc_i_diagsel<br>Enable diagnostics output for ADC I |
| 9-8 | rmap_rx_adc_fs_sel[1:0]<br>Clock frequency selector |

*Table continues on the next page...*

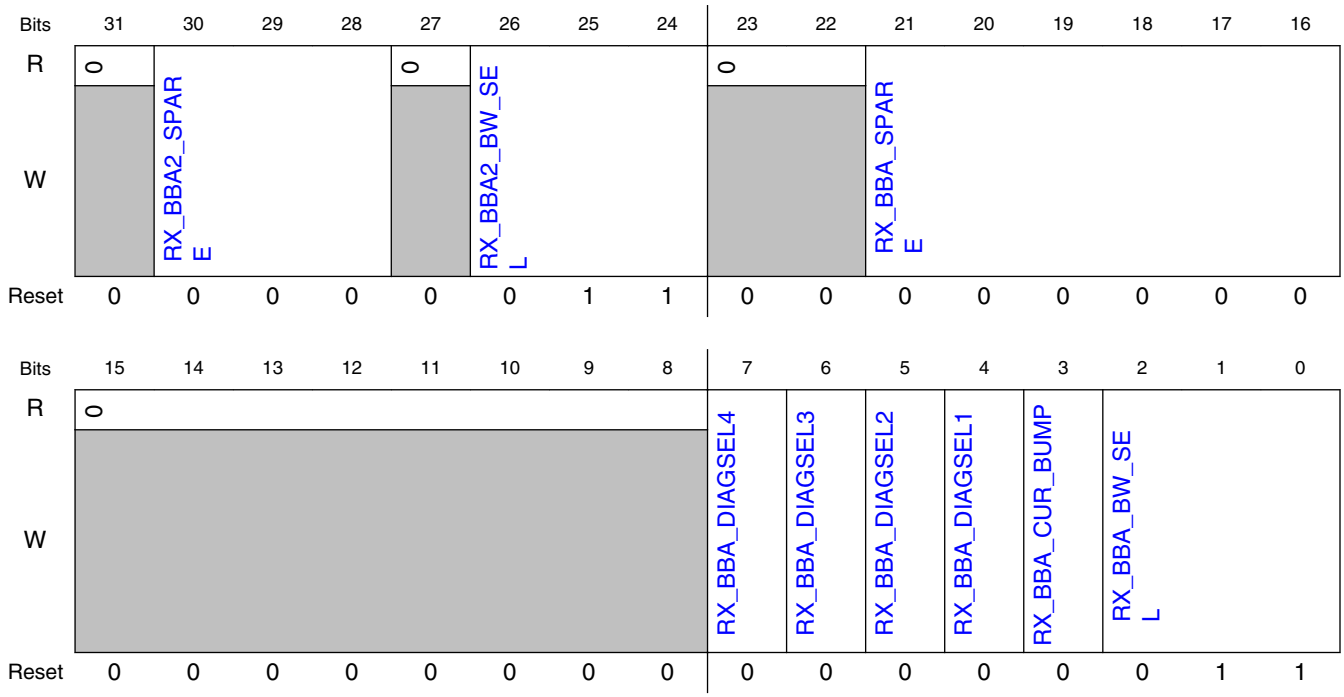
| Field | Function |
|---|---|
| RX_ADC_FS_SEL | 00b - 52MHz (2x26MHz)<br>01b - 64MHz (2x32MHz)<br>10b - +13% of 64MHz<br>11b - - 11% of 64MHz |
| 7-0<br>RX_ADC_BUMP | rmap_rx_adc_bump[7:0]<br>Bump bits for both ADCs :<br>bits [7:6] bump all opamps bias current<br>00 : 0.75x<br>01 : 1.00x<br>10 : 1.25x<br>11 : 1.50x<br>bit [5] bump dac opamp bias current.<br>0 : 1.0x<br>1 : 1.5x<br>bit [4] bump dac vrefn<br>0 : 0.0V<br>1 : 0.1V<br>bit [3] bump opamp CM output voltage<br>0 : vcmo = 0.63V<br>1 : vcmo = 0.57V<br>bits [2:1] Reserved<br>bit [0] bump Rg resistor<br>0 : resonator notch at nominal freq.<br>1 : resonator notch at 10% higher freq. |

## A.2.6.5 RF Analog BBA Control (RX_BBA)

### A.2.6.5.1 Offset

| Register | Offset |
|---|---|
| RX_BBA | Ch |

## A.2.6.5.2 Diagram




## A.2.6.5.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>RX_BBA2_SPARE | rmap_rx_bba2_spare[2:0]<br>Spare bits reserved for future use. |
| 27<br>— | Reserved |
| 26-24<br>RX_BBA2_BW_SEL | rmap_bba2_bw_sel[2:0]<br>Bandwidth control bit for baseband filter chain<br>000b - 1000K<br>001b - 900K<br>010b - 800K<br>011b - 700K Default<br>100b - 600K<br>101b - 500K |
| 23-22<br>— | Reserved |
| 21-16<br>RX_BBA_SPARE | rmap_rx_bba_spare[5:0]<br>Spare bits reserved for future use. bit[5:2] are unused. bit[1:0] are mapped as follows for baseband common mode programming: |

*Table continues on the next page...*

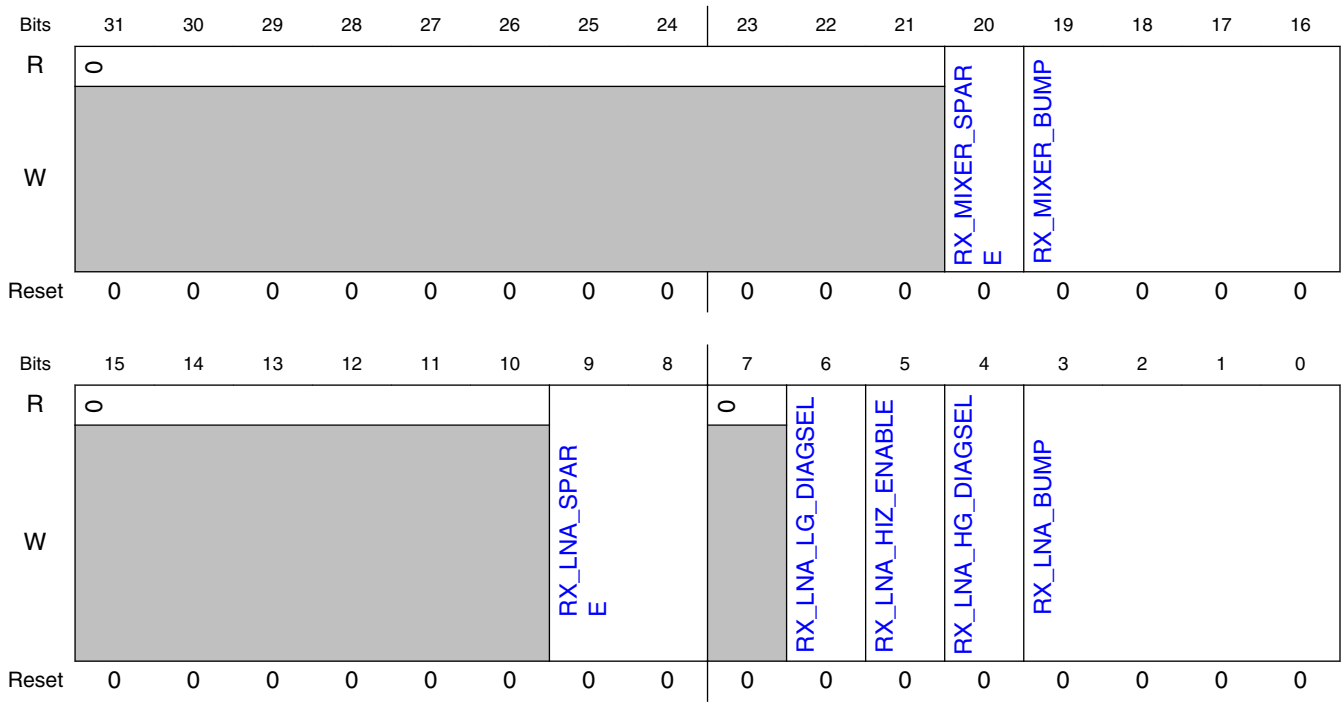
| Field | Function |
|---|---|
| | 000000b - 600mV (Default)<br>000001b - 675mV<br>000010b - 450mV<br>000011b - 525mV |
| 15-8<br>— | Reserved |
| 7<br>RX_BBA_DIAGSEL4 | rmap_rx_bba_diagsel4<br>Diagsel for BBA<br>0b - Diag disable<br>1b - Diag enable |
| 6<br>RX_BBA_DIAGSEL3 | rmap_rx_bba_diagsel3<br>Diagsel for BBA<br>0b - Diag disable<br>1b - Diag enable |
| 5<br>RX_BBA_DIAGSEL2 | rmap_rx_bba_diagsel2<br>Diagsel for BBA<br>0b - Diag disable<br>1b - Diag enable |
| 4<br>RX_BBA_DIAGSEL1 | rmap_rx_bba_diagsel1<br>Diagsel for BBA<br>0b - Diag disable<br>1b - Diag enable |
| 3<br>RX_BBA_CUR_BUMP | rmap_rx_bba_cur_bump<br>BBA amplifier bias current setting. Default= 0. Setting to 1 increases amp current by 1.5X |
| 2-0<br>RX_BBA_BW_SEL | rmap_rx_bba_bw_sel[2:0]<br>Bandwidth control bit for baseband filter chain<br>000b - 1000K<br>001b - 900K<br>010b - 800K<br>011b - 700K Default<br>100b - 600K<br>101b - 500K |

## A.2.6.6 RF Analog LNA Control (RX_LNA)

### A.2.6.6.1 Offset

| Register | Offset |
|---|---|
| RX_LNA | 10h |

### A.2.6.6.2 Diagram




### A.2.6.6.3 Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20<br>RX_MIXER_SPARE | rmap_rx_mixer_spare<br>Spare bit |
| 19-16<br>RX_MIXER_BUMP | rmap_rx_mixer_bump[3:0]<br>[1:0] used as Bump bits for Mixer gate bias:<br>[3:2] bits are unused.<br>0000b - 825mV (Default)<br>0001b - 750mV<br>0010b - 900mV<br>0011b - 975mV |
| 15-10<br>— | Reserved |
| 9-8<br>RX_LNA_SPARE | rmap_rx_lna_spare[1:0]<br>Spare [1] is not used.<br>Spare [0] is used to Bump the PTAT current generating resistor in LNA. Default is 0, Setting to a 1 increases the resistor by 5.6% |

*Table continues on the next page...*

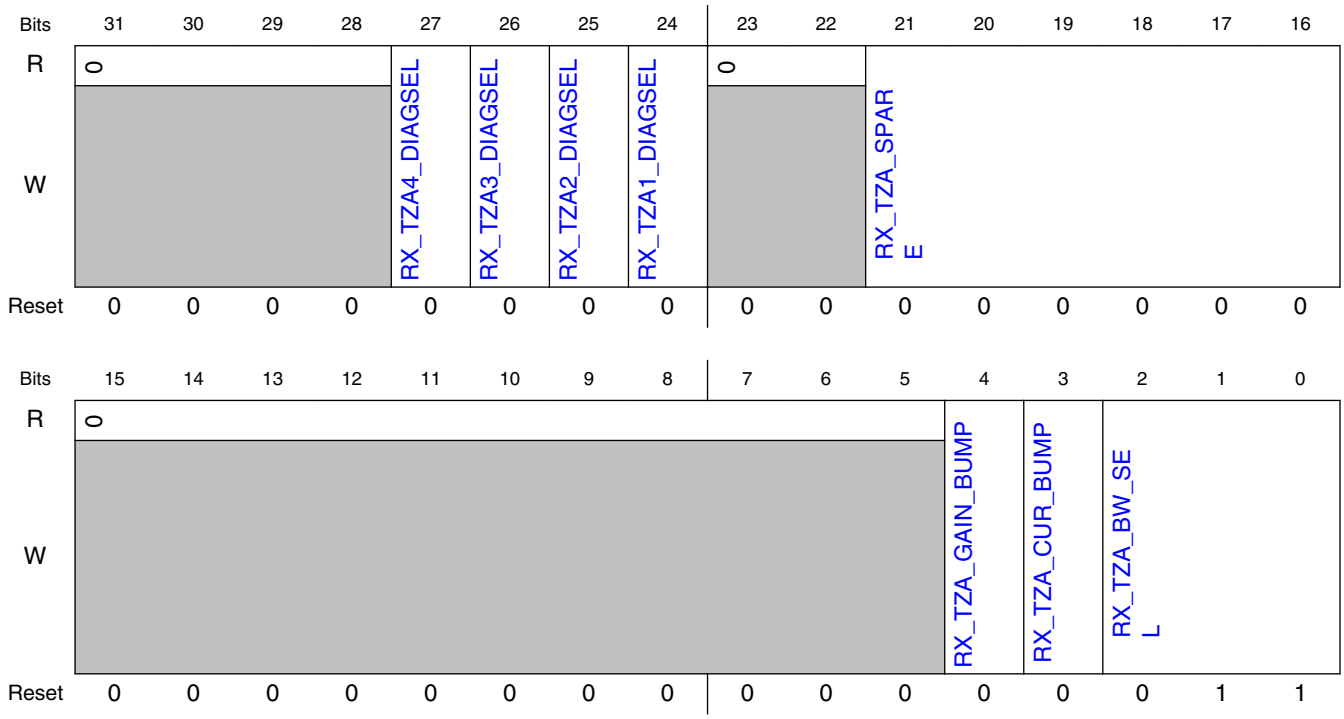
| Field | Function |
|---|---|
| 7<br>— | Reserved |
| 6<br>RX_LNA_LG_DIAGSEL | rmap_rx_lna_lg_diagsel<br>diagsel to check voltage bias in HG pathway circuits. (LNA bias/cmfb block and PTAT current gen). |
| 5<br>RX_LNA_HIZ_ENABLE | rmap_rx_lna_hiZ_enable<br>This bit doesn't exist in LNA top. When LNA is powered down, outputs are highZ. |
| 4<br>RX_LNA_HG_DIAGSEL | rmap_rx_lna_hg_diagsel<br>diagsel to check voltage bias in HG pathway circuits. (LNA bias/cmfb block and PTAT current gen). |
| 3-0<br>RX_LNA_BUMP | rmap_rx_lna_bump[3:0]<br>[1:0] used as Bump bits for LNA current:<br>[3:2] used to Bump LNA output CM:<br>0000b - Default<br>0001b - -25%<br>0010b - +50%<br>0011b - +25%<br>0100b - CM 480mV<br>1000b - CM 600mV<br>1100b - CM 660mV |

## A.2.6.7 RF Analog TZA Control (RX_TZA)

### A.2.6.7.1 Offset

| Register | Offset |
|---|---|
| RX_TZA | 14h |

## A.2.6.7.2 Diagram




## A.2.6.7.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>RX_TZA4_DIAGSEL | rmap_rx_tza4_diagsel<br>Diagsel for TZA<br>0b - Diag disable<br>1b - Diag enable |
| 26<br>RX_TZA3_DIAGSEL | rmap_rx_tza3_diagsel<br>Diagsel for TZA<br>0b - Diag disable<br>1b - Diag enable |
| 25<br>RX_TZA2_DIAGSEL | rmap_rx_tza2_diagsel<br>Diagsel for TZA<br>0b - Diag disable<br>1b - Diag enable |
| 24<br>RX_TZA1_DIAGSEL | rmap_rx_tza1_diagsel<br>Diagsel for TZA<br>0b - Diag disable<br>1b - Diag enable |

*Table continues on the next page...*

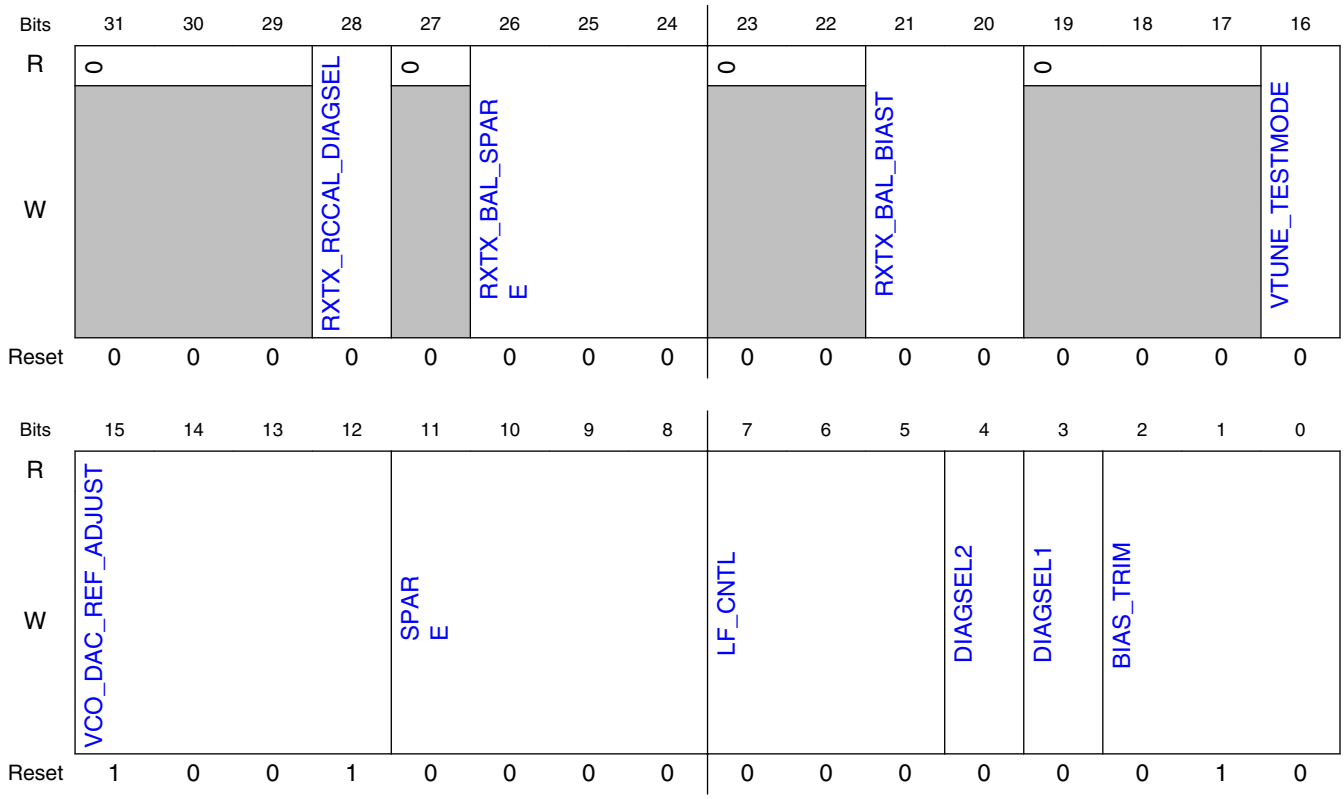
| Field | Function |
|---|---|
| 23-22<br>— | Reserved |
| 21-16<br>RX_TZA_SPARE | rmap_rx_tza_spare[5:0]<br>bit[5:3] are unused.bit[2] when clear uses a limited range for the TZA DCOC DAC; when set to 1 reverts to full range for TZA DCOC DAC. bit[1:0] are mapped as follows for baseband common mode programming:<br>000000b - 600mV (Default)<br>000001b - 675mV<br>000010b - 450mV<br>000011b - 525mV |
| 15-5<br>— | Reserved |
| 4<br>RX_TZA_GAIN_BUMP | rmap_rx_tza_gain_bump<br>TZA Trans-impedance gain setting. Default=0.<br>Setting it to 1 bumps DOWN the Trans-impedance of the TZA by 25% |
| 3<br>RX_TZA_CUR_BUMP | rmap_rx_tza_cur_bump<br>TZA amplifier bias current setting. Default= 0. Setting to 1 increases amp current by 1.5X |
| 2-0<br>RX_TZA_BW_SEL | rmap_rx_tza_bw_sel[2:0]<br>Bandwidth control bit for baseband filter chain<br>000b - 1000K<br>001b - 900K<br>010b - 800K<br>011b - 700K Default<br>100b - 600K<br>101b - 500K |

## A.2.6.8 RF Analog Aux PLL Control (RX_AUXPLL)

### A.2.6.8.1 Offset

| Register | Offset |
|---|---|
| RX_AUXPLL | 18h |

## A.2.6.8.2 Diagram




## A.2.6.8.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>RXTX_RCCAL_DIAGSEL | rmap_rxtx_rccal_diagsel<br>Enables diagnostics output |
| 27<br>— | Reserved |
| 26-24<br>RXTX_BAL_SPARE | rmap_rxtx_bal_spare[2:0]<br>Spare bits. Not used so far. |
| 23-22<br>— | Reserved |
| 21-20<br>RXTX_BAL_BIAST | rmap_rxtx_bal_biast[1:0]<br>Internal bias bump for the cap array.<br>00b - 0.6<br>01b - 0.4 |

*Table continues on the next page...*

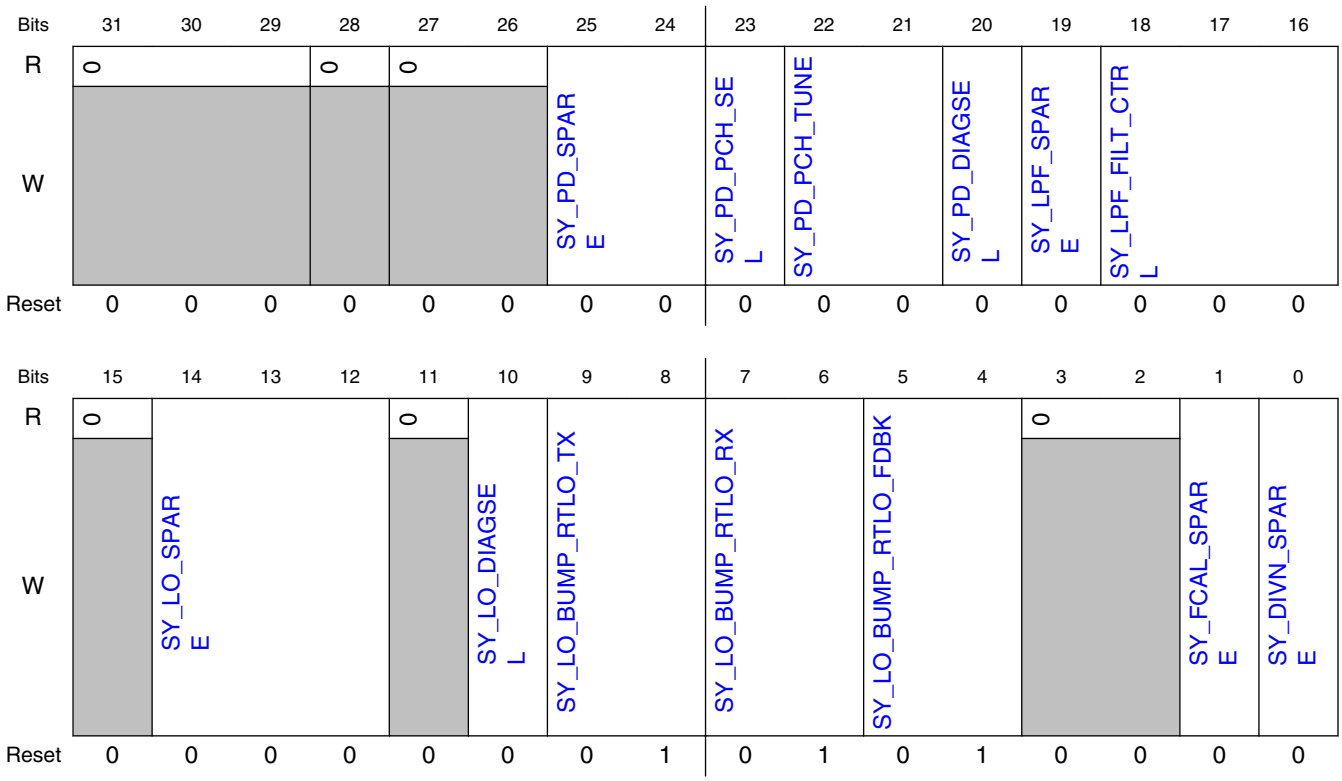
| Field | Function |
|---|---|
| | 10b - 0.9<br>11b - 1.2 |
| 19-17<br>— | Reserved |
| 16<br>VTUNE_TESTMODE | rmap_rxtx_auxpll_vtune_testmode<br>Testmode to force the vtune externally |
| 15-12<br>VCO_DAC_REF_ADJUST | rmap_rxtx_auxpll_vco_dac_ref_adjust[3:0]<br>Bits to tune the DAC current based on reference frequency of 26MHz or 32 MHz<br>4 = 26MHz<br>9 = 32MHz |
| 11-8<br>SPARE | rmap_rxtx_auxpll_spare[3:0]<br>Spare bits - Spare [3] and Spare[2] are used as pins with following functions:<br>Spare [3] - auxpll_clk_out_sel<br>0 selects the output of auxpll to analog(ADC) and digital<br>1 selects DSM clock from divN as the output to analog(ADC) and digital.<br>Spare [2]<br>0 auxpll output to XO is disabled<br>1 The output of auxpll is sent to the XO to be sent out to xo output pad.<br>Spare [0] is used to Bump the PTAT current generating resistor in AUXPLL. Default is 0, Setting to a 1 increases the resistor by 5.6% |
| 7-5<br>LF_CNTL | rmap_rxtx_auxpll_lf_cntl[2:0]<br>Bits to configure the low pass filter poles |
| 4<br>DIAGSEL2 | rmap_rxtx_auxpll_diagsel2<br>Diagnostic select 2 |
| 3<br>DIAGSEL1 | rmap_rxtx_auxpll_diagsel1<br>Diagnostic select 1 |
| 2-0<br>BIAS_TRIM | rmap_rxtx_auxpll_bias_trim[2:0]<br>Change the ratio in which the BG and PTAT currents are added in IREF block which supplies current to RO VCO |

## A.2.6.9 RF Analog Synthesizer Control 1 (SY_CTRL_1)

### A.2.6.9.1 Offset

| Register | Offset |
|---|---|
| SY_CTRL_1 | 1Ch |

## A.2.6.9.2 Diagram




## A.2.6.9.3 Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>— | Reserved |
| 27-26<br>— | Reserved |
| 25-24<br>SY_PD_SPARE | rmap_sy_pd_spare[1:0]<br>Spare[0] is used as pd_out pull down mode -<br>00b - Default ;<br>01b - PD output is pulled down. |
| 23<br>SY_PD_PCH_SEL | rmap_sy_pd_pch_sel<br>pch_sel is used to select the inverter based precharge or resistor divider based precharge<br>0b - inverter based precharge<br>1b - resistor divider based precharge |
| 22-21 | rmap_sy_pd_pch_tune[1:0] |

*Table continues on the next page...*

| Field | Function |
|---|---|
| SY_PD_PCH_TUNE | These bits adjust the precharge voltage |
| 20<br>SY_PD_DIAGSEL | rmap_sy_pd_diagsel<br>Enables the diagnostic select |
| 19<br>SY_LPF_SPARE | rmap_sy_lpf_spare<br>Spare bits |
| 18-16<br>SY_LPF_FILT_CTRL | rmap_sy_lpf_filt_ctrl[2:0]<br>Configures the pole location and the order of the filer in LPF |
| 15<br>— | Reserved |
| 14-12<br>SY_LO_SPARE | rmap_sy_lo_spare[2:0]<br>Spare bits |
| 11<br>— | Reserved |
| 10<br>SY_LO_DIAGSEL | rmap_sy_lo_diagsel<br>Diagsel for LO.<br>0b - Diag disable<br>1b - Diag enable |
| 9-8<br>SY_LO_BUMP_RTLO_TX | rmap_sy_lo_bump_rtlo_tx[1:0]<br>Change resistance from supply to RX LO to change DC drop(vddl_div/vddl_buf)<br>00b - 1.071/1.065 V<br>01b - 1.092/1.090 V<br>10b - 1.099/1.098 V<br>11b - 1.10/1.1 V |
| 7-6<br>SY_LO_BUMP_RTLO_RX | rmap_sy_lo_bump_rtlo_rx[1:0]<br>Change resistance from supply to RX LO to change DC drop(vddl_div/vddl_buf)<br>00b - 1.051/1.037 V<br>01b - 1.082/1.075 V<br>10b - 1.092/1.088 V<br>11b - 1.098/1.094 V |
| 5-4<br>SY_LO_BUMP_RTLO_FDBK | rmap_sy_lo_bump_rtlo_fdbk[1:0]<br>Change resistance from supply to RX LO to change DC drop(vddl)<br>00b - 1.045 V<br>01b - 1.084 V<br>10b - 1.097 V<br>11b - 1.10 V |
| 3-2<br>— | Reserved |
| 1<br>SY_FCAL_SPARE | rmap_sy_fcal_spare<br>Spare bits |

*Table continues on the next page...*

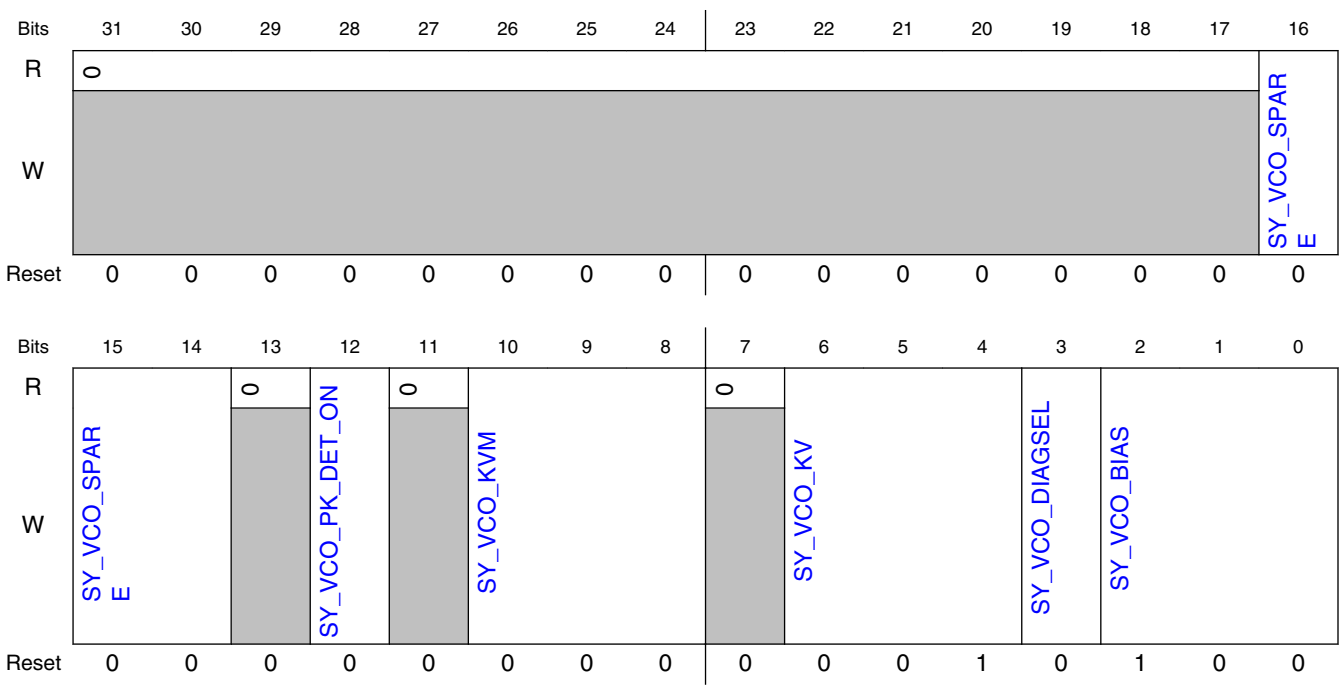
| Field | Function |
|---|---|
| 0<br>SY_DIVN_SPARE | rmap_sy_divn_spare<br>Spare bits |

## A.2.6.10 RF Analog Synthesizer Control 2 (SY_CTRL_2)

### A.2.6.10.1 Offset

| Register | Offset |
|---|---|
| SY_CTRL_2 | 20h |

### A.2.6.10.2 Diagram




### A.2.6.10.3 Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 16-14<br>SY_VCO_SPARE | rmap_sy_vco_spare[2:0]<br>Spare bits |
| 13<br>— | Reserved |
| 12<br>SY_VCO_PK_DET_ON | rmap_sy_vco_pk_det_on<br>Enable peak detect for VCO.<br>0b - Disable<br>1b - Enable |
| 11<br>— | Reserved |
| 10-8<br>SY_VCO_KVM | rmap_sy_vco_kvm[2:0]<br>Changes modulation port Kv.<br>000b - 10MHz/V<br>001b - 20MHz/V<br>010b - 30MHz/V<br>011b - 40MHz/V<br>100b - 40MHz/V<br>101b - 40MHz/V<br>110b - 40MHz/V<br>111b - 40MHz/V |
| 7<br>— | Reserved |
| 6-4<br>SY_VCO_KV | rmap_sy_vco_kv[2:0]<br>Changes control port Kv.<br>000b - 50MHz/V<br>001b - 60MHz/V<br>010b - 70MHz/V<br>011b - 80MHz/V<br>100b - 80MHz/V<br>101b - 80MHz/V<br>110b - 80MHz/V<br>111b - 80MHz/V |
| 3<br>SY_VCO_DIAGSEL | rmap_sy_vco_diagsel<br>Diagsel for VCO.<br>0b - Diag disable<br>1b - Diag enable |
| 2-0<br>SY_VCO_BIAS | rmap_sy_vco_bias[2:0]<br>Change resistance from supply to VCO to change the DC drop (ldo supply = 1.124V and dc drop of 10mV)<br>000b - 0.97V<br>001b - 1.033V<br>010b - 1.06V<br>011b - 1.07V<br>100b - 1.08V<br>101b - 1.085V<br>110b - 1.090V |

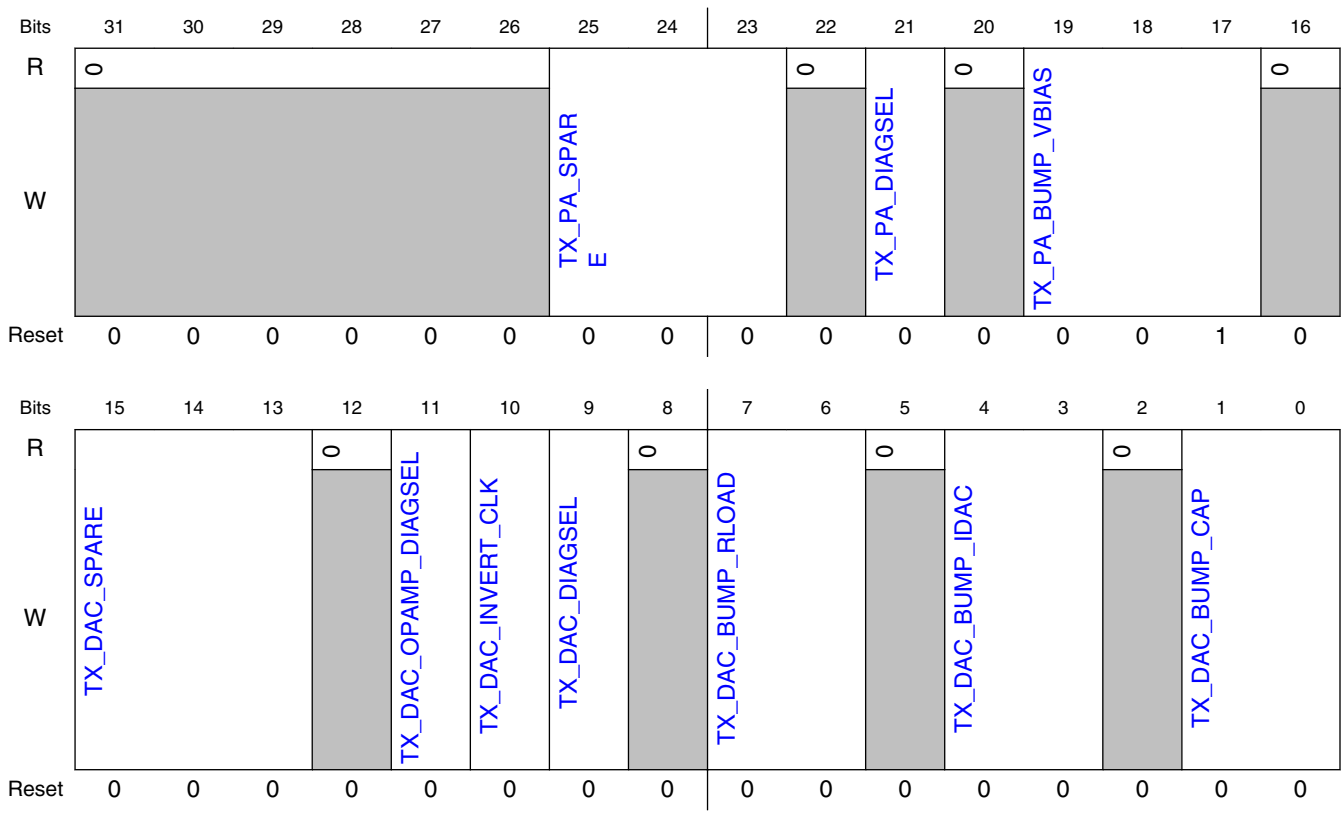
| Field | Function |
|---|---|
| | 111b - 1.095V |

## A.2.6.11 RF Analog TX HPM DAC and PA Control (TX_DAC_PA)

### A.2.6.11.1 Offset

| Register | Offset |
|---|---|
| TX_DAC_PA | 24h |

### A.2.6.11.2 Diagram




### A.2.6.11.3 Fields

| Field | Function |
|---|---|
| 31-26 | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 25-23<br>TX_PA_SPARE | rmap_tx_pa_spare[2:0]<br>Spare bits. Not used so far.<br>Spare bits [2:1] are not used.<br>Spare [0] is used to Bump the PTAT current generating resistor in TX PA. Default is 0, Setting to a 1 increases the resistor by 5.6% |
| 22<br>— | Reserved |
| 21<br>TX_PA_DIAGSEL | rmap_tx_pa_diagsel<br>Block level diag select bit |
| 20<br>— | Reserved |
| 19-17<br>TX_PA_BUMP_VBIAS | rmap_tx_pa_bump_vbias[2:0]<br>Bump bits for the PA bias voltage for the gm/Nmos.<br>000b - 0.557<br>001b - 0.651<br>010b - 0.741<br>011b - 0.822<br>100b - 0.590<br>101b - 0.683<br>110b - 0.771<br>111b - 0.850 |
| 16<br>— | Reserved |
| 15-13<br>TX_DAC_SPARE | rmap_tx_dac_spare[2:0]<br>Spare bits for DAC/OPMAP |
| 12<br>— | Reserved |
| 11<br>TX_DAC_OPAMP_DIAGSEL | rmap_tx_dac_opamp_diagsel<br>Enables Diagnostics for OPAMP<br>0b - Disable Diag<br>1b - Enable Diag |
| 10<br>TX_DAC_INVERT_CLK | rmap_tx_dac_invert_clk |
| 9<br>TX_DAC_DIAGSEL | rmap_tx_dac_diagsel<br>Enables Diagnostics for DAC<br>0b - Disable Diag<br>1b - Enable Diag |
| 8<br>— | Reserved |
| 7-6 | rmap_tx_dac_bump_rload[1:0] |

*Table continues on the next page...*

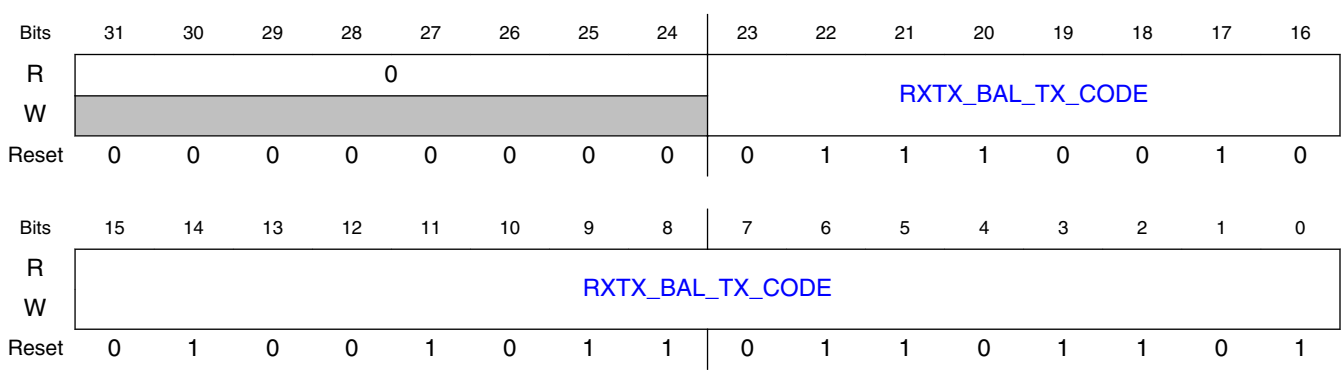
| Field | Function |
|---|---|
| TX_DAC_BUMP_RLOAD | change DAC resistor load.<br>00b - 3.12 kohms(default)<br>01b - 2.34 kohms<br>10b - 3.9 kohms<br>11b - 4.6 kohms |
| 5<br>— | Reserved |
| 4-3<br>TX_DAC_BUMP_IDAC | rmap_tx_dac_bump_idac[1:0]<br>change DAC LSB current.(4*lsb current quoted here)<br>00b - 250nA(default)<br>01b - 207nA<br>10b - 312nA<br>11b - 415nA |
| 2<br>— | Reserved |
| 1-0<br>TX_DAC_BUMP_CAP | rmap_tx_dac_bump_cap[1:0]<br>change filter cap at opamp to filter DAC images further.<br>00b - 1pF(default)<br>01b - 1.5pF<br>10b - 1.5pF<br>11b - 2pF |

## A.2.6.12 RF Analog Balun TX Mode Control (BALUN_TX)

### A.2.6.12.1 Offset

| Register | Offset |
|---|---|
| BALUN_TX | 28h |

### A.2.6.12.2 Diagram



### A.2.6.12.3 Fields

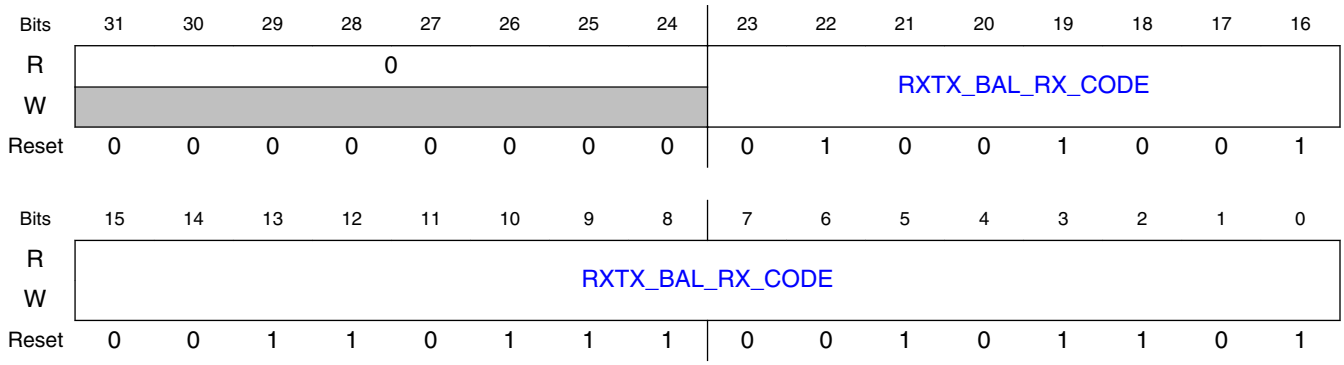
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>RXTX_BAL_TX_CODE | Balun Tuning Cap Settings in Transmit Mode<br>This register contains the Balun Cap Settings for Eight Tuning Ranges. The Ranges and the associated register bits that apply for each Range are as follows :<br>Tuning Range 0 = Radio channel less than or equal to 2375 MHz<br>Tuning Range 1 = Radio channel between 2375 and 2390 MHz<br>Tuning Range 2 = Radio channel between 2390 and 2405 MHz<br>Tuning Range 3 = Radio channel between 2405 and 2420 MHz<br>Tuning Range 4 = Radio channel between 2420 and 2435 MHz<br>Tuning Range 5 = Radio channel between 2435 and 2450 MHz<br>Tuning Range 6 = Radio channel between 2450 and 2465 MHz<br>Tuning Range 7 = Radio channel above 2465 MHz<br>Tuning Range 0 = RXTX_BAL_TX_CODE[ 2:0 ]<br>Tuning Range 1 = RXTX_BAL_TX_CODE[ 5:3 ]<br>Tuning Range 2 = RXTX_BAL_TX_CODE[ 8:6 ]<br>Tuning Range 3 = RXTX_BAL_TX_CODE[11:9 ]<br>Tuning Range 4 = RXTX_BAL_TX_CODE[14:12]<br>Tuning Range 5 = RXTX_BAL_TX_CODE[17:15]<br>Tuning Range 6 = RXTX_BAL_TX_CODE[20:18]<br>Tuning Range 7 = RXTX_BAL_TX_CODE[23:21] |

## A.2.6.13 RF Analog Balun RX Mode Control (BALUN_RX)

### A.2.6.13.1 Offset

| Register | Offset |
|---|---|
| BALUN_RX | 2Ch |

## A.2.6.13.2 Diagram




## A.2.6.13.3 Fields

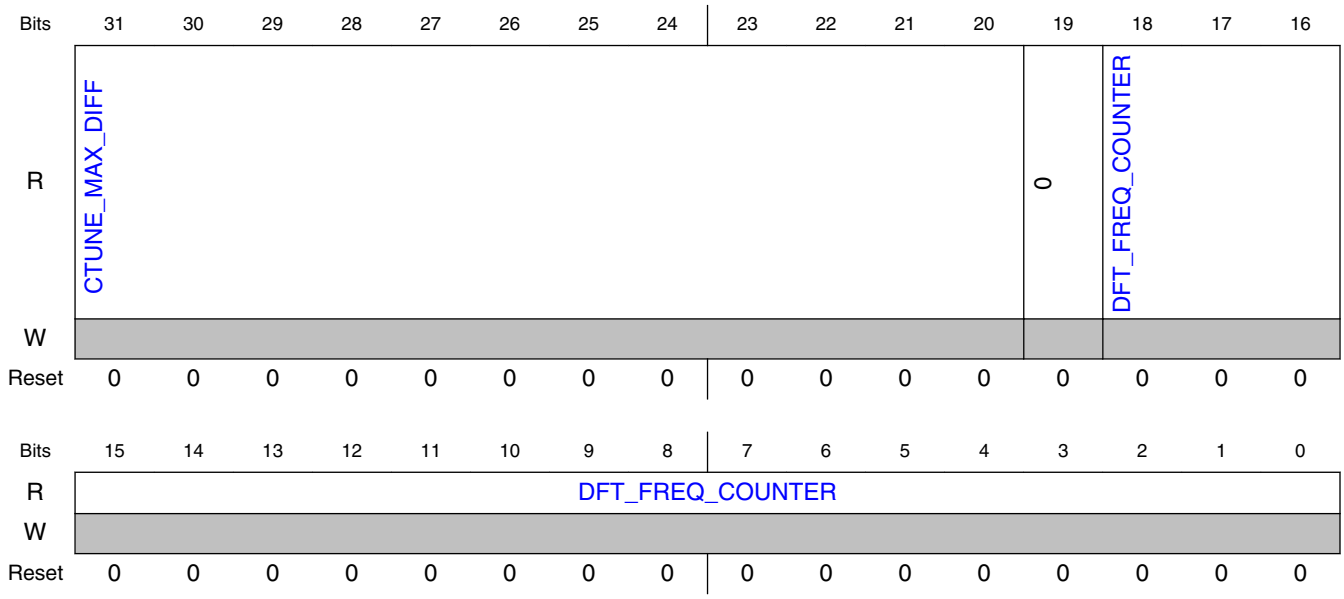
| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>RXTX_BAL_RX_CODE | Balun Tuning Cap Settings in Receive Mode<br>This register contains the Balun Cap Settings for Eight Tuning Ranges. The Ranges and the associated register bits that apply for each Range are as follows :<br>Tuning Range 0 = Radio channel less than or equal to 2375 MHz<br>Tuning Range 1 = Radio channel between 2375 and 2390 MHz<br>Tuning Range 2 = Radio channel between 2390 and 2405 MHz<br>Tuning Range 3 = Radio channel between 2405 and 2420 MHz<br>Tuning Range 4 = Radio channel between 2420 and 2435 MHz<br>Tuning Range 5 = Radio channel between 2435 and 2450 MHz<br>Tuning Range 6 = Radio channel between 2450 and 2465 MHz<br>Tuning Range 7 = Radio channel above 2465 MHz<br>Tuning Range 0 = RXTX_BAL_RX_CODE[ 2:0 ]<br>Tuning Range 1 = RXTX_BAL_RX_CODE[ 5:3 ]<br>Tuning Range 2 = RXTX_BAL_RX_CODE[ 8:6 ]<br>Tuning Range 3 = RXTX_BAL_RX_CODE[11:9 ]<br>Tuning Range 4 = RXTX_BAL_RX_CODE[14:12]<br>Tuning Range 5 = RXTX_BAL_RX_CODE[17:15]<br>Tuning Range 6 = RXTX_BAL_RX_CODE[20:18]<br>Tuning Range 7 = RXTX_BAL_RX_CODE[23:21] |

# A.2.6.14 RF Analog DFT Observation Register 1 (DFT_OBSV_1)

## A.2.6.14.1 Offset

| Register | Offset |
|---|---|
| DFT_OBSV_1 | 30h |

## A.2.6.14.2 Diagram




## A.2.6.14.3 Fields

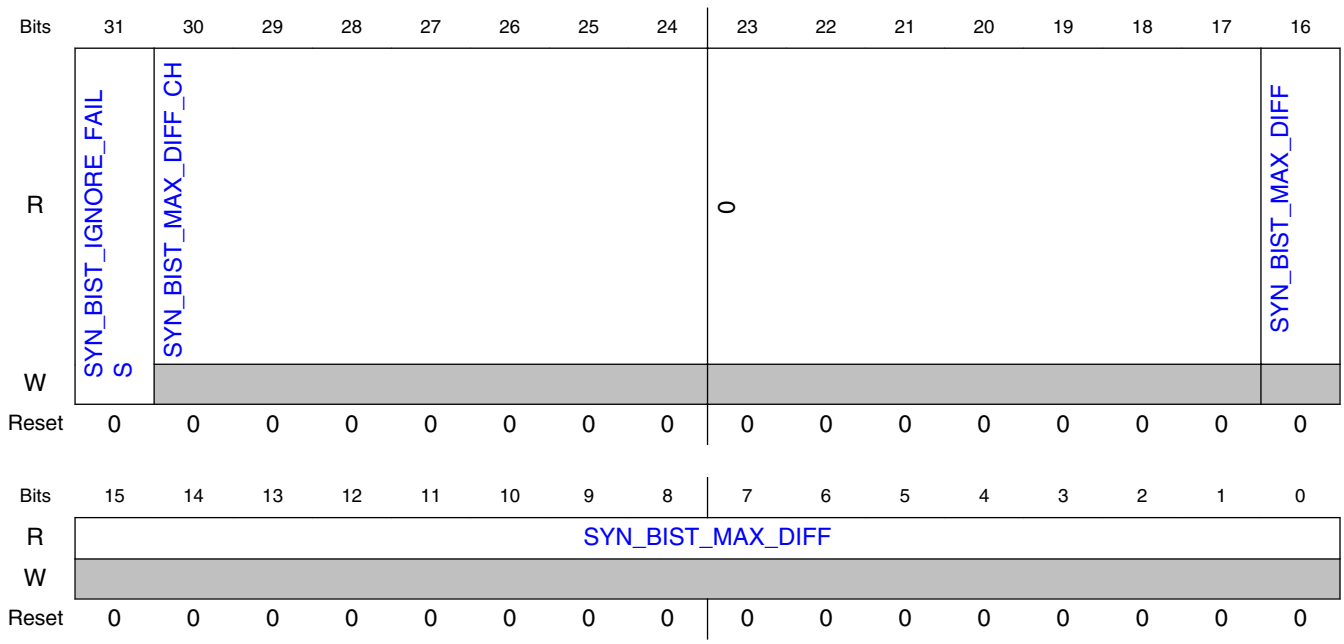
| Field | Function |
|---|---|
| 31-20<br>CTUNE_MAX_DIFF | Maximum Frequency Count Difference found by the Coarse Tune BIST<br>This register is the Absolute Value of the maximum frequency count difference that the Coarse Tune BIST found after tuning to all the Radio channels. |
| 19<br>— | Reserved |
| 18-0<br>DFT_FREQ_COUNTER | VCO Frequency Counter Value<br>This Register shows the value of the VCO Frequency Meter after a Count Time has completed.<br>The control bits for the Frequency Meter are in the PLL_DIG register space: FREQ_COUNT_GO, FREQ_COUNT_FINISHED, and FREQ_COUNT_TIME.<br>Note that the counting is done at the VCO frequency which is 2X the Carrier Frequency. |

# A.2.6.15 RF Analog DFT Observation Register 2 (DFT_OBSV_2)

## A.2.6.15.1 Offset

| Register | Offset |
|---|---|
| DFT_OBSV_2 | 34h |

## A.2.6.15.2 Diagram




## A.2.6.15.3 Fields

| Field | Function |
|---|---|
| 31<br>SYN_BIST_IGNORE_FAILS | PLL Frequency Synthesizer BIST Ignore Fails<br>If this bit is set, then the PLL Frequency Synthesizer BIST will not stop after it is unable to lock on a Radio Channel. Instead the BIST will ignore any channel locking fails and will continue to attempt to lock on every Radio Channel requested |
| 30-24<br>SYN_BIST_MAX_DIFF_CH | PLL Frequency Synthesizer BIST Worst Channel<br>This register shows the Radio Channel that had the maximum Frequency Meter count difference during the PLL Frequency Synthesizer BIST. |
| 23-17<br>— | Reserved |

*Table continues on the next page...*

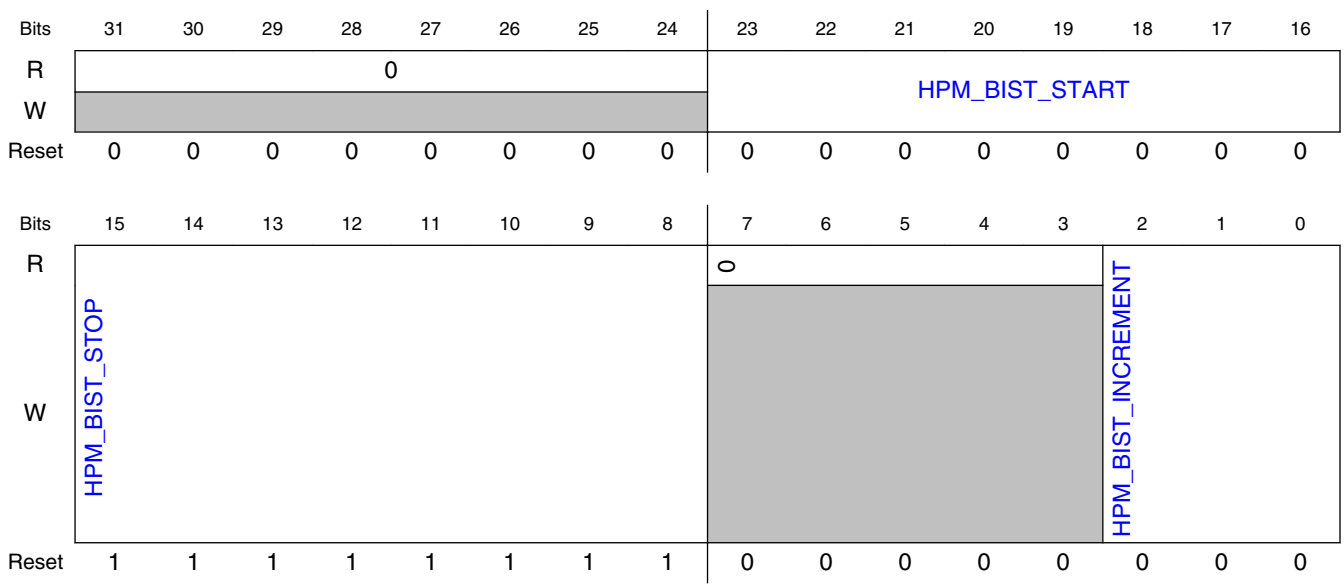
| Field | Function |
|---|---|
| 16-0<br>SYN_BIST_MAX_DIFF | PLL Frequency Synthesizer BIST Worst Frequency Count<br>This register shows the maximum Frequency Meter count difference on a channel during the PLL Frequency Synthesizer BIST. |

## A.2.6.16 RF Analog DFT Observation Register 3 (DFT_OBSV_3)

### A.2.6.16.1 Offset

| Register | Offset |
|---|---|
| DFT_OBSV_3 | 38h |

### A.2.6.16.2 Diagram




### A.2.6.16.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16 | HPM BIST Start Value<br>This is the value at which the HPM BIST will Start |

*Table continues on the next page...*

| Field | Function |
|---|---|
| HPM_BIST_START | |
| 15-8<br>HPM_BIST_STOP | HPM BIST Stop Value<br>This is the value at which the HPM BIST will Stop |
| 7-3<br>— | Reserved |
| 2-0<br>HPM_BIST_INCREMENT | HPM BIST Increment Value<br>This selects the choices for the HPM BIST range or increment.<br>HPM INL BIST Range<br>0 : dac_range = 8<br>1 : dac_range = 16<br>2 : dac_range = 32<br>3 : dac_range = 64<br>4 : dac_range = 128<br>5 : dac_range = 255<br>6,7 : Reserved<br>HPM DNL BIST Increment<br>0 : dac_increment = 1<br>1 : dac_increment = 2<br>2 : dac_increment = 4<br>3 : dac_increment = 8<br>4 : dac_increment = 16<br>5 : dac_increment = 32<br>6,7 : Reserved |

## A.2.7 XCVR_MISC register descriptions

### A.2.7.1 XCVR_CTRL_ADDR Memory map

XCVR_MISC base address: 4103_0280h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TRANSCEIVER CONTROL (XCVR_CTRL) | 32 | RW | 0010_1000h |

*Table continues on the next page...*

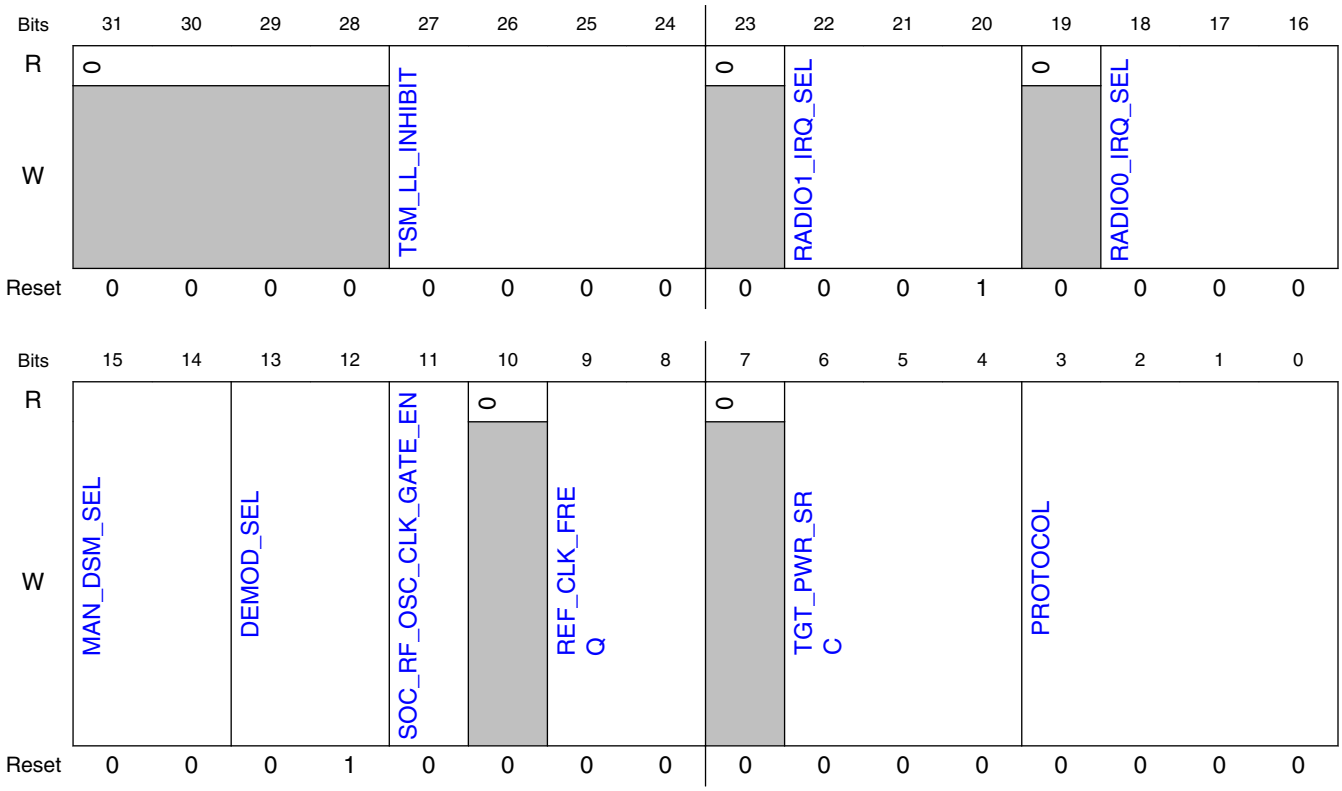
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | TRANSCEIVER STATUS (XCVR_STATUS) | 32 | W1C | See description. |
| 8h | BLE ARBITRATION CONTROL (BLE_ARB_CTRL) | 32 | RW | 0000_0000h |
| Ch | OVERWRITE VERSION (OVERWRITE_VER) | 32 | RW | 0000_0000h |
| 10h | DIGITAL TEST MUX CONTROL (DTEST_CTRL) | 32 | RW | 0000_0000h |
| 14h | TRANSCEIVER DMA CONTROL (DMA_CTRL) | 32 | RW | 0000_0300h |
| 18h | TRANSCEIVER DMA DATA (DMA_DATA) | 32 | RO | 0000_0000h |
| 1Ch | PACKET RAM CONTROL (PACKET_RAM_CTRL) | 32 | RW | 0000_0000h |
| 20h | PACKET RAM DEBUG RAM STOP ADDRESS (RAM_STOP_ADDR) | 32 | RO | 0000_0000h |
| 24h | FAD CONTROL (FAD_CTRL) | 32 | RW | 0000_F080h |
| 28h | LOW POWER PREAMBLE SEARCH CONTROL (LPPS_CTRL) | 32 | RW | 6426_0000h |
| 2Ch | COEXISTENCE CONTROL (COEX_CTRL) | 32 | RW | 0000_0000h |
| 30h | CRC/WHITENER CONFIG REGISTER (CRCW_CFG) | 32 | RW | 0000_0000h |
| 34h | CRC ERROR CORRECTION MASK (CRC_EC_MASK) | 32 | RO | 0000_0000h |
| 38h | CRC RESULT (CRC_RES_OUT) | 32 | RO | 0000_0000h |
| 3Ch | CRC/WHITENER CONFIG 2 REGISTER (CRCW_CFG2) | 32 | RW | 0000_0000h |

## A.2.7.2 TRANSCEIVER CONTROL (XCVR_CTRL)

### A.2.7.2.1 Offset

| Register | Offset |
|---|---|
| XCVR_CTRL | 0h |

## A.2.7.2.2 Diagram




## A.2.7.2.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>TSM_LL_INHIBIT | TSM Per-Link-Layer Inhibit<br>Nominally, any protocol engine can request TSM to start a TX or RX sequence at any time. In a multiprotocol setting, individual protocol engines can (optionally) be selectively blocked from accessing TSM by setting one of the following bits of TSM_LL_INHIBIT[3:0]:<br>**xxx1**: BLE commands to TSM are inhibited<br>**xxx0**: BLE commands to TSM are permitted<br>**xx1x**: 802.15.4 commands to TSM are inhibited<br>**xx0x**: 802.15.4 commands to TSM are permitted<br>**1xxx**: GENERIC_FSK commands to TSM are inhibited<br>**0xxx**: GENERIC_FSK commands to TSM are permitted |
| 23<br>— | Reserved |
| 22-20 | RADIO1_IRQ_SEL<br>Assigns Radio #1 Interrupt (ipi_int_radio1) to a Protocol Engine |

*Table continues on the next page...*

| Field | Function |
|---|---|
| RADIO1_IRQ_SEL | 000b - Assign Radio #1 Interrupt to BLE<br>001b - Assign Radio #1 Interrupt to 802.15.4<br>010b - Radio #1 Interrupt unassigned<br>011b - Assign Radio #1 Interrupt to GENERIC_FSK<br>100b - Radio #1 Interrupt unassigned<br>101b - Radio #1 Interrupt unassigned<br>110b - Radio #1 Interrupt unassigned<br>111b - Radio #1 Interrupt unassigned |
| 19<br>— | Reserved |
| 18-16<br>RADIO0_IRQ_SEL | RADIO0_IRQ_SEL<br>Assigns Radio #0 Interrupt (ipi_int_radio0) to a Protocol Engine<br>000b - Assign Radio #0 Interrupt to BLE<br>001b - Assign Radio #0 Interrupt to 802.15.4<br>010b - Radio #0 Interrupt unassigned<br>011b - Assign Radio #0 Interrupt to GENERIC_FSK<br>100b - Radio #0 Interrupt unassigned<br>101b - Radio #0 Interrupt unassigned<br>110b - Radio #0 Interrupt unassigned<br>111b - Radio #0 Interrupt unassigned |
| 15-14<br>MAN_DSM_SEL | Manual DSM Selector<br>Assigning Manual DSM to a protocol is done by way of the MAN_DSM_SEL[1:0] register, as shown in the following table:<br>0: Manual DSM Not Assigned<br>1: Manual DSM Assigned to 802.15.4<br>2: Manual DSM Assigned to GENERIC_FSK |
| 13-12<br>DEMOD_SEL | Demodulator Selector<br>This bit selects the demodulator used during reception<br>00b - No demodulator selected<br>01b - Use NXP Multi-standard PHY demodulator<br>10b - Use Legacy 802.15.4 demodulator<br>11b - Reserved |
| 11<br>SOC_RF_OSC_CLK_GATE_EN | SOC_RF_OSC_CLK_GATE_EN<br>Enable 3V version of RF OSC Clock for use by the SoC |
| 10<br>— | Reserved |
| 9-8<br>REF_CLK_FREQ | Radio Reference Clock Frequency<br>This register selects the Reference Clock Frequency for the Radio.<br>00b - 32 MHz<br>01b - 26 MHz<br>10b - Reserved<br>11b - Reserved |
| 7<br>— | Reserved |
| 6-4<br>TGT_PWR_SRC | Target Power Source<br>For determining transmit power, the TGT_PWR_SRC[2:0] bits control target power selection, according to the following table. |

*Table continues on the next page...*

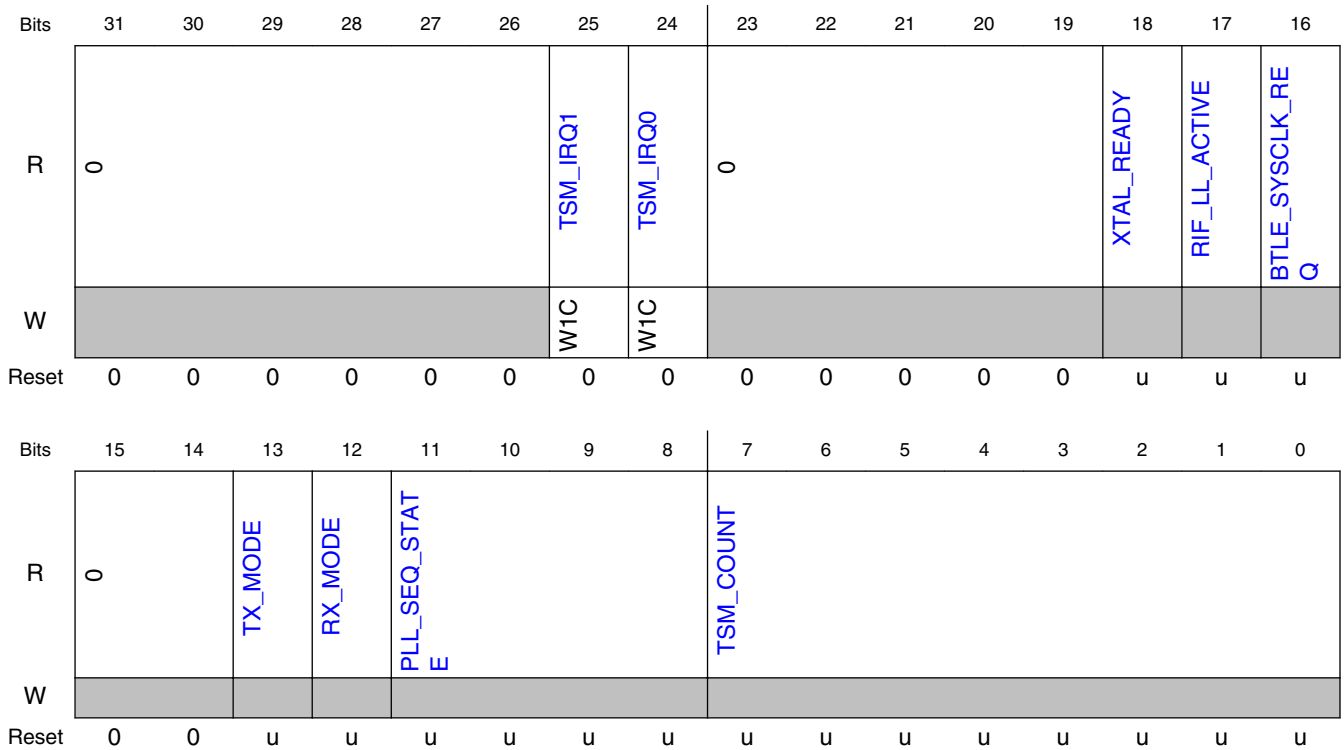
| Field | Function |
|---|---|
| | 0: TARGET POWER SOURCE is PA_POWER[5:0] register (XCVR space)<br>1: TARGET POWER SOURCE is BTLE Link Layer<br>2: TARGET POWER SOURCE is 802.15.4 Link Layer (PA_PWR[5:0] register in 802.15.4 space)<br>3: Reserved<br>4: TARGET POWER SOURCE is GENERIC_FSK Link Layer (TX_POWER[5:0] register in GENERIC_FSK space)<br>5: Reserved<br>6: Reserved<br>7: TARGET POWER SOURCE is determined by the PROTOCOL[3:0] bits |
| 3-0<br>PROTOCOL | Radio Protocol Selection<br>This register selects the Radio Communication Protocol.<br>0000b - BLE<br>0001b - BLE in MBAN<br>0010b - BLE overlap MBAN<br>0011b - Reserved<br>0100b - 802.15.4<br>0101b - 802.15.4j<br>0110b - Radio Channels 0-127 selectable, FSK<br>0111b - Radio Channels 0-127 selectable, GFSK<br>1000b - Generic GFSK, with Gaussian Filter<br>1001b - Generic MSK, O-QPSK encoding<br>1010b - Generic FSK, direct +/- Fdev FSK |

## A.2.7.3 TRANSCEIVER STATUS (XCVR_STATUS)

### A.2.7.3.1 Offset

| Register | Offset |
|---|---|
| XCVR_STATUS | 4h |

## A.2.7.3.2 Diagram




## A.2.7.3.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>TSM_IRQ1 | TSM Interrupt #1<br>0b - TSM Interrupt #1 is not asserted.<br>1b - TSM Interrupt #1 is asserted. Write '1' to this bit to clear it. |
| 24<br>TSM_IRQ0 | TSM Interrupt #0<br>0b - TSM Interrupt #0 is not asserted.<br>1b - TSM Interrupt #0 is asserted. Write '1' to this bit to clear it. |
| 23-19<br>— | Reserved |
| 18<br>XTAL_READY | RF Osciallator Xtal Ready<br>Oscillator warmup count complete.<br>0b - Indicates that the RF Oscillator is disabled or has not completed its warmup.<br>1b - Indicates that the RF Oscillator has completed its warmup count and is ready for use. |
| 17<br>RIF_LL_ACTIVE | Link Layer Active Indication<br>Reflects the state of the BTLE “Link Layer Active” status bit. RIF_LL_ACTIVE is to be used by the host as an 'early' indication to prevent host to do any operations while BTLE IP is doing transceiver operations, so as to reduce the peak power and noise. |
| 16 | BTLE System Clock Request |

*Table continues on the next page...*

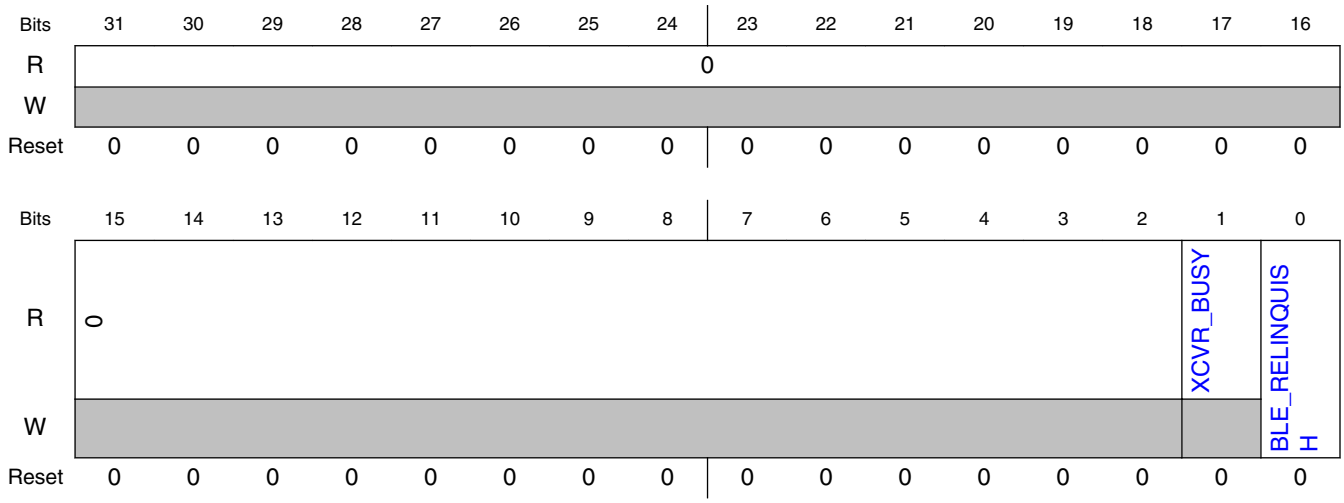
| Field | Function |
|---|---|
| BTLE_SYSCLK_REQ | Reflects the state of the BTLE oscillator request signal. BTLE_SYSCLK_REQ is the BTLE control for the RF Oscillator. BTLE will deassert this signal upon entering DSM (deep sleep mode) to request oscillator turn-off, and will re-assert it prior to exiting DSM. The turn-on leadtime on this signal for exiting DSM, is programmable with the BTLE block. This read-only bit can thus be queried to ascertain the power-state of BTLE. |
| 15-14<br>— | Reserved |
| 13<br>TX_MODE | Transmit Mode<br>Indicates an TX transceiver operation is in progress. |
| 12<br>RX_MODE | Receive Mode<br>Indicates an RX transceiver operation is in progress. |
| 11-8<br>PLL_SEQ_STATE | PLL Sequence State<br>Reflects the state of the PLL digital state machine.<br>0000b - PLL OFF<br>0010b - CTUNE<br>0011b - CTUNE_SETTLE<br>0110b - HPMCAL1<br>1000b - HPMCAL1_SETTLE<br>1010b - HPMCAL2<br>1100b - HPMCAL2_SETTLE<br>1111b - PLLREADY |
| 7-0<br>TSM_COUNT | TSM_COUNT<br>Reflects the instantaneous value of the TSM counter. |

## A.2.7.4 BLE ARBITRATION CONTROL (BLE_ARB_CTRL)

### A.2.7.4.1 Offset

| Register | Offset |
|---|---|
| BLE_ARB_CTRL | 8h |

### A.2.7.4.2 Diagram




### A.2.7.4.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>XCVR_BUSY | Transceiver Busy Status Bit<br>0b - RF Channel in available (TSM is idle)<br>1b - RF Channel in use (TSM is busy) |
| 0<br>BLE_RELINQUISH | BLE Relinquish Control<br>This bit forces the BLE protocol engine to immediately relinquish access to the RF Channel, in favor of another protocol with higher arbitration priority. BLE is denied RF channel access by blocking BLE access to the TSM; when this bit is subsequently cleared, BLE access is restored. |

## A.2.7.5 OVERWRITE VERSION (OVERWRITE_VER)

### A.2.7.5.1 Offset

| Register | Offset |
|---|---|
| OVERWRITE_VER | Ch |

### A.2.7.5.2 Function

The Overwrite Version allows software to store a version number of trim and calibration values which are used to overwrite the chip default values in the registers. Typically,

software would perform the overwrite of the defaults in transceiver registers and then write the version number from the file containing the overwrite values into this register.

**NOTE**

This register has no hardware connections, it is simply a designated storage location for a version number.

### A.2.7.5.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | OVERWRITE_VER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.7.5.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>OVERWRITE_VER | Overwrite Version Number.<br>Points to the version number of the overwrites.h file used to initialize the device; can be used by software to identify a version-controlled set of non-default values to be written into the transceiver's register map. |

## A.2.7.6 DIGITAL TEST MUX CONTROL (DTEST_CTRL)

### A.2.7.6.1 Offset

| Register | Offset |
|---|---|
| DTEST_CTRL | 10h |

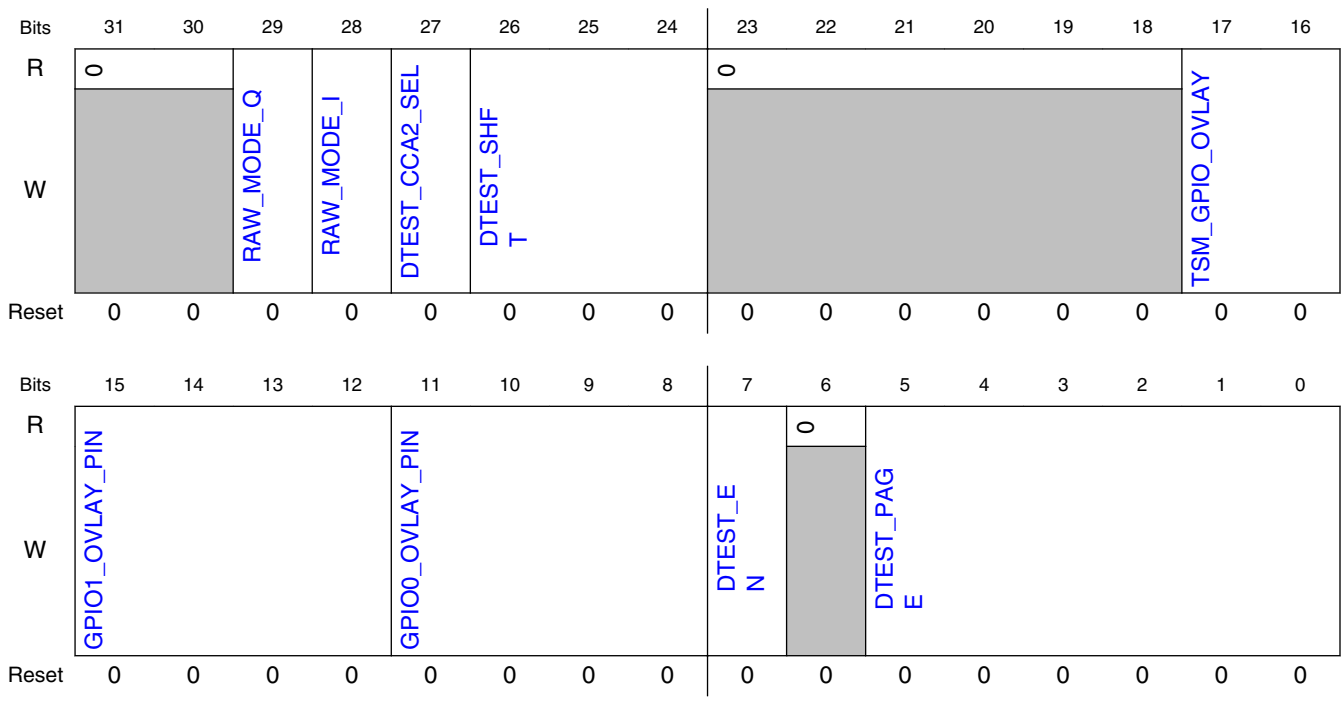
### A.2.7.6.2 Function

Digital Test Control. Allows selection and enablement of a page of DTEST signals to appear on the SoC DTEST pins.

**NOTE**

This register configures only the transceiver for DTEST mode; since DTEST pads on the SoC are multiplexed with other functions, SoC Port Pin programming is also required for each DTEST output

### A.2.7.6.3 Diagram




### A.2.7.6.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>RAW_MODE_Q | DTEST Raw Mode Enable for Q Channel<br>DTEST Q Channel Raw Mode allows raw, unfiltered ADC samples to be brought out to DTEST pins on the RXDIGIQ DTEST page. In raw mode, 2 5-bit ADC samples are concatenated into a single 12-bit DMA sample. DMA transfers these samples to memory in the same way it transfers filtered samples. Raw mode is only supported when the RX_DIG is programmed to decimate-by-2. The procedure to active Raw mode on Q channel is as follows:<br>1. RX_DIG_CTRL[RX_ADC_RAW_EN]=1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 2. RX_DIG_CTRL[RX_DEC_FILT_OSR]=0<br>3. DTEST_CTRL[DTEST_PAGE]=0x0E (RXDIGIQ)<br>4. DTEST_CTRL[DTEST_EN]=1<br>5. DTEST_CTRL[RAW_MODE_Q]=1 |
| 28<br>RAW_MODE_I | DTEST Raw Mode Enable for I Channel<br>DTEST I Channel Raw Mode allows raw, unfiltered ADC samples to be brought out to DTEST pins on the RXDIGIQ DTEST page. In raw mode, 2 5-bit ADC samples are concatenated into a single 12-bit DMA sample. DMA transfers these samples to memory in the same way it transfers filtered samples. Raw mode is only supported when the RX_DIG is programmed to decimate-by-2. The procedure to active Raw mode on I channel is as follows:<br>1. RX_DIG_CTRL[RX_ADC_RAW_EN]=1<br>2. RX_DIG_CTRL[RX_DEC_FILT_OSR]=0<br>3. DTEST_CTRL[DTEST_PAGE]=0x0E (RXDIGIQ)<br>4. DTEST_CTRL[DTEST_EN]=1<br>5. DTEST_CTRL[RAW_MODE_I]=1 |
| 27<br>DTEST_CCA2_SEL | DTEST CCA Mode 2 Selector<br>This bit selects which internal node from the CCA2 module, appears on the DTEST ZBCCA2 page in the field cca2_max_or_sym[7:0], according to the following table:<br>0b - cca2_max_or_sym[7:0] = cca2_cnt_sym[7:0]<br>1b - cca2_max_or_sym[7:0] = cca2_cnt_max[7:0] |
| 26-24<br>DTEST_SHFT | DTEST Shift Control<br>This register field DTEST_SHFT[1:0] control the amount of "arithmetic shift", which can optionally be applied to DTEST output busses. DTEST_SHFT affects only 2 DTEST output busses:<br>PLL_RIPPLE_COUNTER[18:0] on DTEST page: PLLRIPPLE (0x02). Shift is to the left (magnitude increasing)<br>RX_DIG_IQ[11:0] on DTEST page: RXDIGIQ (0x0E). Shift is to the right (magnitude decreasing)<br>The bits of PLL_RIPPLE_COUNTER[18:0], an unsigned value, are shifted by DTEST_SHFT[2:0] according to the following table<br>0: PLL_RIPPLE_COUNTER[18:5] appear on DTEST[13:0]<br>1: PLL_RIPPLE_COUNTER[17:4] appear on DTEST[13:0]<br>2: PLL_RIPPLE_COUNTER[16:3] appear on DTEST[13:0]<br>3: PLL_RIPPLE_COUNTER[15:2] appear on DTEST[13:0]<br>4: PLL_RIPPLE_COUNTER[14:1] appear on DTEST[13:0]<br>5: PLL_RIPPLE_COUNTER[13:0] appear on DTEST[13:0]<br>6: Reserved<br>7: Reserved<br>The bits of RX_DIG_IQ[11:0], a signed value, are shifted/sign-extended by DTEST_SHFT[1:0] according to the following table:<br>0: These bits appear on DTEST[13:2] : RX_DIG_IQ[11:0] (no shift)<br>1: These bits appear on DTEST[13:2] : RX_DIG_IQ[11],RX_DIG_IQ[11:1] (right shift by 1)<br>2: These bits appear on DTEST[13:2] : RX_DIG_IQ[11],RX_DIG_IQ[11],RX_DIG_IQ[11:2] (right shift by 2)<br>3: These bits appear on DTEST[13:2] : RX_DIG_IQ[11:0] (no shift) |
| 23-18<br>— | Reserved |
| 17-16<br>TSM_GPIO_OVLAY | TSM GPIO Overlay Pin Control<br>The TSM-controlled output GPIO0_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. Similarly, the TSM- |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | controlled output GPIO1_TRIG_EN can be routed to any DTEST pin, replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[0] = 1, the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[1] = 1, the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[0] = 0 and TSM_GPIO_OVLAY[1] = 0, there is no overlay, and the DTEST Page Table dictates the node that appears on each DTEST pin.<br>00b - there is no overlay, and the DTEST Page Table dictates the node that appears on each DTEST pin.<br>01b - the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear.<br>10b - the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear.<br>11b - the register GPIO0_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO0_TRIG_EN will appear, and the register GPIO1_OVLAY_PIN[3:0] selects the DTEST pin on which GPIO1_TRIG_EN will appear. |
| 15-12<br>GPIO1_OVLAY_PIN | GPIO 1 Overlay Pin<br>The TSM-controlled output GPIO1_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[1] = 1, this register selects the DTEST pin onto which GPIO1_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[1] = 0, this register is ignored, and the DTEST Page Table dictates the node that appears on each DTEST pin. |
| 11-8<br>GPIO0_OVLAY_PIN | GPIO 0 Overlay Pin<br>The TSM-controlled output GPIO0_TRIG_EN can be routed to any DTEST pin, regardless of page selection (DTEST_PAGE), replacing the nominal page-selected output for that pin. When TSM_GPIO_OVLAY[0] = 1, this register selects the DTEST pin onto which GPIO0_TRIG_EN will appear, any of DTEST0 - DTEST13. When TSM_GPIO_OVLAY[0] = 0, this register is ignored, and the DTEST Page Table dictates the node that appears on each DTEST pin. |
| 7<br>DTEST_EN | DTEST Enable<br>DTEST enable<br>0b - Disables DTEST. The DTEST pins assume their mission function.<br>1b - Enables DTEST. The contents of the selected page (DTEST_PAGE) will appear on the DTEST output pins. |
| 6<br>— | Reserved |
| 5-0<br>DTEST_PAGE | DTEST Page Selector<br>DTEST Page signal assignments<br>000000b - PLLFREQCAL<br>000001b - PLLBESTDIFF<br>000010b - PLLRIPPLE<br>000011b - PLLHPMCAL<br>000100b - PLLVCOMOD<br>000101b - PLLUNLOCK<br>000110b - PLLCYCSLIP<br>000111b - PLLCHAN<br>001000b - TXWARMUP<br>001001b - TXPOWER<br>001010b - TXFREQWORD<br>001011b - RXWARMUP<br>001100b - RXADC<br>001101b - RXDMA<br>001110b - RXDIGIQ<br>001111b - RXDMA2 |

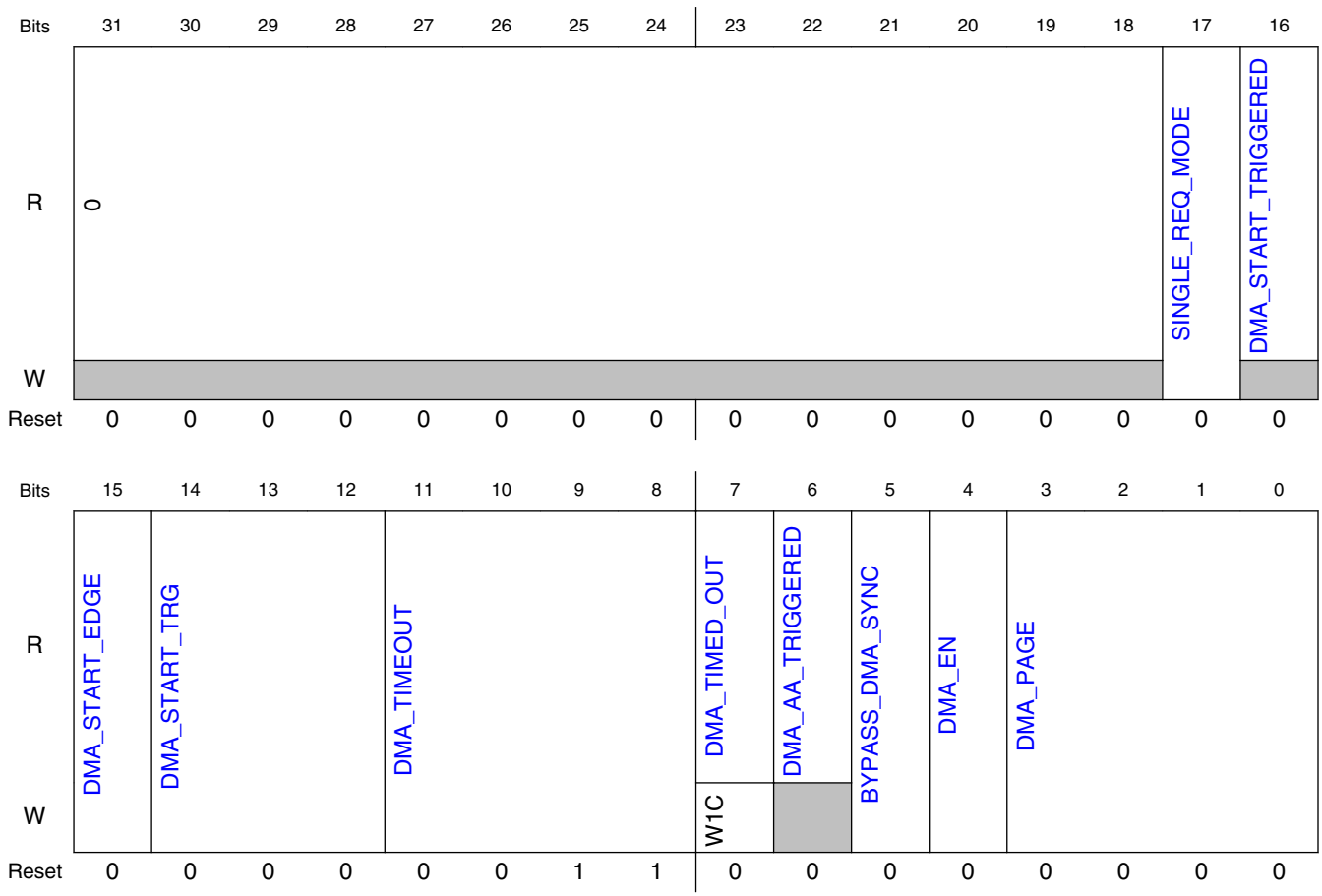
| Field | Function |
|---|---|
| | 010000b - RXINPH<br>010001b - RSSI0<br>010010b - RSSI1<br>010011b - AGC0<br>010100b - AGC1<br>010101b - DCOC0<br>010110b - DCOC1<br>010111b - DCOC2<br>011000b - DCOC3<br>011001b - TSM<br>011010b - MTTSMCAL<br>011011b - MTADV<br>011100b - MTINIT<br>011101b - MTSCAN<br>011110b - MTCONN<br>011111b - MTDTM<br>100000b - MTADVXCV<br>100001b - MTCONXCV<br>100010b - MTDTM2<br>100011b - DSM<br>100100b - PHY_FSK_STATE<br>100101b - PHY_CFO_EST_PD<br>100110b - PHY_CFO_EST_PD2<br>100111b - PHY_EARLY_LATE<br>101000b - PHY_FSK_DEMOD<br>101001b - PHY_AA_SEARCH<br>101010b - PHY_DATA_OUT<br>101011b - PHY_SAMP_TIME<br>101100b - CCA_ED_LQI<br>101101b - CCA_ED_LQI2<br>101110b - Reserved<br>101111b - Reserved<br>110000b - Reserved<br>110001b - Reserved<br>110010b - Reserved<br>110011b - Reserved<br>110100b - Reserved<br>110101b - Reserved<br>110110b - Reserved<br>110111b - Reserved<br>111000b - Reserved<br>111001b - Reserved<br>111010b - RCCAL<br>111011b - AUXPLLFCAL<br>111100b - GENFSKTX<br>111101b - GENFSKRX<br>111110b - GENFSKSTATE<br>111111b - GENFILTER |

## A.2.7.7 TRANSCEIVER DMA CONTROL (DMA_CTRL)

## A.2.7.7.1 Offset

| Register | Offset |
|---|---|
| DMA_CTRL | 14h |

## A.2.7.7.2 Diagram




## A.2.7.7.3 Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17<br>SINGLE_REQ_MODE | DMA Single Request Mode<br>Configures the transceiver to transfer the entire block of DMA data with only a single initial request (ipd_req_radio_rx). The DMA2 controller must be configured accordingly. In this mode, the transceiver will use **ips_xfr_wait** to pace the individual transactions. Single Request Mode should not be used with |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | DMA Pages 11, 12, or 13, because the data rate is too low and therefore ips_xfr_wait would remain assserted for excessively long periods.<br>0b - Disable Single Request Mode. The transceiver will assert ipd_req_radio_rx whenever it has a new sample ready for transfer.<br>1b - Enable Single Request Mode. A single initial request by the transceiver will transfer the entire DMA block of data |
| 16<br>DMA_START_TRIGGERED | DMA Start Trigger Occurred<br>This read-only status bit becomes set, when, during a DMA session (DMA_PAGE > 0), the trigger source selected by DMA_START_TRG[2:0] occurs, with the edge sensitivity selected by DMA_START_EDGE. To clear this bit, set DMA_PAGE=0. |
| 15<br>DMA_START_EDGE | DMA Start Trigger Edge Selector<br>DMA_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 14-12<br>DMA_START_TRG | DMA Start Trigger Selector<br>The DMA Start Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| 11-8<br>DMA_TIMEOUT | DMA Timeout<br>In DMA Single Request Mode, adverse consequences may result if the transceiver's ips_xfr_wait signal is asserted for an excessively long period of time, which could occur due to mis-programming of the DMA2 controller, and/or the transceiver itself. DMA Timeout forces the transceiver's ips_xfr_wait low after *N* microseconds and sets the **DMA_TIMED_OUT** status bit, where *N*=DMA_TIMEOUT. DMA_TIMEOUT is only relavent when SINGLE_REQ_MODE=1, otherwise the transceiver does not assert ips_xfr_wait. |
| 7<br>DMA_TIMED_OUT | DMA Transfer Timed Out<br>Status bit indicates that the transceiver DMA, while operating in Single Request Mode, asserted ips_xfr_wait for a period in excess of the programmed **DMA_TIMEOUT** setting, resulting in a timeout condition where the transceiver has forced ips_xfr_wait low. After a timeout, before resuming DMA operations, set DMA_PAGE=0 to disable DMA, and write 1 to this bit to clear it.<br>0b - A DMA timeout has not occurred<br>1b - A DMA timeout has occurred in Single Request Mode since the last time this bit was cleared |
| 6<br>DMA_AA_TRIGGERED | DMA Access Address triggered<br>This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of DMA transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DMA_PAGE to initiate DMA transfers. This bit is intended for use with DMA_PAGE=13, but is available on all pages. Its usage is optional. |
| 5<br>BYPASS_DMA_SYNC | Bypass External DMA Synchronization<br>When using transceiver DMA with SINGLE_REQ_MODE=0, synchronization external to the transceiver must be bypassed in order to minimize bus access latency. Using DMA in this mode also requires that the MCU and transceiver both are configured to operate in the RF Oscillator clock domain.<br>0b - Don't Bypass External Synchronization. Use this setting if SINGLE_REQ_MODE=1.<br>1b - Bypass External Synchronization. This setting is mandatory if SINGLE_REQ_MODE=0. |
| 4 | DMA Enable |

*Table continues on the next page...*

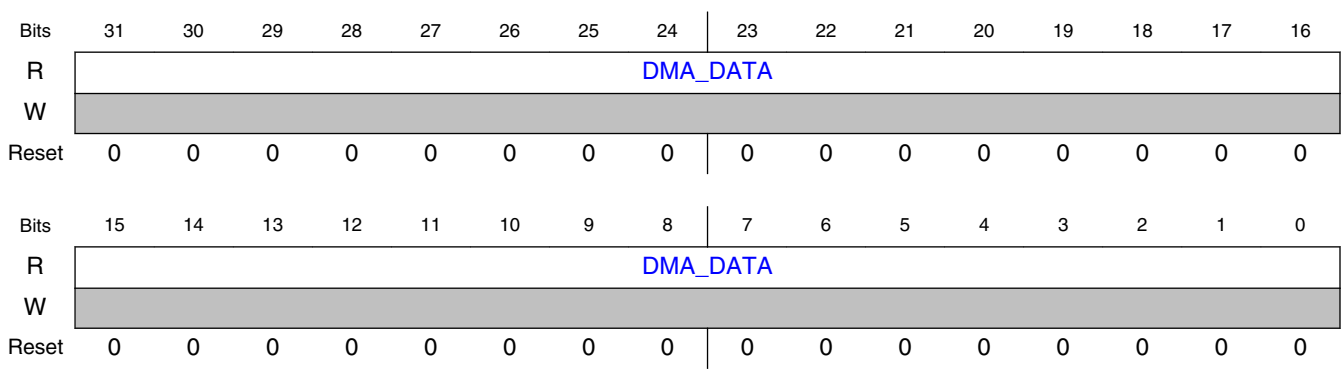
| Field | Function |
|---|---|
| DMA_EN | Setting DMA_EN=1 enables clock gating to the Transciever DMA engine. This bit should be set prior to the start of a DMA session, and remain set for the duration of the session. |
| 3-0<br>DMA_PAGE | Transceiver DMA Page Selector<br>Selects the page of receiver data for storage to system memory when using Transceiver DMA Debug Mode. Setting this register to a non-zero value enables the DMA interface logic. Setting this register to zero disables the interface. **Note:** DMA_PAGE should be set to a non-zero value to begin an acquisition, only after the start trigger has been programmed (DMA_START_TRG), if a trigger is desired. The available DMA pages are listed below.<br>0000b - DMA Idle<br>0001b - RX_DIG I and Q<br>0010b - RX_DIG I Only<br>0011b - RX_DIG Q Only<br>0100b - RAW ADC I and Q<br>0101b - RAW ADC I Only<br>0110b - RAW ADC Q only<br>0111b - DC Estimator I and Q<br>1000b - DC Estimator I Only<br>1001b - DC Estimator Q only<br>1010b - RX_DIG Phase Output<br>1011b - Reserved<br>1100b - Demodulator Soft Decision<br>1101b - Demodulator Data Output<br>1110b - Demodulator CFO Phase Output<br>1111b - Reserved |

## A.2.7.8 TRANSCEIVER DMA DATA (DMA_DATA)

### A.2.7.8.1 Offset

| Register | Offset |
|---|---|
| DMA_DATA | 18h |

### A.2.7.8.2 Diagram



### A.2.7.8.3 Fields

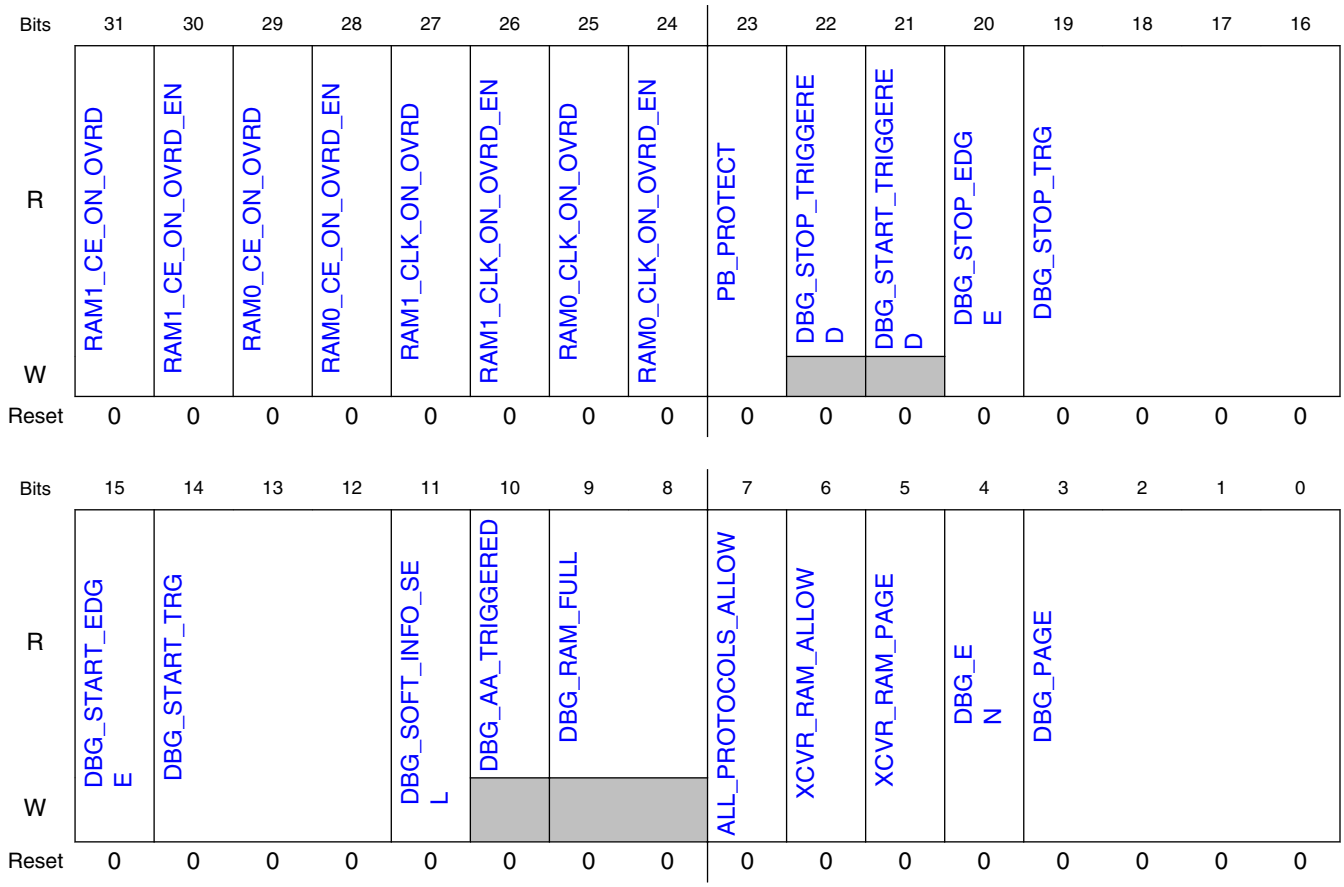
| Field | Function |
|---|---|
| 31-0<br>DMA_DATA | DMA Data Register<br>This register is the singular address location at which the SoC DMA controller accesses samples from the transceiver for transfer to system memory. During DMA operation, the contents of this register depend on the DMA Page selected. The DMA Data register is intended for DMA purposes only, not Host IPS accesses, but Host IPS bus access to this register is not prohibited. |

## A.2.7.9 PACKET RAM CONTROL (PACKET_RAM_CTRL)

### A.2.7.9.1 Offset

| Register | Offset |
|---|---|
| PACKET_RAM_CTRL | 1Ch |

## A.2.7.9.2 Diagram




## A.2.7.9.3 Fields

| Field | Function |
|---|---|
| 31<br>RAM1_CE_ON_OVRD | Override value for RAM1 CE (Chip Enable)<br>When RAM1_CE_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM1 CE. This bit is ignored when RAM1_CE_ON_OVRD_EN=0. |
| 30<br>RAM1_CE_ON_OVRD_EN | Override control for RAM1 CE (Chip Enable)<br>0b - Normal operation.<br>1b - Use the state of RAM1_CE_ON_OVRD to override the RAM1 CE. |
| 29<br>RAM0_CE_ON_OVRD | Override value for RAM0 CE (Chip Enable)<br>When RAM0_CE_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM0 CE. This bit is ignored when RAM0_CE_ON_OVRD_EN=0. |
| 28<br>RAM0_CE_ON_OVRD_EN | Override control for RAM0 CE (Chip Enable)<br>0b - Normal operation.<br>1b - Use the state of RAM0_CE_ON_OVRD to override the RAM0 CE. |
| 27<br>RAM1_CLK_ON_OVRD | Override value for RAM1 Clock Gate Enable<br>When RAM1_CLK_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM1 Clock Gate Enable. This bit is ignored when RAM1_CLK_ON_OVRD_EN=0. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 26<br>RAM1_CLK_ON_OVRD_EN | Override control for RAM1 Clock Gate Enable<br>0b - Normal operation.<br>1b - Use the state of RAM1_CLK_ON_OVRD to override the RAM1 Clock Gate Enable. |
| 25<br>RAM0_CLK_ON_OVRD | Override value for RAM0 Clock Gate Enable<br>When RAM0_CLK_ON_OVRD_EN=1, this value overrides the mission mode state of the RAM0 Clock Gate Enable. This bit is ignored when RAM0_CLK_ON_OVRD_EN=0. |
| 24<br>RAM0_CLK_ON_OVRD_EN | Override control for RAM0 Clock Gate Enable<br>0b - Normal operation.<br>1b - Use the state of RAM0_CLK_ON_OVRD to override the RAM0 Clock Gate Enable. |
| 23<br>PB_PROTECT | Packet Buffer Protect<br>Protect Packet Buffer contents against overwriting by the next received packet. Applies to all protocols except BLE<br>0b - Incoming received packets overwrite Packet Buffer RX contents (default)<br>1b - Incoming received packets are blocked from overwriting Packet Buffer RX contents |
| 22<br>DBG_STOP_TRIGGERED | Packet RAM Debug Stop Triggered<br>This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by DBG_STOP_TRG[3:0] occurs, with the edge sensitivity selected by DBG_STOP_EDGE. To clear this bit, set DBG_PAGE=0. |
| 21<br>DBG_START_TRIGGERED | Packet RAM Debug Start Triggered<br>This read-only status bit becomes set, when, during a Packet RAM Debug session (DBG_PAGE > 0), the trigger source selected by DBG_START_TRG[2:0] occurs, with the edge sensitivity selected by DBG_START_EDGE. To clear this bit, set DBG_PAGE=0. |
| 20<br>DBG_STOP_EDGE | Packet RAM Debug Stop Trigger Edge Selector<br>DBG_STOP_EDGE selects the edge sensitivity (rising or falling) of the selected stop-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 19-16<br>DBG_STOP_TRG | Packet RAM Debug Stop Trigger Selector<br>The Packet RAM Debug Stop Trigger, if desired, can be selected from the sources in the following table:<br>0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare3_en<br>8: TSM: pll unlock<br>9: BLE: crc_err_count_incr (increment DTM RX bad packet counter)<br>10: CRC FAIL: 802.15.4 or GEN_FSK only, selected by PROTOCOL[3:0]<br>11: HEADER FAIL: GEN_FSK only<br>12-15: Reserved |
| 15<br>DBG_START_EDGE | Packet RAM Debug Start Trigger Edge Selector<br>DBG_START_EDGE selects the edge sensitivity (rising or falling) of the selected start-trigger.<br>0b - Trigger fires on a rising edge of the selected trigger source<br>1b - Trigger fires on a falling edge of the selected trigger source |
| 14-12<br>DBG_START_TRG | Packet RAM Debug Start Trigger Selector<br>The Packet RAM Debug Start Trigger, if desired, can be selected from the sources in the following table: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0: no trigger<br>1: FSK PHY: gfsk_preamble_found<br>2: FSK PHY: aa_sfd_matched<br>3: ZBDEMOD: rx_preamble_det<br>4: ZBDEMOD: rx_sfd_detect<br>5: RXDIG: agc_dcoc_gain_chg<br>6: TSM: rx_dig_en<br>7: TSM: tsm_spare2_en |
| 11<br>DBG_SOFT_INFO_SEL | Packet RAM Debug PHY Soft Info Output Selector<br>When acquiring data in Packet RAM Debug Mode with the DBG_PAGE=DMDSOFT page, this bit selects the signal from the PHY used to capture the soft decision data.<br>0b - PHY output bit_valid_int is used to capture soft decision data<br>1b - PHY output demod_vout is used to capture soft decision data |
| 10<br>DBG_AA_TRIGGERED | Packet Ram Debug Access Address triggered<br>This status bit becomes set when the PHY indicates that a Network Address match has occurred. In some cases it may be desirable to delay the start of Packet RAM debug transfers until such a match occurs; in such cases, software can poll this bit, and once it becomes set, immediately select the desired DBG_PAGE to initiate data transfers to Packet RAM. This bit is intended for use with DBG_PAGE=13, but is available on all pages. Its usage is optional. |
| 9-8<br>DBG_RAM_FULL | DBG_RAM_FULL[1:0]<br>Status Bits indicating that Packet RAM0 (or RAM1) is full, and the Packet RAM Debug engine has attempted to write another word to that RAM. This, and any subsequent write attempts, will not alter the contents of the RAM, once it has reached capacity. Set DBG_PAGE=0 to clear the DBG_RAM_FULL[1:0] bits.<br>**NOTE:** If a Debug Stop Trigger is selected (DBG_STOP_TRG > 0), writes to RAM do not stop when DBG_RAM_FULL bit(s) become set; instead, writes continue after an address wrap-around to 0, and will continue indefinitely until either a stop trigger condition occurs, or the debug session is ended by taking DBG_PAGE=0. The DBG_RAM_FULL[1:0] remain a valid indicator that the RAM(s) has reached full status at least once during the debug session.<br>00b - Neither Packet RAM0 nor RAM1 is full<br>1xb - Packet RAM1 has been filled to capacity.<br>x1b - Packet RAM0 has been filled to capacity. |
| 7<br>ALL_PROTOCOLS_ALLOW | Allow IPS bus access to Packet RAM for any protocol at any time.<br>Each supported protocol has an associated IPS bus, with which it can access protocol-specific registers, as well as Packet RAM. Normally IPS bus access to the Packet RAM is restricted to the protocol currently selected by the XCVR_CTRL[PROTOCOL] register, and access attempts by other protocols will result in an assert of **ips_xfr_err** on the offending IPS interface. When ALL_PROTOCOLS_ALLOW=1, these inhibits are removed, and any IPS bus can access Packet RAM at any time without any error signalling.<br>0b - IPS bus access to Packet RAM is restricted to the protocol engine currently selected by XCVR_CTRL[PROTOCOL].<br>1b - All IPS bus access to Packet RAM permitted, regardless of XCVR_CTRL[PROTOCOL] setting |
| 6<br>XCVR_RAM_ALLOW | Allow Packet RAM Transceiver Access<br>This bit must be set before performing accesses to the Packet RAM using transceiver address space. Transceiver space accesses to Packet RAM are intended for debug purposes only; the individual protocol engines, and the associated IPS busses, have exclusive access to Packet RAM in mission modes. Transceiver space accesses to Packet RAM include direct accesses to RAM at transceiver addresses 0x700 - 0xFFF, as well as Packet RAM Debug Mode. When this bit is set, control of RAM clock gating and RAM chip enables are forced on continuously, taking these controls away from the protocol engines.<br>0b - Protocol Engines, and associated IPS busses, have exclusive access to Packet RAM (mission mode)<br>1b - Transceiver-space access to Packet RAM, including Packet RAM debug mode, are allowed |
| 5 | RAM Page Selector for XCVR Access |

*Table continues on the next page...*

| Field | Function |
|---|---|
| XCVR_RAM_PAGE | This bit selects which of the 2 Packet RAM blocks is mapped into XCVR address space starting at XCVR_BASE + 0x700.<br>0b - RAM0 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF<br>1b - RAM1 is mapped into XCVR address space, between XCVR_BASE + 0x700, and XCVR_BASE + 0xFFF |
| 4<br>DBG_EN | Packet RAM Debug Mode Enable<br>Setting DBG_EN=1 enables clock gating to the Packet RAM Debug engine. This bit should be set prior to the start of a Debug session, and remain set for the duration of the session. |
| 3-0<br>DBG_PAGE | Packet RAM Debug Page Selector<br>Selects the page of receiver data for storage to Packet RAM using Transceiver Packet RAM Debug Mode. Setting this register to a non-zero value enables the Packet RAM Debug Mode interface logic. Setting this register to zero disables the interface. **Note:** DBG_PAGE should be set to a non-zero value to begin an acquisition, only after the start and stop triggers have been programmed (DBG_START_TRG and DBG_STOP_TRG), if a trigger is desired. The available RAM Debug pages are listed below.<br>0000b - Packet RAM Debug Mode Idle<br>0001b - RX_DIG I and Q<br>0010b - Reserved<br>0011b - Reserved<br>0100b - RAW ADC I and Q<br>0101b - Reserved<br>0110b - Reserved<br>0111b - DC Estimator I and Q<br>1000b - Reserved<br>1001b - Reserved<br>1010b - RX_DIG Phase Output<br>1011b - Reserved<br>1100b - Demodulator Soft Decision<br>1101b - Demodulator Data Output<br>1110b - Demodulator CFO Phase Output<br>1111b - Reserved |

## A.2.7.10 PACKET RAM DEBUG RAM STOP ADDRESS (RAM_STOP_ADDR)

### A.2.7.10.1 Offset

| Register | Offset |
|---|---|
| RAM_STOP_ADDR | 20h |

### A.2.7.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | RAM1_STOP_ADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | RAM0_STOP_ADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.7.10.3 Fields

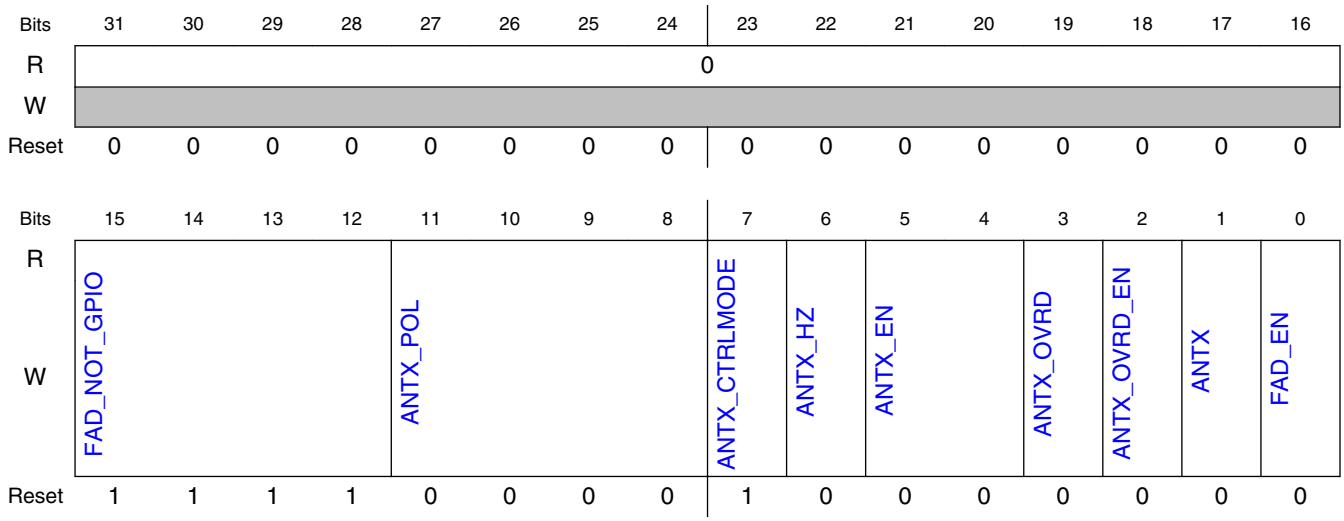
| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-16<br>RAM1_STOP_ADDR | RAM1 Stop Address<br>After a Packet RAM Debug Mode acquisition, RAM1_STOP_ADDR represents the last (most recent) RAM1 location filled with sample data before the acquision terminated. The acquisition terminates when a stop-trigger occurs, or due to a manual termination by setting DBG_PAGE=0. RAM1_STOP_ADDR is in units of RAM "words", where each word represents 2 bytes of XCVR RAM address space. |
| 15-11<br>— | Reserved |
| 10-0<br>RAM0_STOP_ADDR | RAM0 Stop Address<br>After a Packet RAM Debug Mode acquisition, RAM0_STOP_ADDR represents the last (most recent) RAM0 location filled with sample data before the acquision terminated. The acquisition terminates when a stop-trigger occurs, or due to a manual termination by setting DBG_PAGE=0. RAM0_STOP_ADDR is in units of RAM "words", where each word represents 2 bytes of XCVR RAM address space. |

## A.2.7.11 FAD CONTROL (FAD_CTRL)

### A.2.7.11.1 Offset

| Register | Offset |
|---|---|
| FAD_CTRL | 24h |

## A.2.7.11.2 Diagram




## A.2.7.11.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>FAD_NOT_GPIO | FAD versus GPIO Mode Selector<br>**xxx1**: The ANT_A pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**xxx0**: The ANT_A pad is controlled directly by the TSM output gpio0_trig_en<br>**xx1x**: The ANT_B pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**xx0x**: The ANT_B pad is controlled directly by the TSM output gpio1_trig_en<br>**x1xx**: The TX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**x0xx**: The TX_SWITCH pad is controlled directly by the TSM output gpio2_trig_en<br>**1xxx**: The RX_SWITCH pad is controlled by the antenna-selection algorithm, modified by the register settings above {ANTX_CTRLMODE, ANTX_EN, ANTX_POL}<br>**0xxx**: The RX_SWITCH pad is controlled directly by the TSM output gpio3_trig_en |
| 11-8<br>ANTX_POL | FAD Antenna Controls Polarity<br>Control the polarity of the FAD pins:<br>ANTX_POL<0> : This bit currently has no functionality<br>ANTX_POL<1> : This bit currently has no functionality<br>ANTX_POL<2>=1 : invert the TX_SWITCH output<br>ANTX_POL<3>=1 : invert the RX_SWITCH output |
| 7 | Antenna Diversity Control Mode |

*Table continues on the next page...*

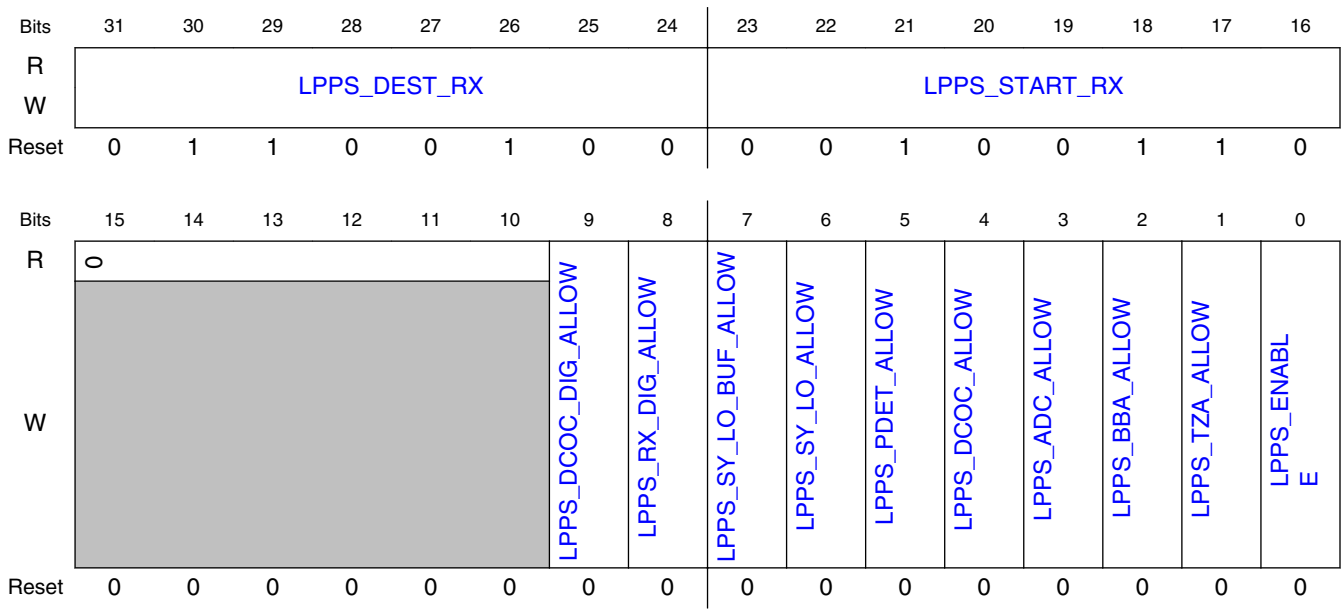
| Field | Function |
|---|---|
| ANTX_CTRLMODE | When ANTX_CTRLMODE=1 (dual mode):<br>ANT_A=NOT(ANTX) AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>ANT_B=ANTX AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>TX_SWITCH=GPIO2_TRIG_EN<br>RX_SWITCH=GPIO3_TRIG_EN<br><br>When ANTX_CTRLMODE=0 (single mode):<br>ANT_A=NOT(ANTX) AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>ANT_B=ANTX AND (GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br>TX_SWITCH=GPIO2_TRIG_EN<br>RX_SWITCH=(GPIO3_TRIG_EN OR GPIO2_TRIG_EN)<br><br>GPIO2_TRIG_EN and GPIO3_TRIG_EN are outputs of the Transceiver Sequence Manager (TSM). The TSM timing registers associated with GPIO2_TRIG_EN and GPIO3_TRIG_EN should be programmed with the desired TX_SWITCH and RX_SWITCH timing, before enabling Fast Antenna Diversity. |
| 6<br>ANTX_HZ | FAD PAD Tristate Control<br>0b - ANT_A, ANT_B, RX_SWITCH and TX_SWITCH are actively driven outputs.<br>1b - Antenna controls high impedance- Set ANT_A, ANT_B, RX_SWITCH and TX_SWITCH in high impedance. |
| 5-4<br>ANTX_EN | FAD Antenna Controls Enable<br>For any of the 4 FAD-related outputs {ANT_A, ANT_B, TX_SWITCH, RX_SWITCH}, when the associated register control bit (FAD_NOT_GPIO[x]=1) puts that output in FAD mode, ANTX_EN[1:0] determines which pairs of FAD related outputs are enabled:<br>00b - all disabled (held low)<br>01b - only RX/TX_SWITCH enabled<br>10b - only ANT_A/B enabled<br>11b - all enabled |
| 3<br>ANTX_OVRD | Antenna State Override Value<br>When ANTX_OVRD_EN=1, the FAD antenna state (ANTX) is overriden by the value of this register; otherwise, antenna state is determined by the 802.15.4 FAD state machine, if XCVR_CTRL[DEMOD_SEL]=0b10, or the FSK PHY FAD state machine if XCVR_CTRL[DEMOD_SEL]=0b01. |
| 2<br>ANTX_OVRD_EN | Antenna State Override Enable<br>When ANTX_OVRD_EN=1, the FAD antenna state (ANTX) is overriden by the value of the ANTX_OVRD register; otherwise, antenna state is determined by the 802.15.4 FAD state machine, if XCVR_CTRL[DEMOD_SEL]=0b10, or the FSK PHY FAD state machine if XCVR_CTRL[DEMOD_SEL]=0b01. |
| 1<br>ANTX | Antenna Selection State<br>If FAD_EN=0, the ANTX bit is used to take manual (software) control of the antenna selection, overriding the FAD state machine; in this case, the readback value of ANTX is whatever was last written by the host.<br>If FAD_EN=1, the FAD state machine controls antenna selection, and the readback value of ANTX reflects the machine-selected antenna. |
| 0<br>FAD_EN | Fast Antenna Diversity Enable<br>0b - Fast Antenna Diversity disabled<br>1b - Fast Antenna Diversity enabled |

## A.2.7.12 LOW POWER PREAMBLE SEARCH CONTROL (LPPS_CTRL)

## A.2.7.12.1 Offset

| Register | Offset |
|---|---|
| LPPS_CTRL | 28h |

## A.2.7.12.2 Diagram




## A.2.7.12.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LPPS_DEST_RX | LPPS Fast TSM RX Warmup "Jump-to" Point<br>During a LPPS Fast RX Warmup, this value represents the ending TSM count for the portion of the warmup that is to be skipped over. In an LPPS Fast RX Warmup, LPPS_START_RX[7:0] will never be reached and will instead be replaced by the contents of this register, thereby executing the jump.<br>Example:<br>LPPS_START_RX = 10<br>LPPS_DEST_RX = 15<br>The TSM will count as follows: … 7,8,9,15,16,17 … |
| 23-16<br>LPPS_START_RX | LPPS Fast TSM RX Warmup "Jump-from" Point<br>During a LPPS Fast RX Warmup, this value represents the starting TSM count for the portion of the warmup that is to be skipped over. In an LPPS Fast RX Warmup, this count will never be reached; its place in the TSM warmup will be replaced by the contents of LPPS_DEST_RX[7:0], thereby executing the jump. |
| 15-10<br>— | Reserved |

*Table continues on the next page...*

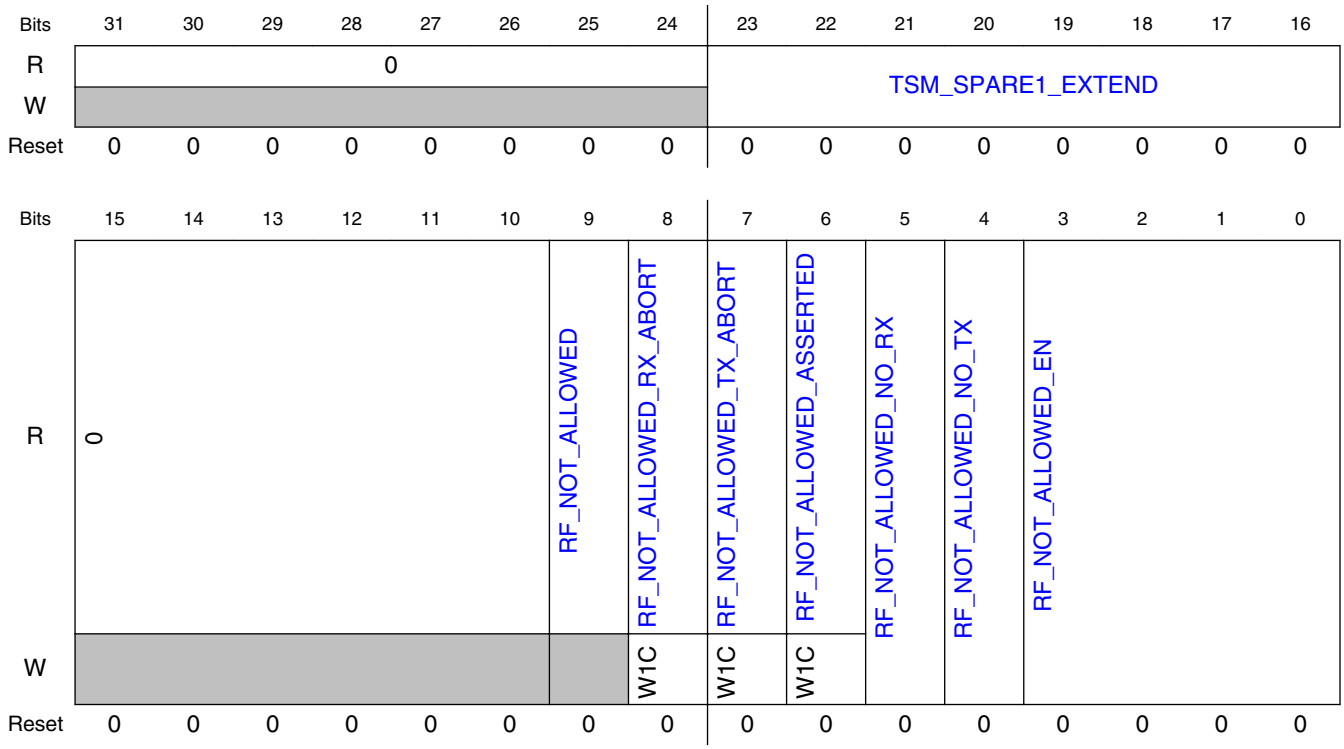
| Field | Function |
|---|---|
| 9<br>LPPS_DCOC_DIG_ALLOW | LPPS_DCOC_DIG_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: dcoc_en |
| 8<br>LPPS_RX_DIG_ALLOW | LPPS_RX_DIG_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_dig_en |
| 7<br>LPPS_SY_LO_BUF_ALLOW | LPPS_SY_LO_BUF_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_buf_en |
| 6<br>LPPS_SY_LO_ALLOW | LPPS_SY_LO_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: sy_lo_rx_en |
| 5<br>LPPS_PDET_ALLOW | LPPS_PDET_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_pdet_en, rx_tza_pdet_en |
| 4<br>LPPS_DCOC_ALLOW | LPPS_DCOC_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_dcoc_en, rx_tza_dcoc_en |
| 3<br>LPPS_ADC_ALLOW | LPPS_ADC_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_adc_en, rx_adc_i_en, rx_adc_q_en |
| 2<br>LPPS_BBA_ALLOW | LPPS_BBA_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_bba_i_en, rx_bba_q_en |
| 1<br>LPPS_TZA_ALLOW | LPPS_TZA_ALLOW<br>When set to one, allows the following TSM outputs to be controlled by the LPPS state machine: rx_lna_en, rx_tza_i_en, rx_tza_q_en |
| 0<br>LPPS_ENABLE | LPPS_ENABLE<br>Master enable for LPPS mode. Allows correlators to be duty-cycled during Preamble Search, and selected analog and digital blocks to be duty-cycled simultaneously. |

# A.2.7.13 COEXISTENCE CONTROL (COEX_CTRL)

## A.2.7.13.1 Offset

| Register | Offset |
|---|---|
| COEX_CTRL | 2Ch |

## A.2.7.13.2 Diagram




## A.2.7.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>TSM_SPARE1_EXTEND | TSM_SPARE1_EX Extension Duration<br>When TSM output TSM_SPARE1_EN is enabled to assert during any TX or RX sequence and TSM_SPARE1_EXTEND[7:0] > 0, this register specifies the number of microseconds for which TSM_SPARE1_EN is to remain asserted beyond the end of the TSM warmdown. This extension is intended to close any gap in the coexistence output RF_ACTIVE that may occur between consecutive TX/RX or RX/TX TSM operations. |
| 15-10<br>— | Reserved |
| 9<br>RF_NOT_ALLOWED | RF_NOT_ALLOWED<br>Reflects the instantaneous state of the RF_NOT_ALLOWED pin, synchronized into the RF OSC clock domain. |
| 8<br>RF_NOT_ALLOWED_RX_ABORT | RF_NOT_ALLOWED_RX_ABORT<br>0b - A RX abort due to assertion on RF_NOT_ALLOWED has not occurred<br>1b - A RX abort due to assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 7 | RF_NOT_ALLOWED_TX_ABORT<br>0b - A TX abort due to assertion on RF_NOT_ALLOWED has not occurred |

*Table continues on the next page...*

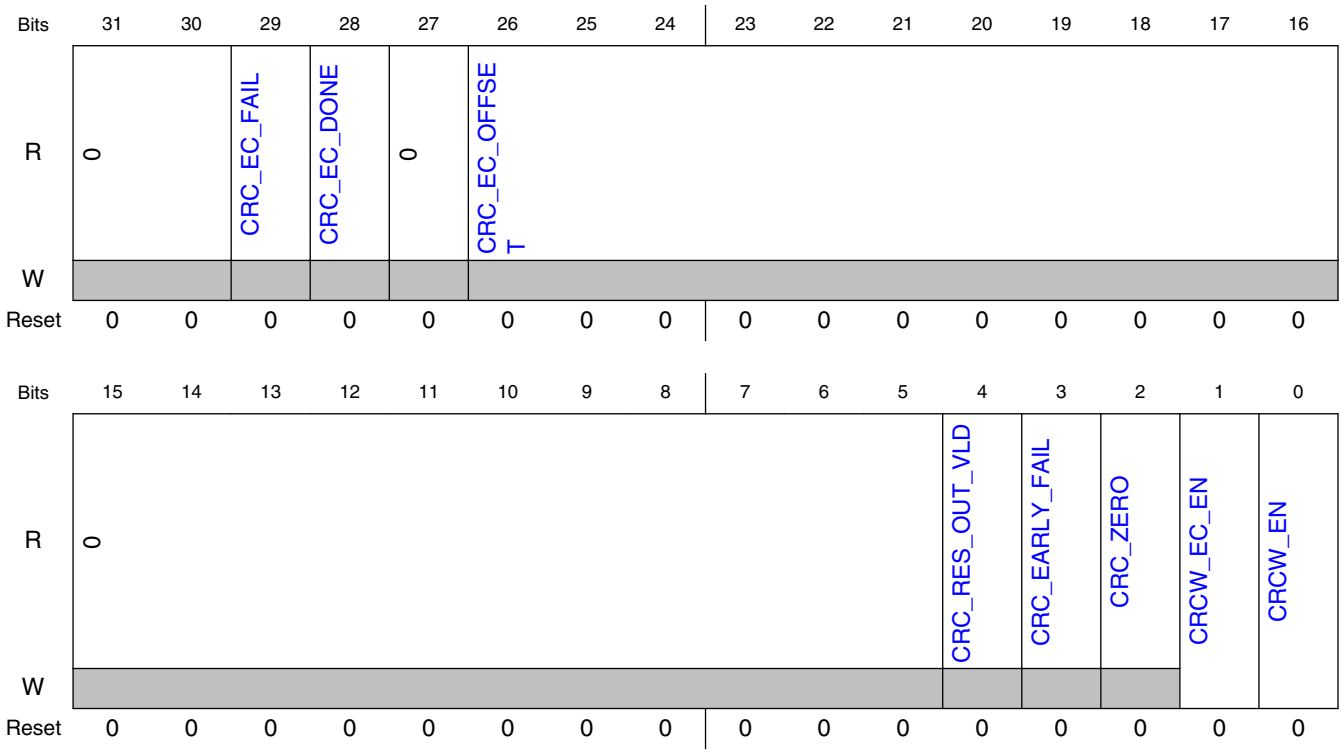
| Field | Function |
|---|---|
| RF_NOT_ALLOWED_TX_ABORT | 1b - A TX abort due to assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 6<br>RF_NOT_ALLOWED_ASSERTED | RF_NOT_ALLOWED_ASSERTED<br>0b - Assertion on RF_NOT_ALLOWED has not occurred<br>1b - Assertion on RF_NOT_ALLOWED has occurred since the last time this bit was cleared |
| 5<br>RF_NOT_ALLOWED_NO_RX | RF_NOT_ALLOWED_NO_RX<br>0b - Assertion on RF_NOT_ALLOWED has no effect on RX<br>1b - Assertion on RF_NOT_ALLOWED can abort RX |
| 4<br>RF_NOT_ALLOWED_NO_TX | RF_NOT_ALLOWED_NO_TX<br>0b - Assertion on RF_NOT_ALLOWED has no effect on TX<br>1b - Assertion on RF_NOT_ALLOWED can abort TX |
| 3-0<br>RF_NOT_ALLOWED_EN | RF_NOT_ALLOWED PER-LINK-LAYER ENABLE<br>The coexistence input RF_NOT_ALLOWED can be enabled to selectively abort TX or RX sequences, with individual enables for each supported protocol, subject also to the state of the RF_NOT_ALLOWED_NO_TX and RF_NOT_ALLOWED_NO_RX control bits, according to the following table:<br>**xxx1**: RF_NOT_ALLOWED assertions are enabled to abort BLE TX and RX sequences<br>**xxx0**: RF_NOT_ALLOWED assertions are not enabled to abort BLE TX and RX sequences<br>**1xxx**: RF_NOT_ALLOWED assertions are enabled to abort GENERIC_FSK TX and RX sequences<br>**0xxx**: RF_NOT_ALLOWED assertions are not enabled to abort GENERIC_FSK TX and RX sequences |

## A.2.7.14 CRC/WHITENER CONFIG REGISTER (CRCW_CFG)

### A.2.7.14.1 Offset

| Register | Offset |
|---|---|
| CRCW_CFG | 30h |

## A.2.7.14.2 Diagram




## A.2.7.14.3 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>CRC_EC_FAIL | CRC error correction fail<br>This signal is cleared when *crc_init* is asserted. It is set after the completion of a CRC error correction calculation that does not find a valid data correction solution or if error correction was disabled and the CRC check found an error. |
| 28<br>CRC_EC_DONE | CRC error correction done<br>This signal is cleared when *crc_init* is asserted. It is set after the error correction logic completes processing the syndrome. It is immediately set after a packet is received if no error was detected or if error correction is disabled. Otherwise, it can take up to N system clocks after the packet is received to assert, with N = (number of bits used in the CRC calculation, including the CRC value). |
| 27<br>— | Reserved |
| 26-16<br>CRC_EC_OFFSET | CRC error correction offset<br>This value provides the byte offset within the data packet to which the CRC error correction mask should be XOR-ed. This value is valid if the *crc_ec_done* signal is asserted and the *crc_ec_fail* signal is not asserted. The offset includes only the bytes used in the CRC calculation, including the CRC value. |
| 15-5 | Reserved |

*Table continues on the next page...*

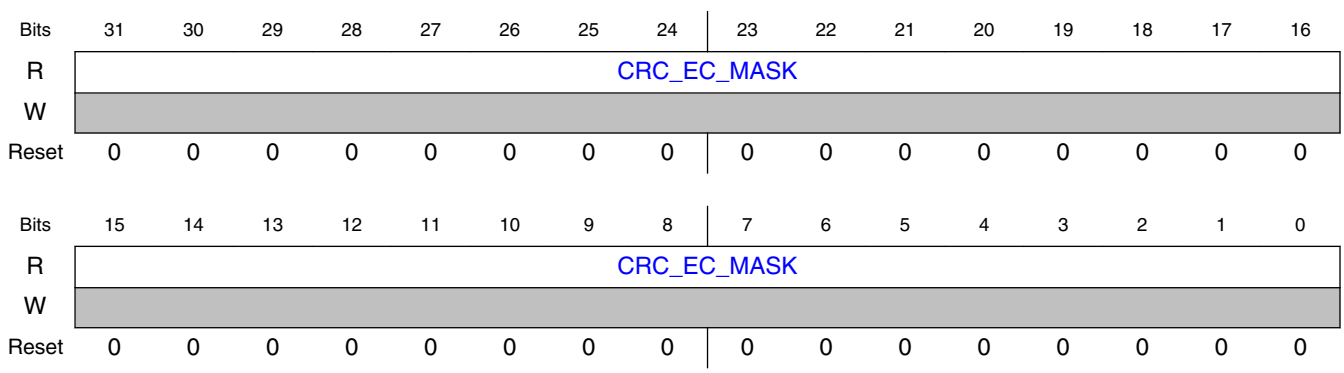
| Field | Function |
|---|---|
| — | |
| 4<br>CRC_RES_OUT_VLD | CRC result output valid<br>CRC result output valid |
| 3<br>CRC_EARLY_FAIL | CRC error correction fail<br>This signal is cleared when *crc_init* is asserted. It is set after the completion of a CRC error correction calculation that does not find a valid data correction solution or if error correction was disabled and the CRC check found an error. |
| 2<br>CRC_ZERO | CRC zero<br>This signal is asserted at any time that the CRC shift register contains a value of zero |
| 1<br>CRCW_EC_EN | CRC Error Correction Enable<br>CRC Error Correction Enable |
| 0<br>CRCW_EN | CRC calculation enable<br>Input data bits loaded with this signal asserted are used in the CRC calculation. |

# A.2.7.15 CRC ERROR CORRECTION MASK (CRC_EC_MASK)

## A.2.7.15.1 Offset

| Register | Offset |
|---|---|
| CRC_EC_MASK | 34h |

## A.2.7.15.2 Diagram



### A.2.7.15.3 Fields

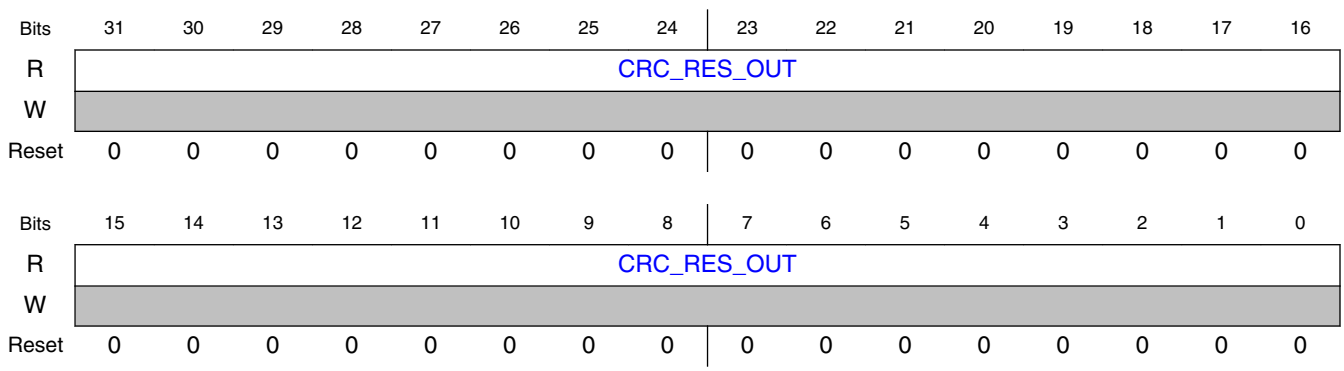
| Field | Function |
|---|---|
| 31-0<br>CRC_EC_MASK | CRC error correction mask<br>This value provides a 32-bit XOR mask that must be applied to the input data packet to correct the burst errors detected by the error correction calculation. This value is valid if the *crc_ec_done* signal is asserted and the *crc_ec_fail* signal is not asserted. |

## A.2.7.16 CRC RESULT (CRC_RES_OUT)

### A.2.7.16.1 Offset

| Register | Offset |
|---|---|
| CRC_RES_OUT | 38h |

### A.2.7.16.2 Diagram




### A.2.7.16.3 Fields

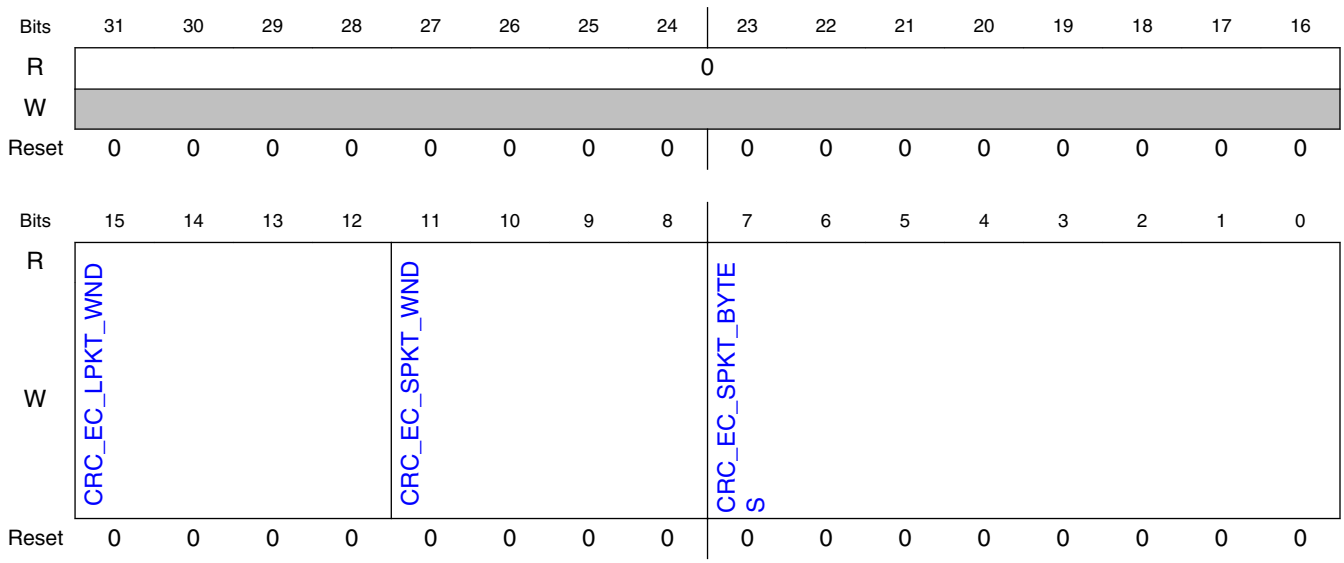
| Field | Function |
|---|---|
| 31-0<br>CRC_RES_OUT | CRC result output<br>This bus provides the instantaneous value of the CRC shift register. |

## A.2.7.17 CRC/WHITENER CONFIG 2 REGISTER (CRCW_CFG2)

## A.2.7.17.1 Offset

| Register | Offset |
|---|---|
| CRCW_CFG2 | 3Ch |

## A.2.7.17.2 Diagram




## A.2.7.17.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>CRC_EC_LPKT_WND | Error correction long packet burst error window<br>This value determines the largest burst error corrected when long packets are processed. A value of zero will disable long packet error correction. The max value is 15. |
| 11-8<br>CRC_EC_SPKT_WND | Error correction short packet burst error window<br>This value determines the largest burst error corrected when short packets are processed. A value of zero will disable short packet error correction. The max value is 15. |
| 7-0<br>CRC_EC_SPKT_BYTES | Error Correction Short Packet Bytes<br>Short packets are defined as those where the number of bytes used in the CRC calculation, including the CRC value, do not exceed this value. The max value is 255. |

## A.2.8 XCVR_PHY register descriptions

**Note**

To decipher number formats such as ‘ufix4en2’ or ‘sfix8en3’:

General form: [u/s]fix[Nb]e[n][Nx]

Where,

| | |
|---|---|
| [u/s] | denotes whether the number is signed or not |
| fix | denotes that the number is fixed point, 2's complement |
| Nb | indicates total number of bits, including a sign bit if signed |
| e | indicates that the number is exponentiated like num x 2^(exponent) |
| n | indicates the exponent is negative |
| Nx | indicates the value of the exponent |

For example :

- ufix4en2(1101) = 1101 x 2^(-2) = 11.01 = 3.25 (decimal)
- sfix8en3(10011011) = -1100101 x 2^(-3) = -1100.101 = -12.625 (decimal)

### A.2.8.1 XCVR_RX_PHY_ADDR Memory map

XCVR_PHY base address: 4103_0400h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Preamble Detect Config 0 (PHY_FSK_PD_CFG0) | 32 | RW | 0021_081Eh |
| 4h | Preamble Detect Config 1 (PHY_FSK_PD_CFG1) | 32 | RW | 0ACD_2BDEh |
| 8h | PHY Configuration (PHY_FSK_CFG) | 32 | RW | E180_B070h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| Ch | PHY Misc. Configuration (PHY_FSK_MISC) | 32 | RW | 0800_0FF0h |
| 10h | PHY BSM Network Address (NTW_ADR_BSM) | 32 | RW | 0000_0000h |
| 14h | PHY Status (FSK_STAT) | 32 | RO | See description. |
| 18h | PHY FAD control (FSK_FAD_CTRL) | 32 | RW | 0E41_8140h |

## A.2.8.2 Preamble Detect Config 0 (PHY_FSK_PD_CFG0)

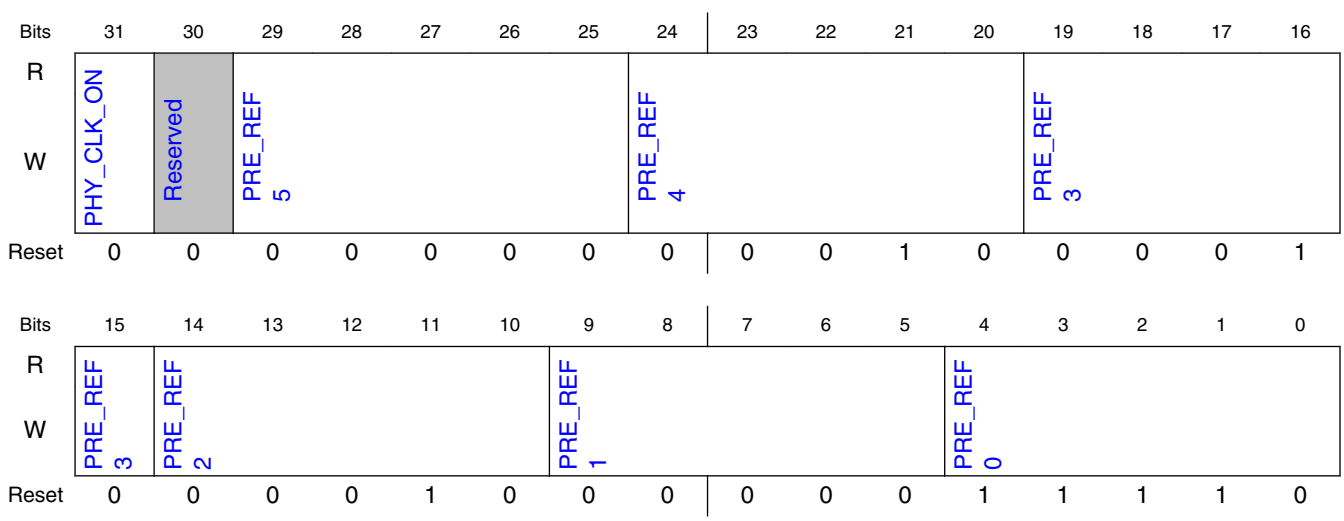
### A.2.8.2.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_PD_CFG0 | 0h |

### A.2.8.2.2 Function

Reference waveform configuration and PHY clock force.

### A.2.8.2.3 Diagram



#### A.2.8.2.4 Fields

| Field | Function |
|---|---|
| 31<br>PHY_CLK_ON | Force PHY clock ON<br>Forces the system to keep the PHY clock on. |
| 30<br>— | Reserved. |
| 29-25<br>PRE_REF5 | Reference waveform sample 5 (sfix5en5) |
| 24-20<br>PRE_REF4 | Reference waveform sample 4 (sfix5en5) |
| 19-15<br>PRE_REF3 | Reference waveform sample 3 (sfix5en5) |
| 14-10<br>PRE_REF2 | Reference waveform sample 2 (sfix5en5) |
| 9-5<br>PRE_REF1 | Reference waveform sample 1 (sfix5en5) |
| 4-0<br>PRE_REF0 | Reference waveform sample 0 (sfix5en5) |

### A.2.8.3 Preamble Detect Config 1 (PHY_FSK_PD_CFG1)

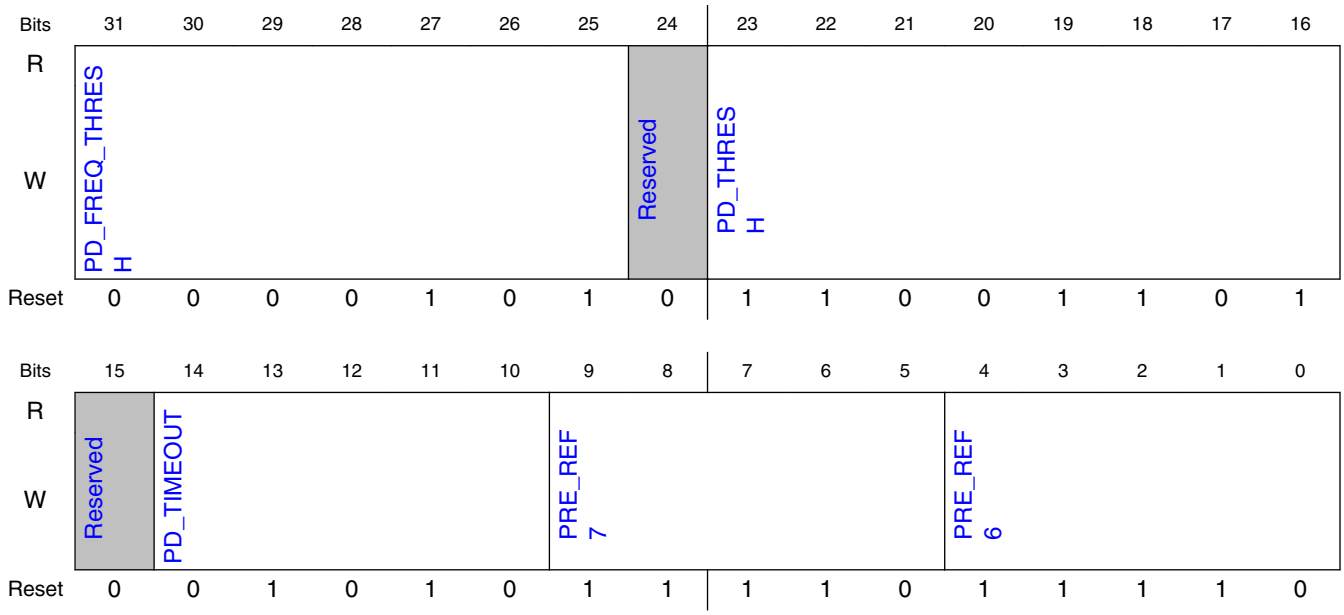
#### A.2.8.3.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_PD_CFG1 | 4h |

#### A.2.8.3.2 Function

Preamble detect configuration.

## A.2.8.3.3 Diagram




## A.2.8.3.4 Fields

| Field | Function |
|---|---|
| 31-25<br>PD_FREQ_THRESH | Preamble Detect frequency threshold<br>Frequency range within which 2 preamble detects are considered similar in frequency and, thus, correlated. Format is ufix7en9 and can be estimated by: PD_FREQ_THRESH = round(1024*(Freq_Thresh/(OSR*SYMBOL_RATE)) |
| 24<br>— | Reserved. |
| 23-16<br>PD_THRESH | Preamble detect threshold<br>Preamble detect correlation threshold (ufix8en8). This threshold is used to define high correlation for the preamble detect logic. It is the default threshold used and corresponds to PD MODE A. |
| 15<br>— | Reserved. |
| 14-10<br>PD_TIMEOUT | Preamble Detect Timeout<br>Timeout parameter for preamble detect (in samples). After timeout occurs, a better correlation is no longer searched for, and the acquired frequency estimate is committed. |
| 9-5<br>PRE_REF7 | Reference waveform sample 7 (sfix5en5) |
| 4-0<br>PRE_REF6 | Reference waveform sample 6 (sfix5en5) |

# A.2.8.4 PHY Configuration (PHY_FSK_CFG)

## A.2.8.4.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_CFG | 8h |

## A.2.8.4.2 Function

PHY configuration.

## A.2.8.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | PD_MODE_B | | PD_MODE_A | | PD_MODE_SW_EN | PD_LAT_BASE | | | | BLE_NTW_ADR_THR | | | PDA_COMP_EN | PDB_COMP_EN | DEMOD_TIMEOUT | |
| Reset | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | DEMOD_TIMEOUT | | | | PDA_CONF_EN | PDB_CONF_EN | AA_CORR_GAIN | | | | | | BSM_EN_BLE | FSK_BIT_INVERT | AA_OUT_SEL | AA_PLAYBACK |
| Reset | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

## A.2.8.4.4 Fields

| Field | Function |
|---|---|
| 31-30<br>PD_MODE_B | Preamble detect configuration for PD Mode B.<br>10b - PD mode 2, pattern based preamble detection.<br>11b - PD mode 3, peak based preamble detection. |
| 29-28<br>PD_MODE_A | Preamble detect configuration for PD Mode A.<br>10b - PD mode 2, pattern based preamble detection.<br>11b - PD mode 3, peak based preamble detection. |

*Table continues on the next page...*

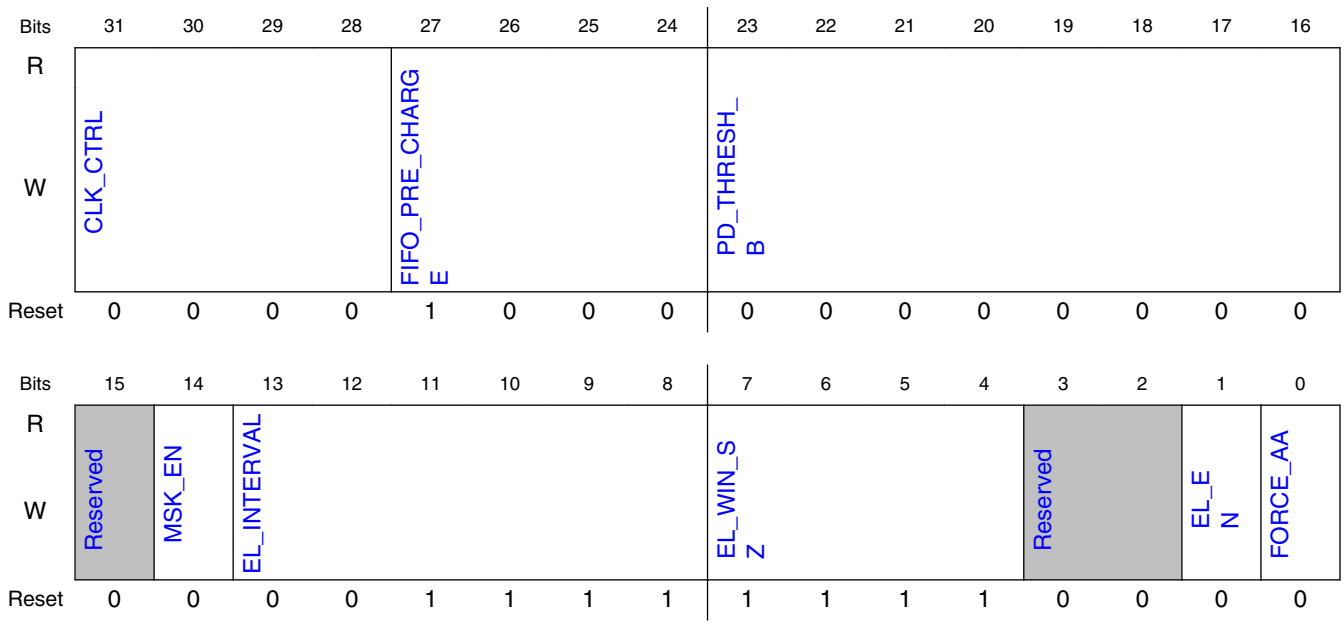
| Field | Function |
|---|---|
| 27<br>PD_MODE_SW_EN | Allow PD mode switching. When enabled, an external signal is observed to switch between PD mode A and PD mode B. If not enabled, PD mode A is always used. |
| 26-23<br>PD_LAT_BASE | User configurable adjustment to the PD latency, in samples. |
| 22-20<br>BLE_NTW_ADR_THR | Allowable hamming distance for BLE AA search. |
| 19<br>PDA_COMP_EN | Enables compressed correlation in PD Mode A. |
| 18<br>PDB_COMP_EN | Enables compressed correlation in PD Mode B. |
| 17-12<br>DEMOD_TIMEOUT | Number of symbols for which the state machine will wait past the expected end of the AA to declare a false PD and resume scanning. The actual timeout for the AA search is calculated as: TIMEOUT = (max_AA_length + DEMOD_TIMEOUT + 1.5) * OSR |
| 11<br>PDA_CONF_EN | Enables preamble AA confirmation for preamble detect mode A. |
| 10<br>PDB_CONF_EN | Enables preamble AA confirmation for preamble detect mode B. |
| 9-4<br>AA_CORR_GAIN | AA correlator gain. Format ufix6en3. This gain is applied to soft bits from the demodulator before they are used for address search synchronization. |
| 3<br>BSM_EN_BLE | Enable BSM mode for BLE. |
| 2<br>FSK_BIT_INVERT | Reserved.<br>0b - Normal demodulation.<br>1b - Invert demodulated bits. This applies at the demodulator, so it affects both AA and the data portions of the packet. |
| 1<br>AA_OUT_SEL | Reserved.<br>0b - When AA playback is enabled, play back desired AA.<br>1b - When AA playback is enabled, play back received AA. |
| 0<br>AA_PLAYBACK | Reserved.<br>0b - PHY will only output bits after the AA.<br>1b - PHY will output the AA, followed by the rest of the packet bits. |

## A.2.8.5 PHY Misc. Configuration (PHY_FSK_MISC)

## A.2.8.5.1 Offset

| Register | Offset |
|---|---|
| PHY_FSK_MISC | Ch |

## A.2.8.5.2 Diagram




## A.2.8.5.3 Fields

| Field | Function |
|---|---|
| 31-28<br>CLK_CTRL | Enables various clock gating features. Bits are individually decoded, so any combination is allowable.<br>0001b - Gate off PHY clock when phy_en is not asserted.<br>0010b - Gate off preamble detect clock when pd_enable is not asserted (internal signal).<br>0100b - Gate off AA synchronizer clock when synchronizer is not in use.<br>1000b - Gate off demodulator clock when demodulator is not in use. |
| 27-24<br>FIFO_PRE_CHARGE | Number of FIFO entries to load before sending bits out of the PHY. |
| 23-16<br>PD_THRESH_B | Preamble detect correlation threshold (ufix8en8). This threshold is used to define high correlation for the preamble detect logic. It is the threshold used for PD MODE B. |
| 15<br>— | Reserved. |
| 14<br>MSK_EN | Configures the PHY for MSK decoding. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 13-8<br>EL_INTERVAL | Number of symbols between successive EL operation windows. |
| 7-4<br>EL_WIN_SZ | Number of successive symbols over which one EL operation occurs. |
| 3-2<br>— | Reserved. |
| 1<br>EL_EN | Enable Early-Late bit timing logic. |
| 0<br>FORCE_AA | Forces PHY to assume an AA match has occurred. For debug use only. |

## A.2.8.6 PHY BSM Network Address (NTW_ADR_BSM)

### A.2.8.6.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_BSM | 10h |

### A.2.8.6.2 Function

Address used when BLE BSM mode is enabled.

### A.2.8.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_BSM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_BSM | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.8.6.4 Fields

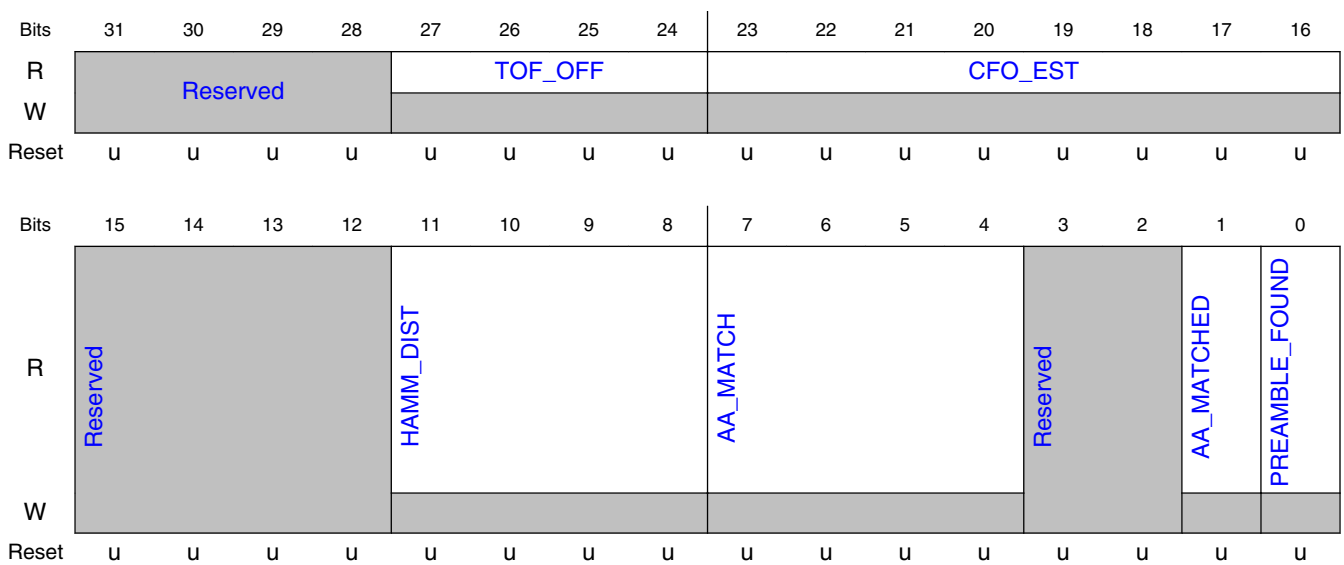
| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_BSM | Address used when BLE BSM mode is enabled. |

## A.2.8.7 PHY Status (FSK_STAT)

### A.2.8.7.1 Offset

| Register | Offset |
|---|---|
| FSK_STAT | 14h |

### A.2.8.7.2 Diagram




### A.2.8.7.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-24 | Timing offset for use in time-of-flight calculation. |

*Table continues on the next page...*

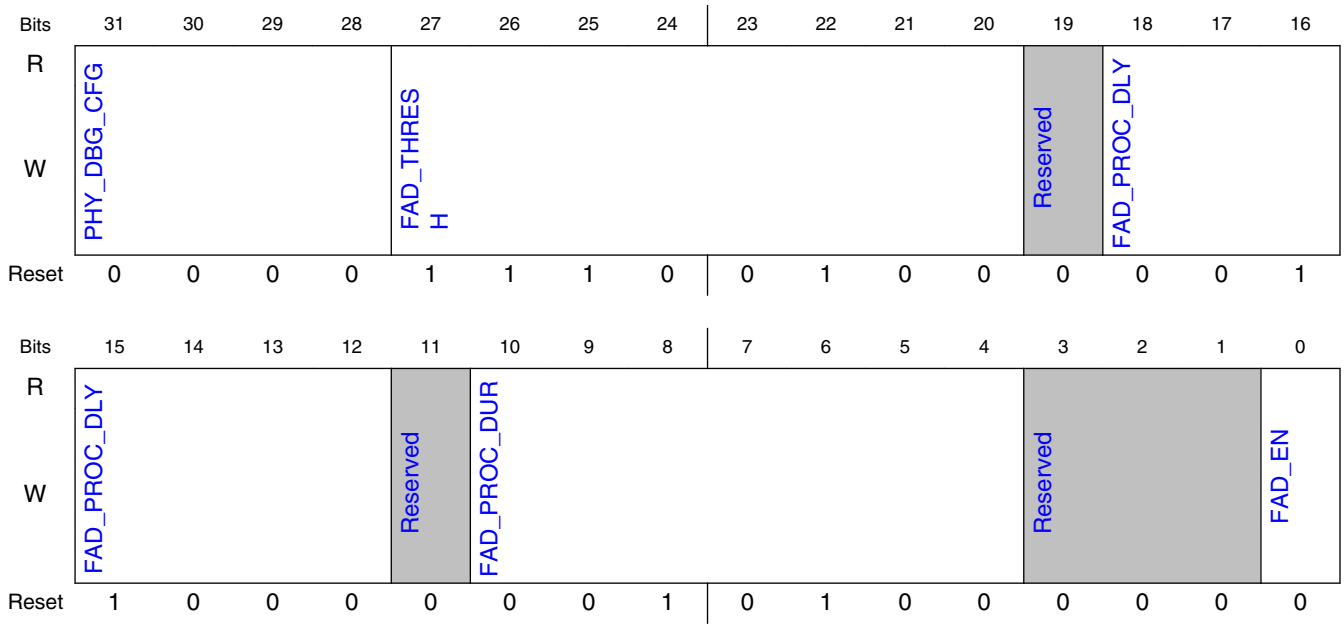
| Field | Function |
|---|---|
| TOF_OFF | |
| 23-16<br>CFO_EST | Indicates the currently estimated CFO. Format is ufix7en9 and can be estimated by: CFO = (CFO_EST/1024)*(OSR*BIT_RATE) |
| 15-12<br>— | Reserved. |
| 11-8<br>HAMM_DIST | Indicates the hamming distance witnessed when AA match occurred. |
| 7-4<br>AA_MATCH | Indicates which AA has matched. This will clear when the PHY is re-initialized. |
| 3-2<br>— | Reserved. |
| 1<br>AA_MATCHED | Indicates that an AA match has occurred. This will only clear when the PHY is re-intialized. |
| 0<br>PREAMBLE_FOUND | Indicates that a preamble detect is active. |

## A.2.8.8 PHY FAD control (FSK_FAD_CTRL)

### A.2.8.8.1 Offset

| Register | Offset |
|---|---|
| FSK_FAD_CTRL | 18h |

## A.2.8.8.2 Diagram




## A.2.8.8.3 Fields

| Field | Function |
|---|---|
| 31-28<br>PHY_DBG_CFG | Individual bits for debug features. |
| 27-20<br>FAD_THRESH | Threshold used to compare correlator outputs when scanning. |
| 19<br>— | Reserved. |
| 18-12<br>FAD_PROC_DLY | Time for FAD to stay in WAIT state (see FAD description). |
| 11<br>— | Reserved. |
| 10-4<br>FAD_PROC_DUR | Cumulative time for WAIT and PROC2 states in FAD engine (see FAD description). |
| 3-1<br>— | Reserved. |
| 0<br>FAD_EN | Enable Fast-Antenna-Diversity |

## A.2.9 XCVR_ZBDEMOD register descriptions

### A.2.9.1 XCVR_ZBDEMOD_ADDR Memory map

XCVR_ZBDEM base address: 4103_0480h

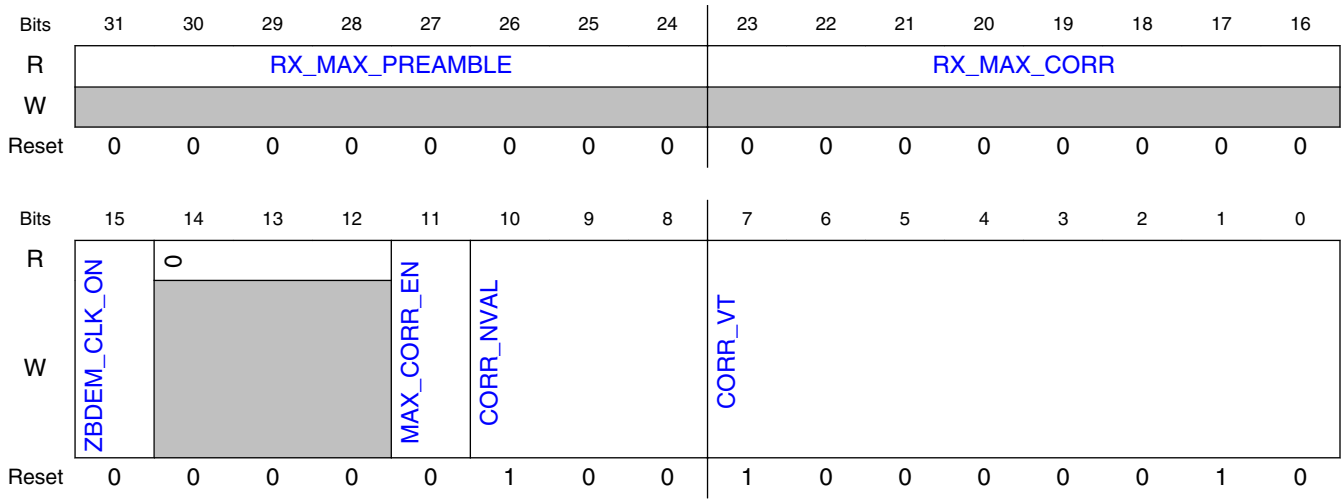
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | 802.15.4 DEMOD CORRELATOR CONTROL (CORR_CTRL) | 32 | RW | 0000_0482h |
| 4h | 802.15.4 DEMOD PN TYPE (PN_TYPE) | 32 | RW | 0000_0001h |
| 8h | 802.15.4 DEMOD PN CODE (PN_CODE) | 32 | RW | 744A_C39Bh |
| Ch | 802.15.4 DEMOD SYMBOL SYNC CONTROL (SYNC_CTRL) | 32 | RW | 0000_0008h |
| 10h | 802.15.4 CCA/LQI SOURCE (CCA_LQI_SRC) | 32 | RW | 0000_000Ch |
| 14h | FAD CORRELATOR THRESHOLD (FAD_LPPS_THR) | 32 | RW | 6060_6082h |
| 18h | 802.15.4 AFC STATUS (ZBDEM_AFC) | 32 | RW | 0000_0001h |
| 1Ch | CCA MODE 2 CONTROL REGISTER (CCA2_CTRL) | 32 | RW | 0000_FF05h |
| 20h | CCA MODE 2 CONTROL REGISTER (CCA2_THRESH) | 32 | RW | 0040_0080h |
| 24h | CCA MODE 2 STATUS REGISTER (CCA2_STATUS) | 32 | RO | 0000_0000h |

### A.2.9.2 802.15.4 DEMOD CORRELATOR CONTROL (CORR_CTRL)

#### A.2.9.2.1 Offset

| Register | Offset |
|---|---|
| CORR_CTRL | 0h |

### A.2.9.2.2 Diagram




### A.2.9.2.3 Fields

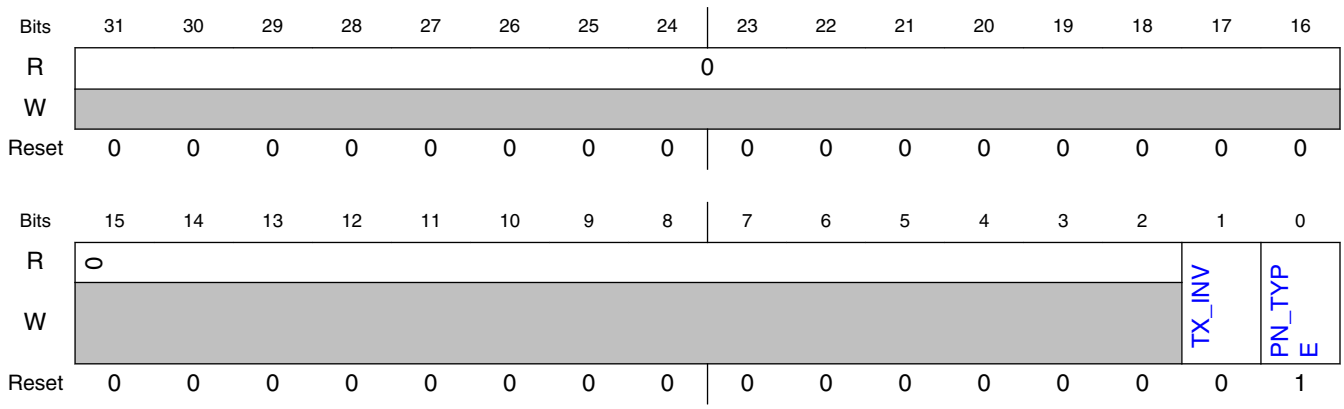
| Field | Function |
|---|---|
| 31-24<br>RX_MAX_PREAMBLE | RX_MAX_PREAMBLE<br>Max correlator during preamble-- max correlator value found during the preamble. |
| 23-16<br>RX_MAX_CORR | RX_MAX_CORR<br>Max correlator after preamble-- max correlator value found in packet after the preamble (refreshed every symbol rate if MAX_CORR_EN=1). |
| 15<br>ZBDEM_CLK_ON | Force 802.15.4 Demodulator Clock On<br>0b - Normal Operation<br>1b - Force 802.15.4 Demodulator Clock On (debug purposes only) |
| 14-12<br>— | Reserved |
| 11<br>MAX_CORR_EN | MAX_CORR_EN<br>Max correlator after preamble enable-- Enable the refresh of the max corr register |
| 10-8<br>CORR_NVAL | CORR_NVAL<br>Number of consecutively detected zero-symbols required to declare a preamble detected |
| 7-0<br>CORR_VT | CORR_VT<br>Correlator threshold, defines the sensitivity of demod during the preamble search state |

## A.2.9.3 802.15.4 DEMOD PN TYPE (PN_TYPE)

### A.2.9.3.1 Offset

| Register | Offset |
|---|---|
| PN_TYPE | 4h |

### A.2.9.3.2 Diagram




### A.2.9.3.3 Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>TX_INV | TX_INV<br>test mode to invert the transmission |
| 0<br>PN_TYPE | PN_TYPE<br>PN Type - Pseudo Noise Chip Code Type (for 802.15.4 = 1) |

## A.2.9.4 802.15.4 DEMOD PN CODE (PN_CODE)

### A.2.9.4.1 Offset

| Register | Offset |
|---|---|
| PN_CODE | 8h |

### A.2.9.4.2 Function

Pseudo Noise Chip Code Seed Value

### A.2.9.4.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PN_MSB | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PN_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

### A.2.9.4.4 Fields

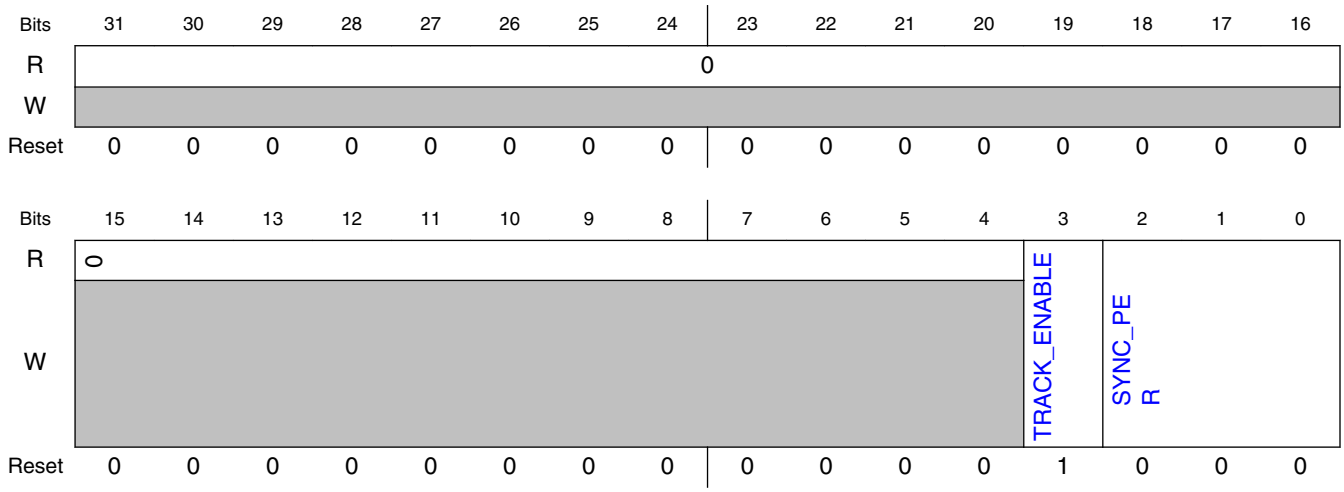
| Field | Function |
|---|---|
| 31-16<br>PN_MSB | PN_MSB<br>PN_CODE MS half |
| 15-0<br>PN_LSB | PN_LSB<br>PN_CODE LS half |

## A.2.9.5 802.15.4 DEMOD SYMBOL SYNC CONTROL (SYNC_CTRL)

### A.2.9.5.1 Offset

| Register | Offset |
|---|---|
| SYNC_CTRL | Ch |

### A.2.9.5.2 Diagram




### A.2.9.5.3 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>TRACK_ENABLE | TRACK_ENABLE<br>0b - symbol timing synchronization tracking disabled in Rx frontend<br>1b - symbol timing synchronization tracking enabled in Rx frontend (default) |
| 2-0<br>SYNC_PER | Symbol Sync Tracking Period<br>determines update rate for symbol timing, per equation. An early/late measurement is made every 2^SYNC_PER[2:0] symbols. Valid range of SYNC_PER[2:0] is 0 to 4. |

## A.2.9.6 802.15.4 CCA/LQI SOURCE (CCA_LQI_SRC)

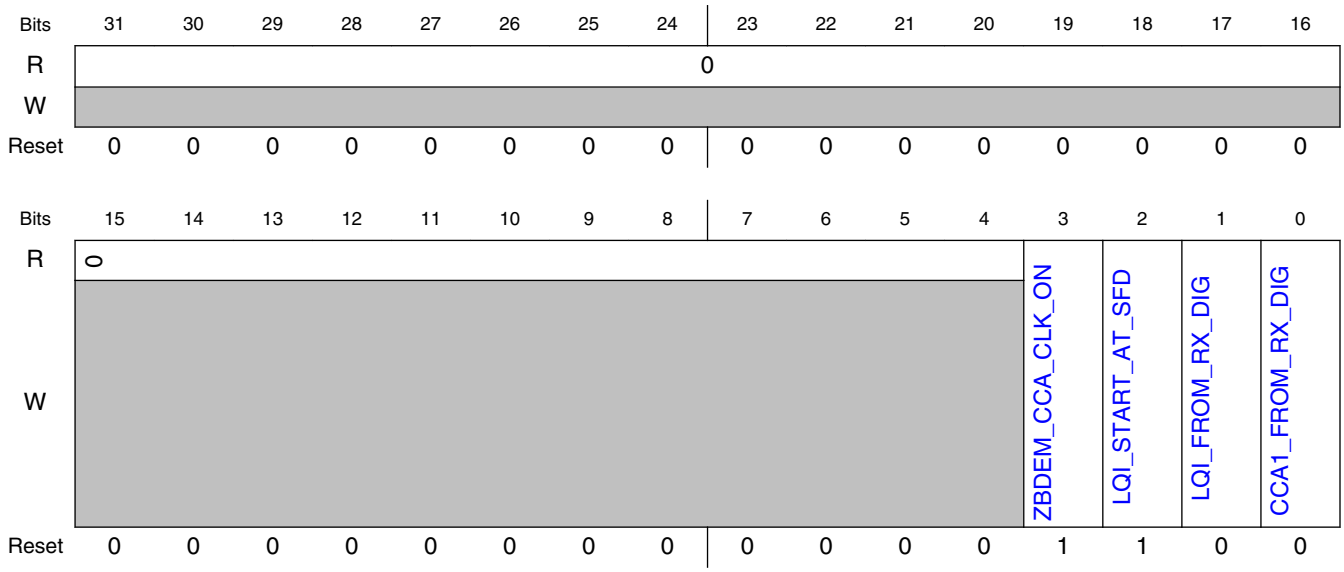
### A.2.9.6.1 Offset

| Register | Offset |
|---|---|
| CCA_LQI_SRC | 10h |

### A.2.9.6.2 Function

Selects the Source of CCA and LQI Information Provided to the 802.15.4 Link Layer

### A.2.9.6.3 Diagram




### A.2.9.6.4 Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>ZBDEM_CCA_CLK_ON | 802.15.4 Demodulator CCA Clock Enable<br>802.15.4 Demodulator CCA Clock Enable. This bit should be left in its default state (1) if any CCA Mode (Modes 1, 2, and 3), Energy Detect, or LQI uses the 802.15.4 Demodulator to compute its value. If 802.15.4 Demodulator is not used to compute any of these, this bit can be cleared to save power. |
| 2<br>LQI_START_AT_SFD | Select Start Point for LQI Computation<br>0b - Start LQI computation at Preamble Detection (similar to previous NXP 802.15.4 products)<br>1b - Start LQI computation at SFD (Start of Frame Delimiter) Detection |
| 1<br>LQI_FROM_RX_DIG | Selects the Source of LQI (Link Quality Indicator) Information Provided to the 802.15.4 Link Layer<br>0b - Use the LQI information computed internally in the 802.15.4 Demod<br>1b - Use the LQI information computed by the RX Digital |
| 0<br>CCA1_FROM_RX_DIG | Selects the Source of CCA1 (Clear Channel Assessment Mode 1) Information Provided to the 802.15.4 Link Layer<br>0b - Use the CCA1 information computed internally in the 802.15.4 Demod<br>1b - Use the CCA1 information computed by the RX Digital |

## A.2.9.7 FAD CORRELATOR THRESHOLD (FAD_LPPS_THR)

## A.2.9.7.1 Offset

| Register | Offset |
|---|---|
| FAD_LPPS_THR | 14h |

## A.2.9.7.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | LPPS_LP_EN_COUNT | | | | | | | 0 | LPPS_FILL_COUNT | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FAD_FILL1 | | | | | | | FAD_THR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

## A.2.9.7.3 Fields

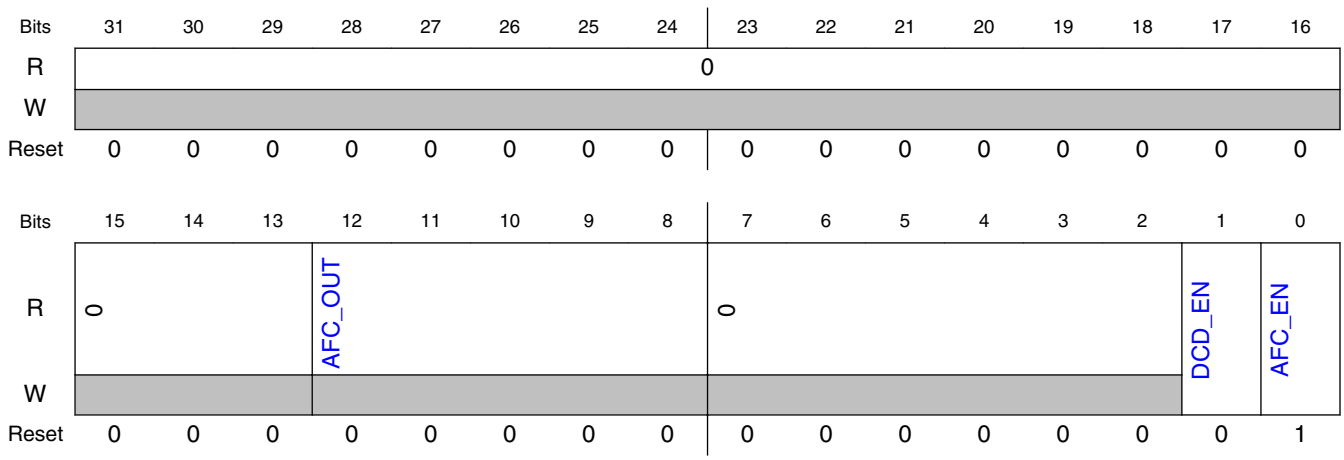
| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-24<br>LPPS_LP_EN_COUNT | LPPS_LP_EN high time<br>Parameter used to configure the duration of time lpps_lp_enable is in high state |
| 23<br>— | Reserved |
| 22-16<br>LPPS_FILL_COUNT | Wait duration after lpps_lp_enable is de-asserted<br>Parameter used to configure the wait duration used for valid samples acquisition after *lpps_lp_enable* is de-asserted. The overall wait duration expressed in base-band samples is determined using formula:<br>*lpps_fill_duration=lpps_fill_count+lpps_tsm_warmup_duration*<br>where *lpps_tsm_warmup_duration* is the duration of the fast TSM warmup, used for the LPPS mode, expressed in samples. |
| 15<br>— | Reserved |
| 14-8<br>FAD_FILL1 | Pre-detection buffer filling duration<br>Interval used to wait for valid samples acquisition after an antenna switch command has been issued. |
| 7-0<br>FAD_THR | FAD_THR<br>Correlator threshold at which the FAD will select the antenna. |

# A.2.9.8 802.15.4 AFC STATUS (ZBDEM_AFC)

## A.2.9.8.1 Offset

| Register | Offset |
|---|---|
| ZBDEM_AFC | 18h |

## A.2.9.8.2 Diagram




## A.2.9.8.3 Fields

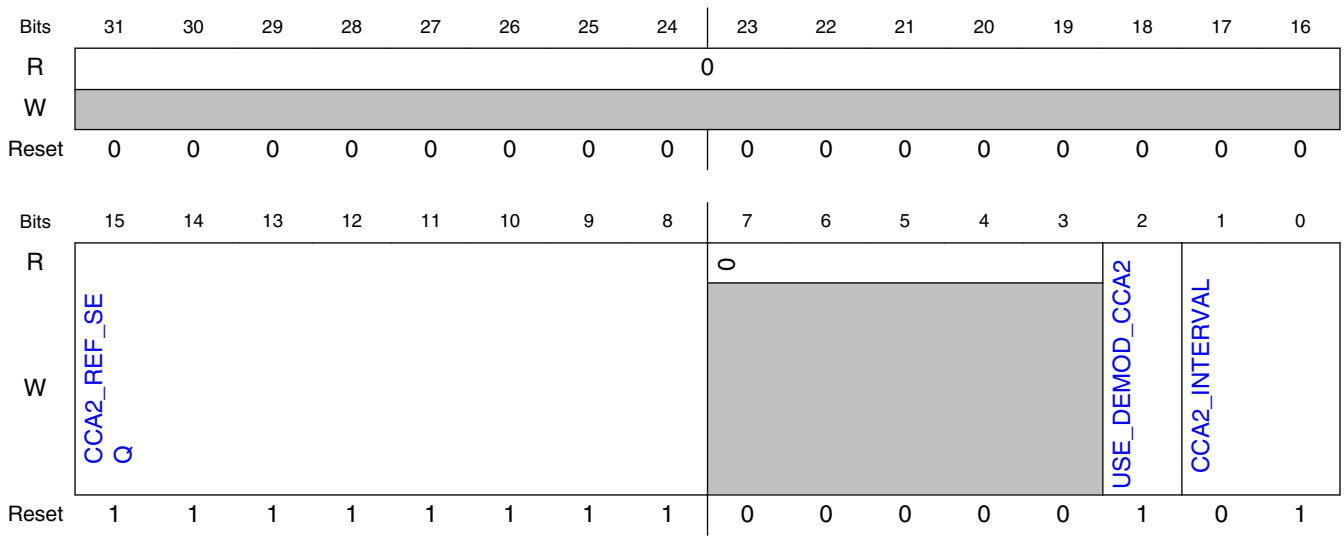
| Field | Function |
|---|---|
| 31-13<br>— | Reserved |
| 12-8<br>AFC_OUT | AFC_OUT<br>AFC Result from the last received packet. Format is Signed, Two's Complement. Each LSB represents 15KHz of Frequency Offset |
| 7-2<br>— | Reserved |
| 1<br>DCD_EN | DCD_EN<br>0b - NCD Mode (default)<br>1b - DCD Mode |
| 0<br>AFC_EN | AFC_EN<br>the AFC Function<br>0b - AFC is disabled<br>1b - AFC is enabled |

# A.2.9.9 CCA MODE 2 CONTROL REGISTER (CCA2_CTRL)

## A.2.9.9.1 Offset

| Register | Offset |
|---|---|
| CCA2_CTRL | 1Ch |

## A.2.9.9.2 Diagram




## A.2.9.9.3 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CCA2_REF_SEQ | CCA Mode 2 Sequence Address<br>Selection of the sequence addresses to be used. |
| 7-3<br>— | Reserved |
| 2<br>USE_DEMOD_CCA2 | Selects CCA Mode 2 Computation Engine<br>Selects between demodulator-based or standalone CCA Mode 2 Computation Engine<br>0b - Use standalone (new) CCA Mode 2 Engine, decoupled from demodulator<br>1b - Use 802.15.4 demodulator-based (legacy) CCA Mode 2 Engine (default) |
| 1-0 | CCA Mode 2 Measurement Window Duration |

| Field | Function |
|---|---|
| CCA2_INTERVAL | Specifies the CCA2 measurement window duration:<br>00b - 64 µs<br>01b - 128 µs<br>10b - 256 µs<br>11b - 512 µs |

## A.2.9.10 CCA MODE 2 CONTROL REGISTER (CCA2_THRESH)

### A.2.9.10.1 Offset

| Register | Offset |
|---|---|
| CCA2_THRESH | 20h |

### A.2.9.10.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CCA2_SYM_THRESH | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CCA2_CNT_THRESH | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.2.9.10.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>CCA2_SYM_THRESH | CCA Mode 2 Symbol Threshold<br>Threshold used to detect repetition sequence addresses corresponding to correlation products that exceed CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |
| 15-10<br>— | Reserved |

*Table continues on the next page...*

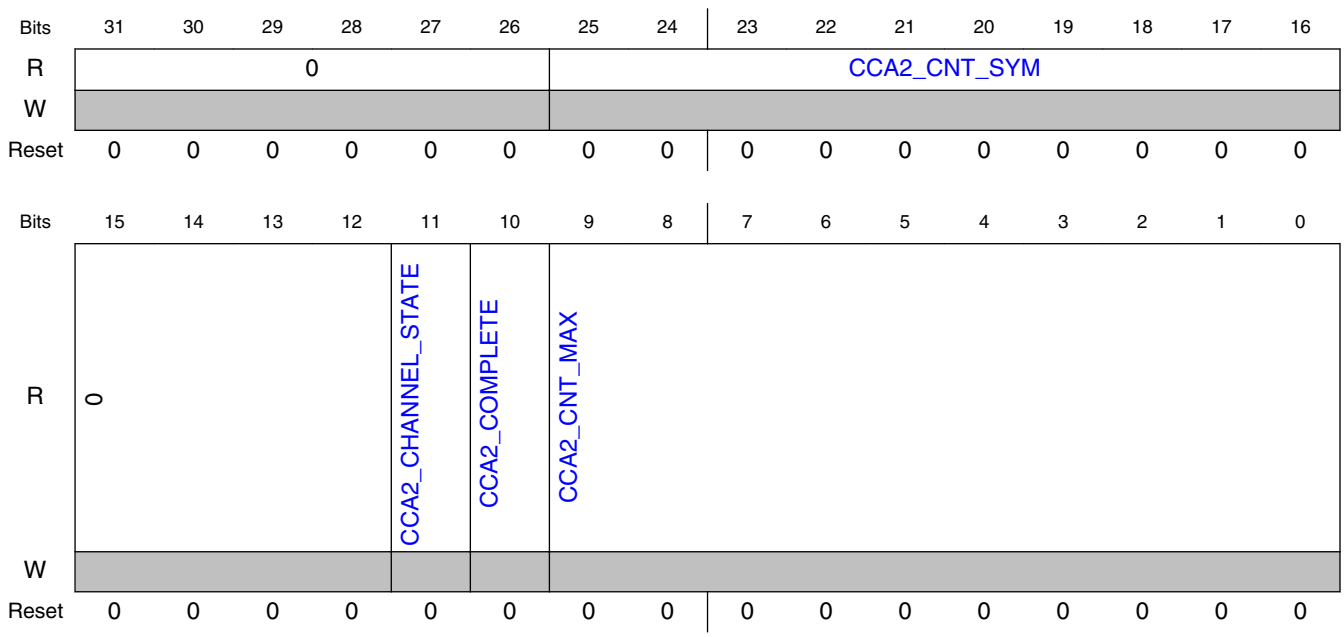
| Field | Function |
|---|---|
| 9-0<br>CCA2_CNT_THRESH | CCA Mode 2 Count Threshold<br>Threshold to compare count of correlation products exceeding CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |

## A.2.9.11 CCA MODE 2 STATUS REGISTER (CCA2_STATUS)

### A.2.9.11.1 Offset

| Register | Offset |
|---|---|
| CCA2_STATUS | 24h |

### A.2.9.11.2 Diagram




### A.2.9.11.3 Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 25-16<br>CCA2_CNT_SYM | CCA Mode 2 Repetition Sequence Addresses Count<br>Actual count of repetition sequence addresses corresponding to correlation products that exceed CCA2_CORR_THRESH[7:0].<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |
| 15-12<br>— | Reserved |
| 11<br>CCA2_CHANNEL_STATE | CCA Mode 2 Channel State<br>Indicates whether the channel is idle or busy; 0=idle, 1=busy. Asserted in any of the following cases:<br>1. CCA mode 2 measurement finds the channel busy when CCA2 duration expires<br>2. *rx_preamble_det* signal is asserted (note that, following this event, if *rx_preamble_det* is deasserted and *zb_cca2_complete* is low, *zb_cca2_channel_state* is deasserted also)<br>3. *rx_sfd_detect* signal is asserted |
| 10<br>CCA2_COMPLETE | CCA Mode 2 Measurement Complete<br>Indicates when a CCA mode 2 measurement is complete. Asserted in any of the following cases:<br>1. CCA mode 2 measurement is completed (when CCA2 duration has expired)<br>2. rx_sfd_detect signal is asserted |
| 9-0<br>CCA2_CNT_MAX | CCA Mode 2 Maximum Count<br>Actual count of correlation products exceeding CCA2_CORR_THRESH[7:0] during the measurement interval.<br>**NOTE:** CCA2_CORR_THRESH[7:0] resides in 802.15.4 address space |

## A.2.10 XCVR_PKT_RAM register descriptions

### A.2.10.1 XCVR_PKT_RAM_ADDR Memory map

XCVR_PKT_RAM base address: 4103_0700h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 8FEh | Shared Packet RAM for multiple Link Layer usage. (PACKET_RAM_0 - PACKET_RAM_1151) | 16 | RW | See description. |

### A.2.10.2 Shared Packet RAM for multiple Link Layer usage. (PACKET_RAM_0 - PACKET_RAM_1151)

### A.2.10.2.1 Offset

For a = 0 to 1151:

| Register | Offset |
|---|---|
| PACKET_RAM_a | 0h + (a × 2h) |

### A.2.10.2.2 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MSBYTE | | | | | | | | LSBYTE | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### A.2.10.2.3 Fields

| Field | Function |
|---|---|
| 15-8<br>MSBYTE | MSBYTE |
| 7-0<br>LSBYTE | LSBYTE |

# A.3 Link Layer Memory Map and Register Definition

## A.3.1 BTLE_RF register descriptions

### A.3.1.1 BTLE_RF Memory map

BTLE_RF base address: 4103_2000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 600h | BLUETOOTH LOW ENERGY PART ID (BLE_PART_ID) | 16 | RO | 0004h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 604h | BLE DSM STATUS (DSM_STATUS) | 16 | RO | See description. |
| 608h | BLE MISCELLANEOUS CONTROL (MISC_CTRL) | 16 | RW | 0000h |
| 60Ch | BLE STATE MACHINE STATUS (BLE_FSM) | 16 | RO | 0000h |

## A.3.1.2 BLUETOOTH LOW ENERGY PART ID (BLE_PART_ID)

### A.3.1.2.1 Offset

| Register | Offset |
|---|---|
| BLE_PART_ID | 600h |

### A.3.1.2.2 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLE_PART_ID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### A.3.1.2.3 Fields

| Field | Function |
|---|---|
| 15-0<br>BLE_PART_ID | BLE Part ID<br>0000000000000000b - Pre-production<br>0000000000000001b - Pre-production<br>0000000000000010b - KW40<br>0000000000000011b - KW41<br>0000000000000100b - K3S<br>0000000000000101b - KW35/KW36 |

## A.3.1.3 BLE DSM STATUS (DSM_STATUS)

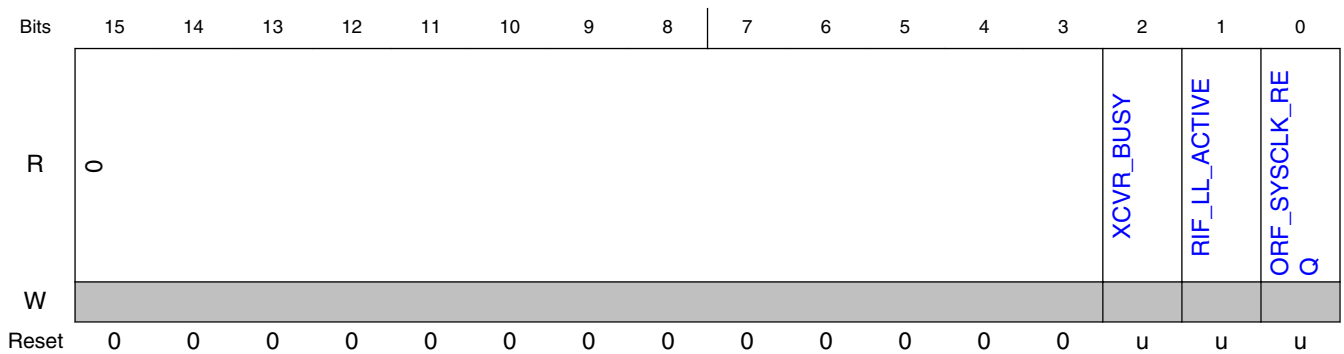
### A.3.1.3.1 Offset

| Register | Offset |
|---|---|
| DSM_STATUS | 604h |

### A.3.1.3.2 Function

BLE Deep Sleep Mode Status Register

### A.3.1.3.3 Diagram




### A.3.1.3.4 Fields

| Field | Function |
|---|---|
| 15-3<br>— | Reserved. |
| 2<br>XCVR_BUSY | Transceiver Busy Status Bit<br>0b - RF Channel in available (TSM is idle)<br>1b - RF Channel in use (TSM is busy) |
| 1<br>RIF_LL_ACTIVE | Link Layer Active<br>Reflects the state of the BLE LL output of the same name, the signal to be used by the host as an 'early' indication to prevent host to do any operations while the BLE block is doing transceiver operations, so as to reduce the peak power and noise. |
| 0<br>ORF_SYSCLK_REQ | RF Oscillator Requested<br>Reflects the state of the BLE LL output of the same name, the control signal used to enable/disable the RF Oscillator for entry and exit from DSM (deep sleep mode). |

## A.3.1.4 BLE MISCELLANEOUS CONTROL (MISC_CTRL)

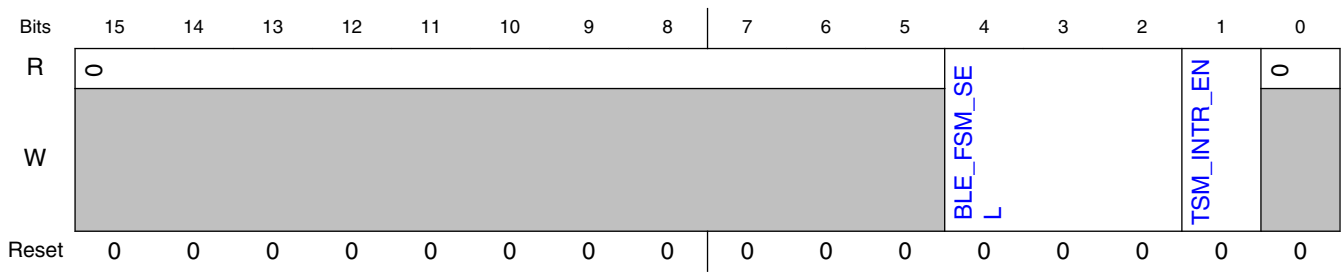
## A.3.1.4.1 Offset

| Register | Offset |
|---|---|
| MISC_CTRL | 608h |

## A.3.1.4.2 Function

BLE Miscellaneous Control Register

## A.3.1.4.3 Diagram




## A.3.1.4.4 Fields

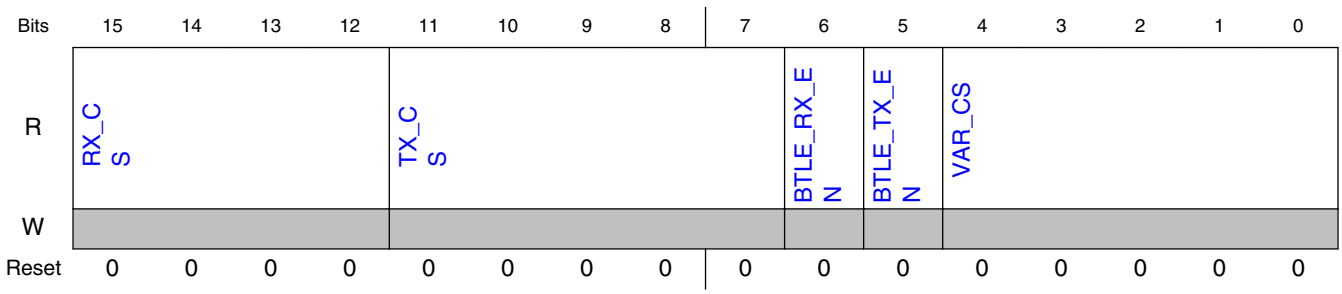
| Field | Function |
|---|---|
| 15-5<br>— | Reserved. |
| 4-2<br>BLE_FSM_SEL | BLE FSM Selector<br>BLE_FSM[2:0] selects which BLE state machine appears in the VAR_CS field of the BLE_FSM register, according to the following table:<br>0: VAR_CS = {2'b00, dtm_cs[2:0]} (DTM Current State)<br>1: VAR_CS = { adv_cs[4:0]} (ADV Current State)<br>2: VAR_CS = { scan_cs[4:0]} (SCAN Curernt State)<br>3: VAR_CS = { init_cs[4:0]} (INIT Current State)<br>4: VAR_CS = {1'b0, connection_cs[3:0]} (CONN Current State, using Connection Engine 0)<br>5: Reserved<br>6: Reserved<br>7: Reserved |
| 1<br>TSM_INTR_EN | TSM Interrupt Enable<br>This control bit enables the TSM Interrupt for BLE. If TSM_INTR_EN=1 and a TSM Interrupt occurs during a BLE TX or RX operation, bit [9] of the BLE EVENT_STATUS register will become set. |
| 0<br>— | Reserved. |

# A.3.1.5 BLE STATE MACHINE STATUS (BLE_FSM)

## A.3.1.5.1 Offset

| Register | Offset |
|---|---|
| BLE_FSM | 60Ch |

## A.3.1.5.2 Diagram




## A.3.1.5.3 Fields

| Field | Function |
|---|---|
| 15-12<br>RX_CS | BLE RX State Machine Current State<br>Current State of *rx_cs[3:0]* (BLE RX state machine) |
| 11-7<br>TX_CS | BLE TX State Machine Current State<br>Current State of *tx_cs[4:0]* (BLE TX state machine) |
| 6<br>BTLE_RX_EN | BLE RX Enable Control to TSM<br>Current State of *btle_rx_en* (BLE TSM RX request input) |
| 5<br>BTLE_TX_EN | BLE TX Enable Control to TSM<br>Current State of *btle_tx_en* (BLE TSM TX request input) |
| 4-0<br>VAR_CS | Variable State Machine Current State<br>One of several BLE state machines can be mapped to VAR_CS[4:0]. See description of BLE_FSM_SEL[2:0] in the MISC_CTRL register |

# A.3.2 ZLL register descriptions

### A.3.2.1 ZLL Memory map

ZLL base address: 4103_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | INTERRUPT REQUEST STATUS (IRQSTS) | 32 | RW | See description. |
| 4h | PHY CONTROL (PHY_CTRL) | 32 | RW | 0805_FF00h |
| 8h | EVENT TIMER (EVENT_TMR) | 32 | RW | See description. |
| Ch | TIMESTAMP (TIMESTAMP) | 32 | RO | See description. |
| 10h | T1 COMPARE (T1CMP) | 32 | RW | 00FF_FFFFh |
| 14h | T2 COMPARE (T2CMP) | 32 | RW | 00FF_FFFFh |
| 18h | T2 PRIME COMPARE (T2PRIMECMP) | 32 | RW | 0000_FFFFh |
| 1Ch | T3 COMPARE (T3CMP) | 32 | RW | 00FF_FFFFh |
| 20h | T4 COMPARE (T4CMP) | 32 | RW | 00FF_FFFFh |
| 24h | PA POWER (PA_PWR) | 32 | RW | 0000_0000h |
| 28h | CHANNEL NUMBER 0 (CHANNEL_NUM0) | 32 | RW | 0000_0012h |
| 2Ch | LQI AND RSSI (LQI_AND_RSSI) | 32 | RO | See description. |
| 30h | MAC SHORT ADDRESS 0 (MACSHORTADDRS0) | 32 | RW | FFFF_FFFFh |
| 34h | MAC LONG ADDRESS 0 LSB (MACLONGADDRS0_LSB) | 32 | RW | FFFF_FFFFh |
| 38h | MAC LONG ADDRESS 0 MSB (MACLONGADDRS0_MSB) | 32 | RW | FFFF_FFFFh |
| 3Ch | RECEIVE FRAME FILTER (RX_FRAME_FILTER) | 32 | RW | See description. |
| 40h | CCA AND LQI CONTROL (CCA_LQI_CTRL) | 32 | RW | 0866_004Bh |
| 44h | CCA2 CONTROL (CCA2_CTRL) | 32 | RW | See description. |
| 4Ch | DSM CONTROL (DSM_CTRL) | 32 | RW | 0000_0000h |
| 50h | BSM CONTROL (BSM_CTRL) | 32 | RW | 0000_0000h |
| 54h | MAC SHORT ADDRESS FOR PAN1 (MACSHORTADDRS1) | 32 | RW | FFFF_FFFFh |
| 58h | MAC LONG ADDRESS 1 LSB (MACLONGADDRS1_LSB) | 32 | RW | FFFF_FFFFh |
| 5Ch | MAC LONG ADDRESS 1 MSB (MACLONGADDRS1_MSB) | 32 | RW | FFFF_FFFFh |
| 60h | DUAL PAN CONTROL (DUAL_PAN_CTRL) | 32 | RW | See description. |
| 64h | CHANNEL NUMBER 1 (CHANNEL_NUM1) | 32 | RW | 0000_007Fh |
| 68h | SAM CONTROL (SAM_CTRL) | 32 | RW | 8080_4000h |
| 6Ch | SOURCE ADDRESS MANAGEMENT TABLE (SAM_TABLE) | 32 | RW | See description. |
| 70h | SOURCE ADDRESS MANAGEMENT MATCH (SAM_MATCH) | 32 | RO | See description. |
| 74h | SAM FREE INDEX (SAM_FREE_IDX) | 32 | RO | See description. |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 78h | SEQUENCE CONTROL AND STATUS (SEQ_CTRL_STS) | 32 | RW | See description. |
| 7Ch | ACK DELAY (ACKDELAY) | 32 | RW | 0000_0007h |
| 80h | FILTER FAIL CODE (FILTERFAIL_CODE) | 32 | RW | See description. |
| 84h | RECEIVE WATER MARK (RX_WTR_MARK) | 32 | RW | 0000_00FFh |
| 8Ch | SLOT PRELOAD (SLOT_PRELOAD) | 32 | RW | 0000_0074h |
| 90h | 802.15.4 SEQUENCE STATE (SEQ_STATE) | 32 | RO | See description. |
| 94h | TIMER PRESCALER (TMR_PRESCALE) | 32 | RW | 0000_0005h |
| 98h | LENIENCY LSB (LENIENCY_LSB) | 32 | RW | 0000_0000h |
| 9Ch | LENIENCY MSB (LENIENCY_MSB) | 32 | RW | 0000_0000h |
| A0h | PART ID (PART_ID) | 32 | RO | 0000_0003h |
| 100h - 17Eh | Packet Buffer TX (PKT_BUFFER_TX0 - PKT_BUFFER_TX63) | 16 | RW | See description. |
| 180h - 1FEh | Packet Buffer RX (PKT_BUFFER_RX0 - PKT_BUFFER_RX63) | 16 | RW | See description. |

## A.3.2.2 INTERRUPT REQUEST STATUS (IRQSTS)

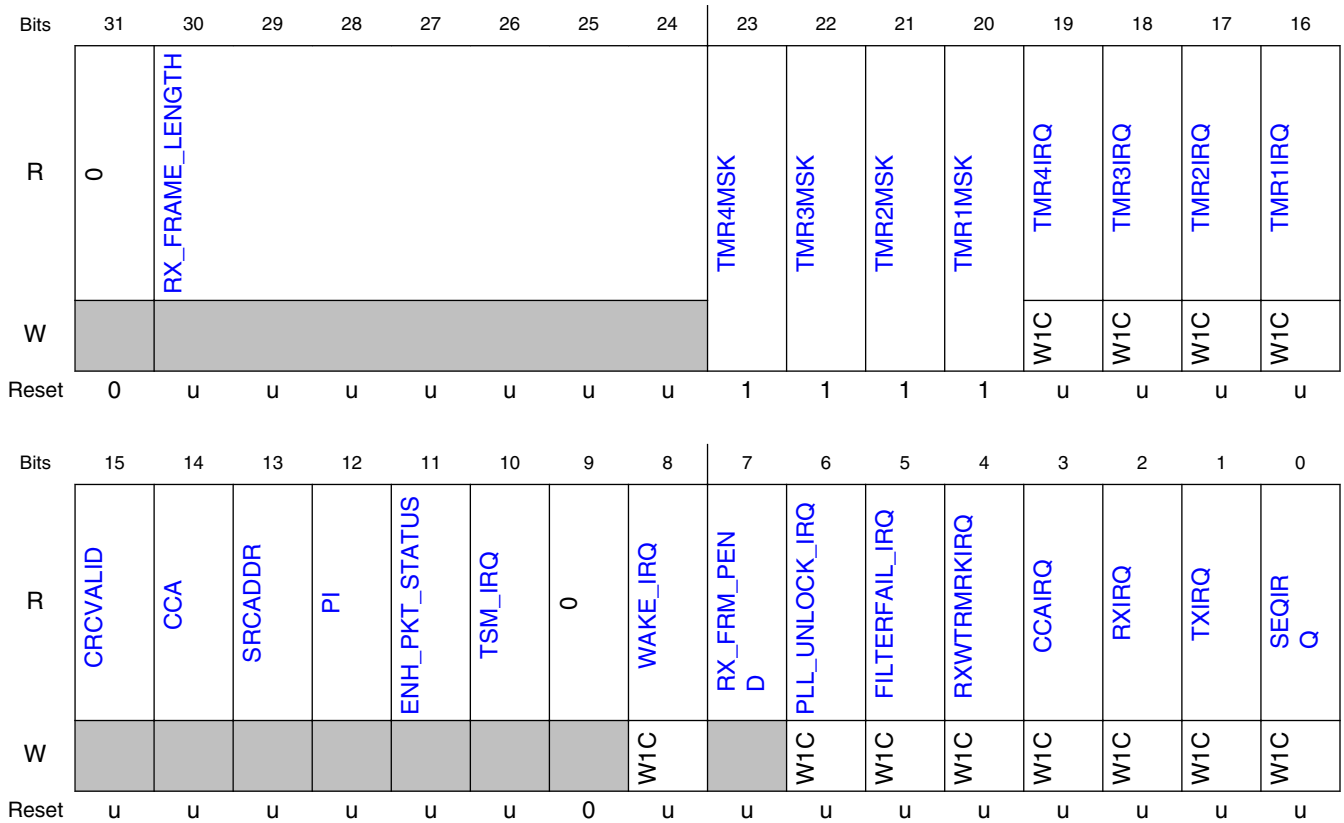
### A.3.2.2.1 Offset

| Register | Offset |
|---|---|
| IRQSTS | 0h |

### A.3.2.2.2 Function

The IRQSTS register indicates the status of interrupt requests in the 802.15.4 module.

## A.3.2.2.3 Diagram




## A.3.2.2.4 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-24<br>RX_FRAME_LENGTH | Receive Frame Length<br>Contents of the PHR (PHY header), or FrameLength field, of the most recently received packet. Read-only. |
| 23<br>TMR4MSK | Timer Comperator 4 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR4IRQ flag can be set |
| 22<br>TMR3MSK | Timer Comperator 3 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR3IRQ flag can be set |
| 21<br>TMR2MSK | Timer Comperator 2 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR2IRQ flag can be set |
| 20<br>TMR1MSK | Timer Comperator 1 Interrupt Mask bit<br>0b - allows interrupt when comparator matches event timer count<br>1b - Interrupt generation is disabled, but a TMR1IRQ flag can be set |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 19<br>TMR4IRQ | Timer 4 IRQ<br>Timer Comparator 4 Interrupt Status bit: Indiates T4CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 18<br>TMR3IRQ | Timer 3 IRQ<br>Timer Comparator 3 Interrupt Status bit: Indiates T3CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 17<br>TMR2IRQ | Timer 2 IRQ<br>Timer Comparator 2 Interrupt Status bit: Indiates comparator value matched event timer counter. This flag is shared between the T2CMP (24-bit) and T2PRIMECMP (16-bit) compare registers. This is write '1' to clear bit |
| 16<br>TMR1IRQ | Timer 1 IRQ<br>Timer Comparator 1 Interrupt Status bit: Indiates T1CMP comparator value matched event timer counter. This is write '1' to clear bit |
| 15<br>CRCVALID | CRC Valid Status<br>Code Redundancy Check Valid: This flag indicates the compare result between the FCS field, in the most-recently received frame, and the internally calculated CRC value. This flag is cleared at next receiver warm up.<br>0b - Rx FCS != calculated CRC (incorrect)<br>1b - Rx FCS = calculated CRC (correct) |
| 14<br>CCA | CCA Status<br>Channel IDLE/BUSY indicator. This indicator is valid at CCAIRQ and also at SEQIRQ. This flag is cleared at next receiver warm up.<br>0b - IDLE<br>1b - BUSY |
| 13<br>SRCADDR | Source Address Match Status<br>If Source Address Management is engaged, meaning at least one of the following bits is set:<br>SAP0_EN<br>SAA0_EN<br>SAP1_EN<br>SAA1_EN<br><br>Then SRCADDR will be set to 1 if the packet just received is a poll request (PI=1), *and* at least one of the following conditions is met:<br>SAP0_EN and SAP0_ADDR_PRESENT<br>SAA0_EN and SAA0_ADDR_ABSENT<br>SAP1_EN and SAP1_ADDR_PRESENT<br>SAA1_EN and SAA1_ADDR_ABSENT<br><br>If SRCADDR=1, this indicates to SW that the Packet Processor has determined that an auto-TxACK frame must be transmitted with the FramePending subfield of the FrameControlField set to 1. HW will assemble and transmit this Ack packet. If the above conditions are not met, SRCADDR will be cleared to 0. |
| 12<br>PI | Poll Indication<br>0b - the received packet was not a data request<br>1b - the received packet was a data request, regardless of whether a Source Address table match occurred, or whether Source Address Management is enabled or not |
| 11<br>ENH_PKT_STATUS | Enhanced Packet Status<br>0b - The last packet received was neither 4e- nor 2015-compliant<br>1b - The last packet received was 4e- or 2015-compliant (RX_FRAME_FILTER register should be queried for additional status bits) |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10<br>TSM_IRQ | TSM IRQ<br>TSM Interrupt Request<br>0b - A TSM Interrupt has not occurred<br>1b - A TSM Interrupt has occurred |
| 9<br>— | Reserved. |
| 8<br>WAKE_IRQ | WAKE Interrupt Request<br>Wake Interrrupt. The RSIM SLEEP_TMR has matched the RSIM ZIGBEE_WAKE register, and DSM has exited. The 802.15.4 EVENT_TMR has resumed counting.<br>0b - A Wake Interrupt has not occurred<br>1b - A Wake Interrupt has occurred |
| 7<br>RX_FRM_PEND | RX Frame Pending<br>Status of the frame pending bit of the frame control field for the most-recently received packet. Read-only. |
| 6<br>PLL_UNLOCK_IRQ | PLL Unlock IRQ<br>PLL Un-lock Interrupt Status bit. A '1' indicates an unlock event has occurred in the PLL. This is write a '1' to clear bit.<br>0b - A PLL Unlock Interrupt has not occurred<br>1b - A PLL Unlock Interrupt has occurred |
| 5<br>FILTERFAIL_IRQ | Filter Fail IRQ<br>Receiver Packet Filter Fail Interrupt Status bit. A '1' indicates that the most-recently received packet has been rejected due to elements within the packet. This is write a '1' to clear bit.<br>In Dual PAN mode, FILTERFAIL_IRQ applies to either or both networks, as follows:<br>**A:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=0, FILTERFAIL_IRQ applies to PAN0.<br>**B:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=1, FILTERFAIL_IRQ applies to PAN1.<br>**C:** If PAN0 and PAN1 occupy the same channel, FILTERFAIL_IRQ is the logical 'AND' of the individual PANs' Filter Fail status.<br>0b - A Filter Fail Interrupt has not occurred<br>1b - A Filter Fail Interrupt has occurred |
| 4<br>RXWTRMRKIRQ | Receive Watermark IRQ<br>Receiver Byte Count Water Mark Interrupt Status bit. A '1' indicates that the number of bytes specified in the RX_WTR_MARK register has been reached. This is write a '1' to clear bit.<br>0b - A Receive Watermark Interrupt has not occurred<br>1b - A Receive Watermark Interrupt has occurred |
| 3<br>CCAIRQ | CCA IRQ<br>Clear Channel Assessment Interrupt Status bit. A '1' indicates completion of CCA operation. This is write '1' to clear bit.<br>0b - A CCA Interrupt has not occurred<br>1b - A CCA Interrupt has occurred |
| 2<br>RXIRQ | RX IRQ<br>Receiver Interrupt Status bit. A '1' indicates the completion of a receive operation. This is write a '1' to clear bit.<br>0b - A RX Interrupt has not occurred<br>1b - A RX Interrupt has occurred |
| 1 | TX IRQ |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TXIRQ | Transmitter Interrupt Status bit. A '1' indicates the completion of a transmit operation. This is write a '1' to clear bit.<br>0b - A TX Interrupt has not occurred<br>1b - A TX Interrupt has occurred |
| 0<br>SEQIRQ | Sequencer IRQ<br>Sequence-end Interrupt Status bit. A '1' indicates the completion of an autosequence. This interrupt will assert whenever the Sequence Manager transitions from non-idle to idle state, for any reason. This is write a '1' to clear bit.<br>0b - A Sequencer Interrupt has not occurred<br>1b - A Sequencer Interrupt has occurred |

## A.3.2.3 PHY CONTROL (PHY_CTRL)

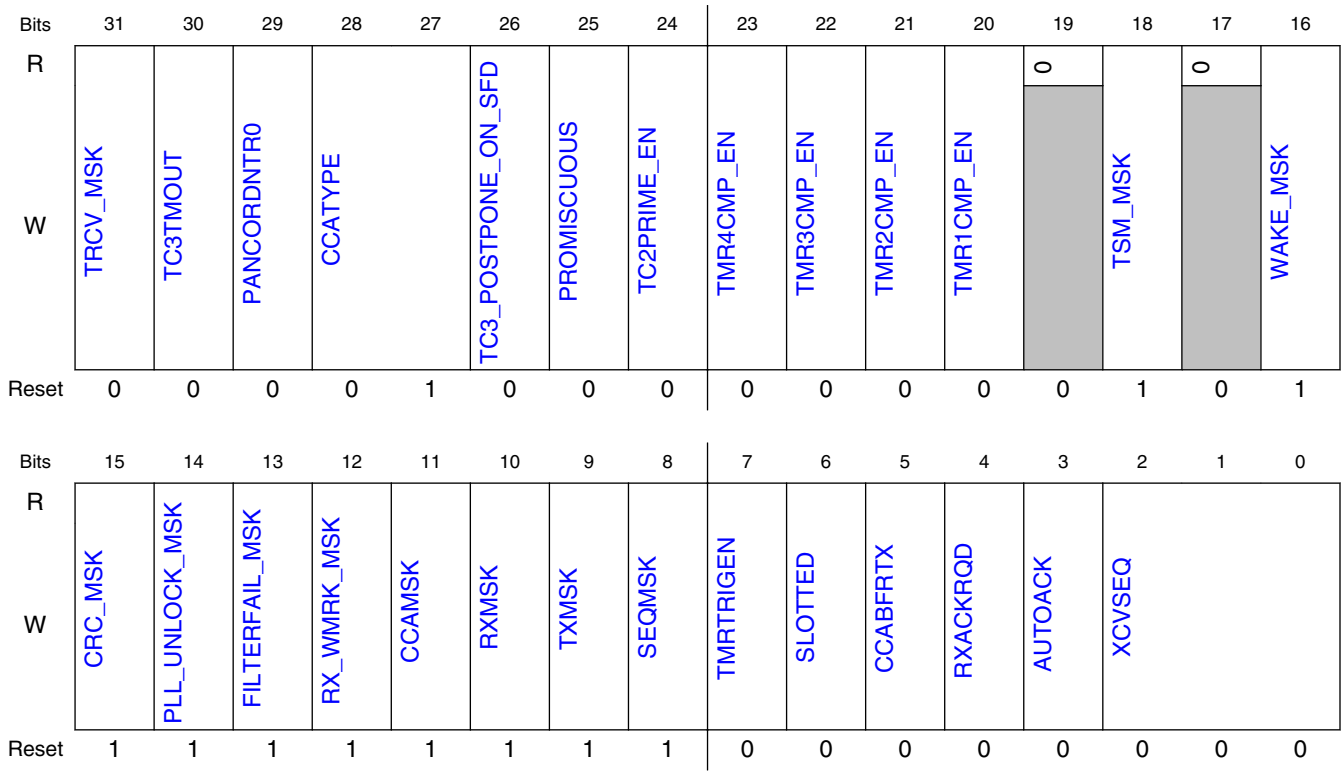
### A.3.2.3.1 Offset

| Register | Offset |
|---|---|
| PHY_CTRL | 4h |

### A.3.2.3.2 Function

PHY Control Register

### A.3.2.3.3 Diagram




### A.3.2.3.4 Fields

| Field | Function |
|---|---|
| 31<br>TRCV_MSK | Transceiver Global Interrupt Mask<br>Transceiver Global Interrupt Mask<br>0b - Enable any unmasked interrupt source to assert zigbee interrupt<br>1b - Mask all interrupt sources from asserting zigbee interrupt |
| 30<br>TC3TMOUT | TMR3 Timeout Enable<br>TMR3 Timeout Enable<br>0b - TMR3 is a software timer only<br>1b - Enable TMR3 to abort Rx or CCCA operations. |
| 29<br>PANCORDNTR0 | Device is a PAN Coordinator on PAN0<br>Device is a PAN Coordinator on PAN0. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| 28-27<br>CCATYPE | Clear Channel Assessment Type<br>Clear Channel Assessment Type. Selects one of four possible functions for CCA or Energy Detect, per below.<br>00b - ENERGY DETECT<br>01b - CCA MODE 1<br>10b - CCA MODE 2<br>11b - CCA MODE 3 |
| 26 | Postpone TC3 Timeout On SFD Enable |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TC3_POSTPONE_ON_SFD | If this bit is set, if a TMR3 timeout occurs while a packet is being received (*rx_sfd_detect* = 1), the timeout-related abort will be deferred untl the packet reception is completed. If the packet is good, HW will signal Data Indication; otherwise the 802.15.4 ZSM (Sequence Manager) will return to SEQ_IDLE and await further commanding.<br>0b - TC3 Abort will occur on TMR3 timeout, regardless of rx_sfd_detect<br>1b - TC3 Abort will be deferred on TMR3 timeout if rx_sfd_detect is asserted; otherwise the TC3 Abort will occur immediately |
| 25<br>PROMISCUOUS | Promiscuous Mode Enable<br>Bypasses most packet filtering.<br>0b - normal mode<br>1b - all packet filtering except frame length checking (FrameLength>=5 and FrameLength<=127) is bypassed. |
| 24<br>TC2PRIME_EN | Timer 2 Prime Compare Enable<br>0b - Don't allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ<br>1b - Allow a match of the lower 16 bits of Event Timer to T2PRIMECMP to set TMR2IRQ |
| 23<br>TMR4CMP_EN | Timer 4 Compare Enable<br>0b - Don't allow an Event Timer Match to T4CMP to set TMR4IRQ<br>1b - Allow an Event Timer Match to T4CMP to set TMR4IRQ |
| 22<br>TMR3CMP_EN | Timer 3 Compare Enable<br>0b - Don't allow an Event Timer Match to T3CMP to set TMR3IRQ<br>1b - Allow an Event Timer Match to T3CMP to set TMR3IRQ |
| 21<br>TMR2CMP_EN | Timer 2 Compare Enable<br>0b - Don't allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ<br>1b - Allow an Event Timer Match to T2CMP or T2PRIMECMP to set TMR2IRQ |
| 20<br>TMR1CMP_EN | Timer 1 Compare Enable<br>0b - Don't allow an Event Timer Match to T1CMP to set TMR1IRQ<br>1b - Allow an Event Timer Match to T1CMP to set TMR1IRQ |
| 19<br>— | Reserved. |
| 18<br>TSM_MSK | Reserved.<br>0b - allows assertion of a TSM interrupt to generate a zigbee interrupt<br>1b - Assertion of a TSM interrupt will set the TSM_IRQ status bit, but a zigbee interrupt is not generated |
| 17<br>— | Reserved. |
| 16<br>WAKE_MSK | Reserved.<br>0b - Allows a wakeup from DSM to generate a zigbee interrupt<br>1b - Wakeup from DSM will set the WAKE_IRQ status bit, but a zigbee interrupt is not generated |
| 15<br>CRC_MSK | CRC Mask<br>CRC Mask<br>0b - sequence manager ignores CRCVALID and considers the receive operation complete after the last octet of the frame has been received.<br>1b - sequence manager requires CRCVALID=1 at the end of the received frame in order for the receive operation to complete successfully; if CRCVALID=0, sequence manager will return to preamble-detect mode after the last octet of the frame has been received. |
| 14<br>PLL_UNLOCK_MSK | PLL Unlock Interrupt Mask<br>0b - allows PLL unlock event to generate a zigbee interrupt<br>1b - A PLL unlock event will set the PLL_UNLOCK_IRQ status bit, but a zigbee interrupt is not generated |
| 13 | FilterFail Interrupt Mask |

*Table continues on the next page...*

| Field | Function |
|---|---|
| FILTERFAIL_MSK | FilterFail Interrupt Mask<br>0b - allows Packet Processor Filtering Failure to generate a zigbee interrupt<br>1b - A Packet Processor Filtering Failure will set the FILTERFAIL_IRQ status bit, but a zigbee interrupt is not generated |
| 12<br>RX_WMRK_MSK | RX Watermark Interrupt Mask<br>RX Watermark Interrupt Mask<br>0b - allows a Received Byte Count match to the RX_WTR_MARK threshold register to generate a zigbee interrupt<br>1b - A Received Byte Count match to the RX_WTR_MARK threshold register will set the RXWTRMRKIRQ status bit, but a zigbee interrupt is not generated |
| 11<br>CCAMSK | CCA Interrupt Mask<br>CCA Interrupt Mask<br>0b - allows completion of a CCA operation to generate a zigbee interrupt<br>1b - Completion of a CCA operation will set the CCA status bit, but a zigbee interrupt is not generated |
| 10<br>RXMSK | RX Interrupt Mask<br>0b - allows completion of a RX operation to generate a zigbee interrupt<br>1b - Completion of a RX operation will set the RXIRQ status bit, but a zigbee interrupt is not generated |
| 9<br>TXMSK | TX Interrupt Mask<br>0b - allows completion of a TX operation to generate a zigbee interrupt<br>1b - Completion of a TX operation will set the TXIRQ status bit, but a zigbee interrupt is not generated |
| 8<br>SEQMSK | Sequencer Interrupt Mask<br>0b - allows completion of an autosequence to generate a zigbee interrupt<br>1b - Completion of an autosequence will set the SEQIRQ status bit, but a zigbee interrupt is not generated |
| 7<br>TMRTRIGEN | Timer2 Trigger Enable<br>0b - programmed sequence initiates immediately upon write to XCVSEQ.<br>1b - allow timer TC2 (or TC2') to initiate a preprogrammed sequence (see XCVSEQ register). |
| 6<br>SLOTTED | Slotted Mode<br>Slotted Mode, for beacon-enabled networks. Applies only to Sequences T, TR, and R, ignored during all other sequences. Used, in concert with CCABFRTX, to determine how many CCA measurements are required prior to a transmit operation. Also used during R sequence to determine whether the ensuing transmit acknowledge frame (if any) needs to be synchronized to a backoff slot boundary. |
| 5<br>CCABFRTX | CCA Before TX<br>Applies only to Sequences T and TR, ignored during all other sequences.<br>0b - no CCA required, transmit operation begins immediately.<br>1b - at least one CCA measurement is required prior to the transmit operation (see also SLOTTED). |
| 4<br>RXACKRQD | Receive Acknowledge Frame required<br>Applies only to Sequence TR, ignored during all other sequences.<br>0b - An ordinary receive frame (any type of frame) follows the transmit frame.<br>1b - A receive Ack frame is expected to follow the transmit frame (non-Ack frames are rejected). |
| 3<br>AUTOACK | Auto Acknowledge Enable<br>Applies only to Sequence R and Sequence TR, ignored during other sequences<br>0b - sequence manager will not follow a receive frame with a Tx Ack frame, under any conditions; the autosequence will terminate after the receive frame.<br>1b - sequence manager will follow a receive frame with an automatic hardware-generated Tx Ack frame, assuming other necessary conditions are met. |

*Table continues on the next page...*

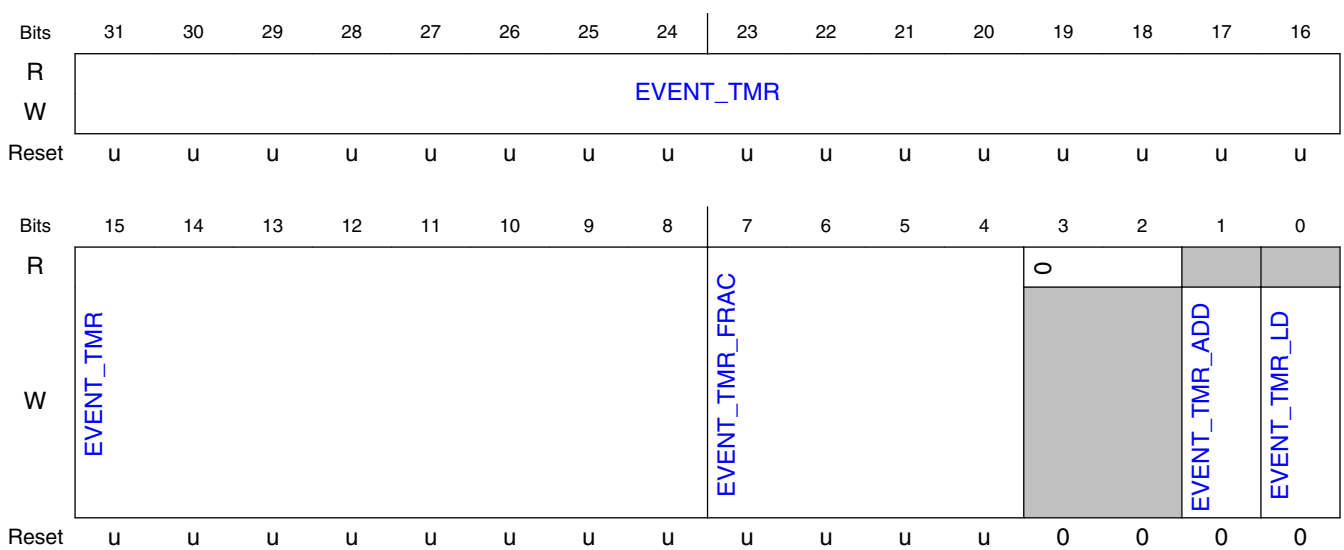
| Field | Function |
|---|---|
| 2-0<br>XCVSEQ | 802.15.4 Transceiver Sequence Selector<br>The Transceiver Sequence Selector register selects an autosequence for the sequence manager to execute. Sequence initiation can be immediate, or scheduled (see TMRTRIGEN). A write of XCVSEQ=IDLE will abort any ongoing sequence. A write of XCVSEQ=IDLE must always be performed after a sequence is complete, and before a new sequence is programmed. Any write to XCVSEQ other than XCVSEQ=IDLE during an ongoing sequence, shall be ignored. The mapping of XCVSEQ to sequence types is as follows:<br>000b - I (IDLE)<br>001b - R (RECEIVE)<br>010b - T (TRANSMIT)<br>011b - C (CCA)<br>100b - TR (TRANSMIT/RECEIVE)<br>101b - CCCA (CONTINUOUS CCA)<br>110b - Reserved<br>111b - Reserved |

## A.3.2.4 EVENT TIMER (EVENT_TMR)

### A.3.2.4.1 Offset

| Register | Offset |
|---|---|
| EVENT_TMR | 8h |

### A.3.2.4.2 Diagram



#### A.3.2.4.3 Fields

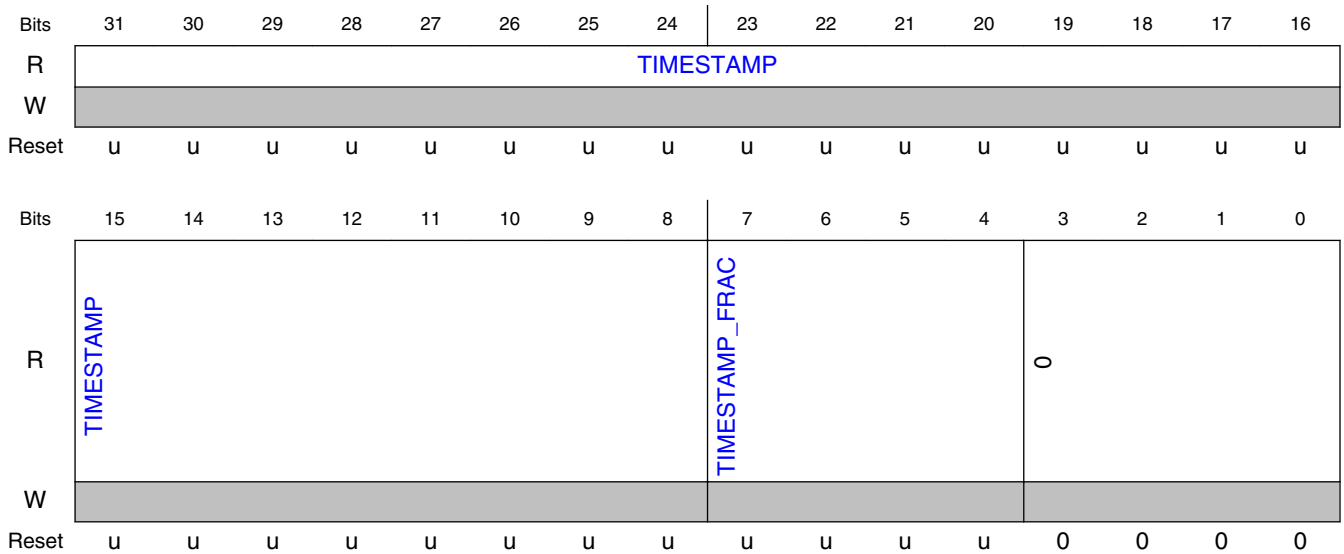
| Field | Function |
|---|---|
| 31-8<br>EVENT_TMR | Event Timer Integer Component<br>The EVENT_TMR represents a 28-bit number with 24 Integer bits and 4 Fractional bits. This field is the integer component, which appears to the left of the decimal point. Most 802.15.4 operations which relate to the Event Timer require only the integer component. |
| 7-4<br>EVENT_TMR_FRAC | Event Timer Fractional Component<br>The EVENT_TMR represents a 28-bit number with 24 Integer bits and 4 Fractional bits. This field is the fractional component, which appears to the right of the decimal point. The fractional component can be ignored for most 802.15.4 operations which relate to the Event Timer; it has been provided only to allow for microsecond accuracy when software updates the Event Timer after a long period of DSM (sleep) mode |
| 3-2<br>— | Reserved. |
| 1<br>EVENT_TMR_ADD | Event Timer Add Enable<br>Increments the Event Timer by the values written to the EVENT_TMR[23:0] and EVENT_TMR_FRAC[3:0] fields of this register. |
| 0<br>EVENT_TMR_LD | Event Timer Load Enable<br>Loads the Event Timer with the values written to the EVENT_TMR[23:0] and EVENT_TMR_FRAC[3:0] fields of this register. |

### A.3.2.5 TIMESTAMP (TIMESTAMP)

#### A.3.2.5.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP | Ch |

### A.3.2.5.2 Diagram




### A.3.2.5.3 Fields

| Field | Function |
|---|---|
| 31-8<br>TIMESTAMP | Timestamp<br>Holds the latched value of the integer portion of the Event Timer current time corresponding to the beginning of the most recently received packet, at the point of SFD detection |
| 7-4<br>TIMESTAMP_FRAC | Timestamp Fractional<br>Holds the latched value of the fractional portion of the Event Timer current time corresponding to the beginning of the most recently received packet, at the point of SFD detection |
| 3-0<br>— | Reserved. |

## A.3.2.6 T1 COMPARE (T1CMP)

### A.3.2.6.1 Offset

| Register | Offset |
|---|---|
| T1CMP | 10h |

### A.3.2.6.2 Function

TMR1 Compare Value

### A.3.2.6.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | T1CMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T1CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.6.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T1CMP | TMR1 Compare Value<br>TMR1 compare value. If TMR1CMP_EN=1 and the Event Timer matches this value, TMR1IRQ is set. |

## A.3.2.7 T2 COMPARE (T2CMP)

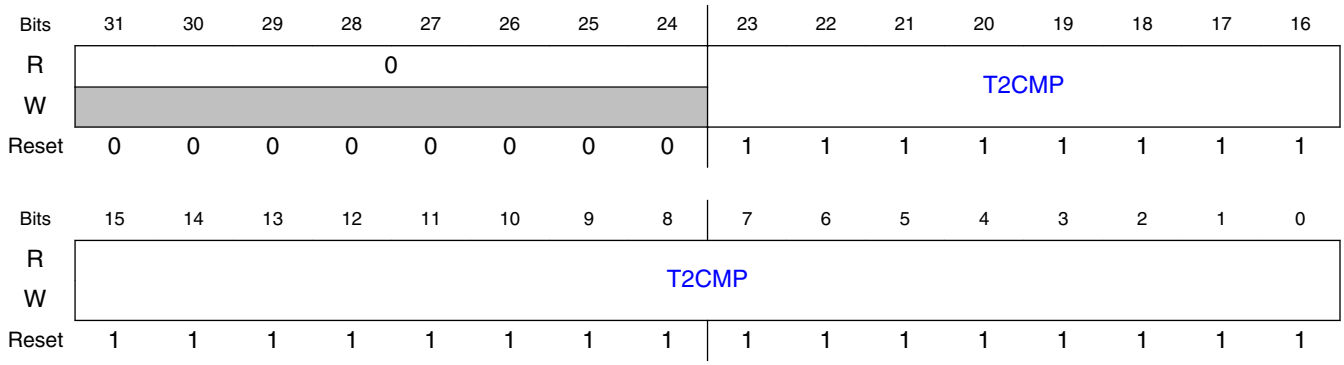
### A.3.2.7.1 Offset

| Register | Offset |
|---|---|
| T2CMP | 14h |

### A.3.2.7.2 Function

TMR2 Compare Value

### A.3.2.7.3 Diagram




### A.3.2.7.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T2CMP | TMR2 Compare Value<br>TMR2 compare value. If TMR2CMP_EN=1 and TC2PRIME_EN=0 and the Event Timer matches this value, TMR2IRQ is set. |

## A.3.2.8 T2 PRIME COMPARE (T2PRIMECMP)

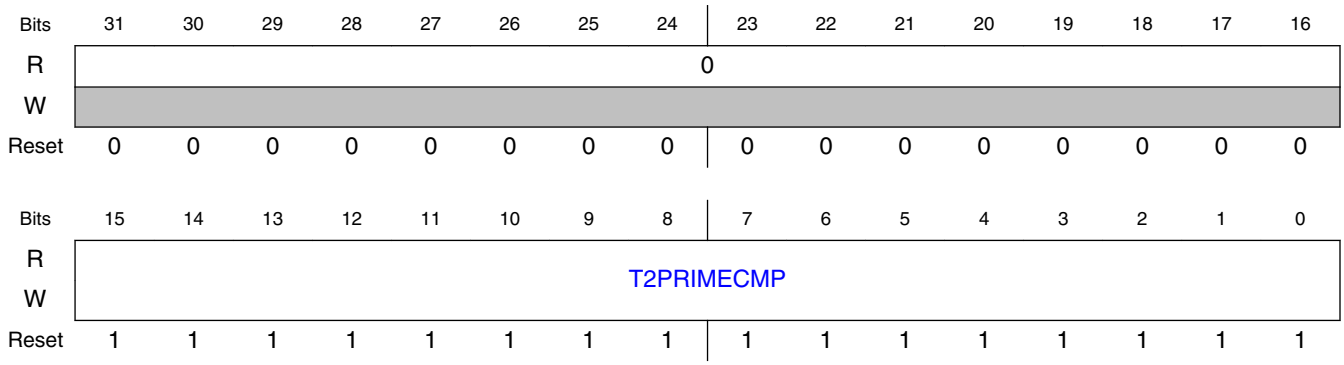
### A.3.2.8.1 Offset

| Register | Offset |
|---|---|
| T2PRIMECMP | 18h |

### A.3.2.8.2 Function

TMR2 Prime Compare Value

### A.3.2.8.3 Diagram




### A.3.2.8.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-0<br>T2PRIMECMP | TMR2 Prime Compare Value<br>TMR2 compare value. If TMR2CMP_EN=1 and TC2PRIME_EN=1 and the Event Timer matches this value, TMR2IRQ is set. |

## A.3.2.9 T3 COMPARE (T3CMP)

### A.3.2.9.1 Offset

| Register | Offset |
|---|---|
| T3CMP | 1Ch |

### A.3.2.9.2 Function

TMR3 Compare Value

### A.3.2.9.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | T3CMP | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T3CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.9.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T3CMP | TMR3 Compare Value<br>TMR3 compare value. If TMR3CMP_EN=1 and the Event Timer matches this value, TMR3IRQ is set. |

## A.3.2.10 T4 COMPARE (T4CMP)

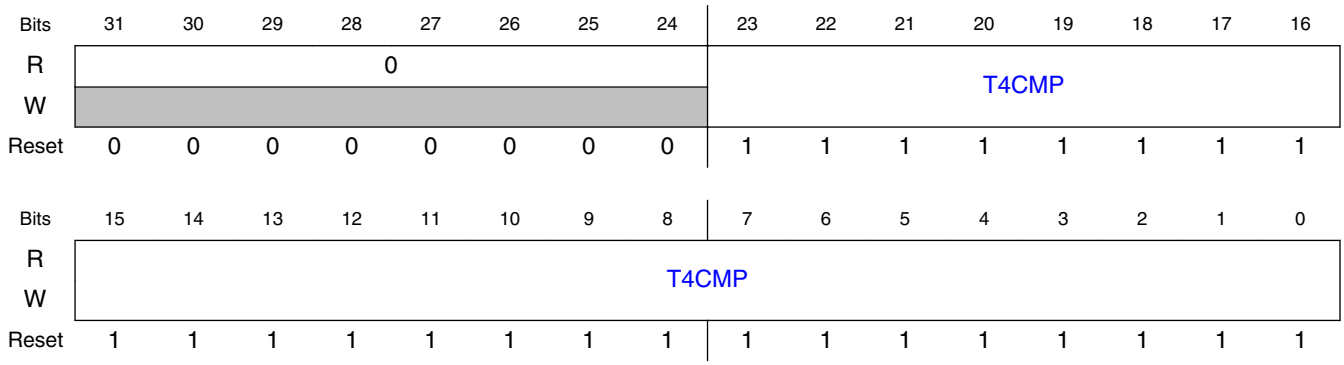
### A.3.2.10.1 Offset

| Register | Offset |
|---|---|
| T4CMP | 20h |

### A.3.2.10.2 Function

TMR4 Compare Value

### A.3.2.10.3 Diagram




### A.3.2.10.4 Fields

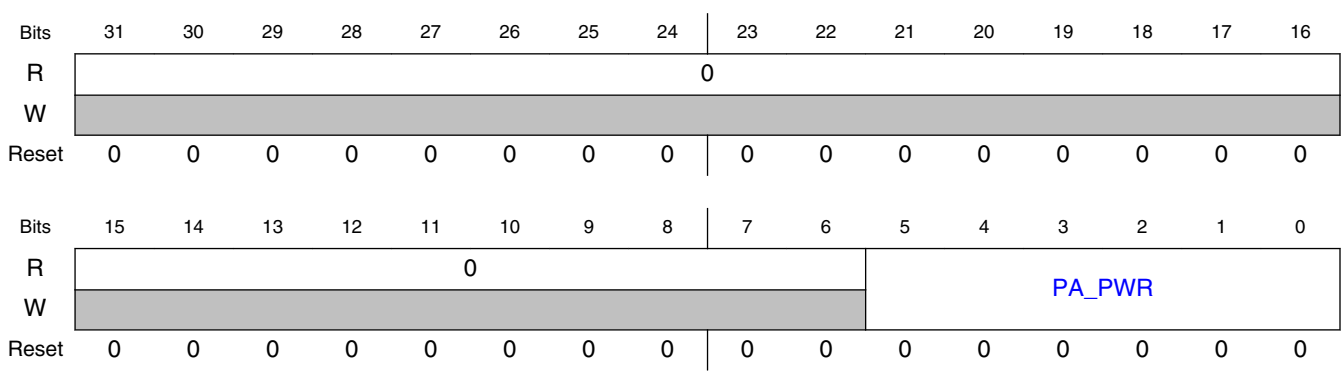
| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>T4CMP | TMR4 Compare Value<br>TMR4 compare value. If TMR4CMP_EN=1 and the Event Timer matches this value, TMR4IRQ is set. |

## A.3.2.11 PA POWER (PA_PWR)

### A.3.2.11.1 Offset

| Register | Offset |
|---|---|
| PA_PWR | 24h |

### A.3.2.11.2 Diagram



### A.3.2.11.3 Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>PA_PWR | PA Power |

## A.3.2.12 CHANNEL NUMBER 0 (CHANNEL_NUM0)

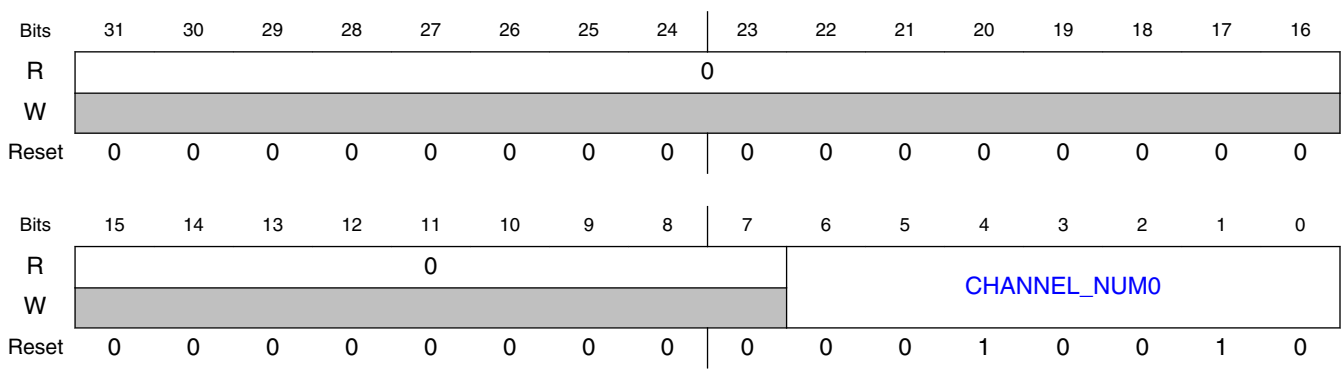
### A.3.2.12.1 Offset

| Register | Offset |
|---|---|
| CHANNEL_NUM0 | 28h |

### A.3.2.12.2 Function

Channel Number for PAN0

### A.3.2.12.3 Diagram




### A.3.2.12.4 Fields

| Field | Function |
|---|---|
| 31-7 | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| — | |
| 6-0<br>CHANNEL_NUM0 | Channel Number for PAN0<br>This is the mapped channel number used to transmit and receive Zigbee packets. If Dual PAN is engaged, this register applies to PAN0. CHANNEL_NUM0 should be in the range:<br>11 <= CHANNEL_NUM0 <= 26 |

## A.3.2.13 LQI AND RSSI (LQI_AND_RSSI)

### A.3.2.13.1 Offset

| Register | Offset |
|---|---|
| LQI_AND_RSSI | 2Ch |

### A.3.2.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CCA1_ED_FNL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RSSI | | | | | | | | LQI_VALUE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### A.3.2.13.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-16<br>CCA1_ED_FNL | Final Result for CCA Mode 1 and Energy Detect<br>Output register to show final averaged RSSI value or compensated value of the same at the end of a CCA Mode1 or Energy Detect computation. |
| 15-8<br>RSSI | RSSI Value<br>Received Signal Strength Indicator, in dBm |
| 7-0 | LQI Value |

| Field | Function |
|---|---|
| LQI_VALUE | Link Quality Indicator for the most recently received packet. (LQI is also available in the Packet Buffer, at the end of the received packet data) |

## A.3.2.14 MAC SHORT ADDRESS 0 (MACSHORTADDRS0)

### A.3.2.14.1 Offset

| Register | Offset |
|---|---|
| MACSHORTADDRS0 | 30h |

### A.3.2.14.2 Function

MAC SHORT ADDRESS for PAN0

### A.3.2.14.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACSHORTADDRS0 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACPANID0 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.14.4 Fields

| Field | Function |
|---|---|
| 31-16<br>MACSHORTADDRS0 | MAC SHORT ADDRESS FOR PAN0<br>MAC Short Address for PAN0, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| 15-0<br>MACPANID0 | MAC PAN ID for PAN0<br>MAC PAN ID for PAN0. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID |

| Field | Function |
|---|---|
| | against this register. Also, if PANCORDNTR0=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |

## A.3.2.15 MAC LONG ADDRESS 0 LSB (MACLONGADDRS0_LSB)

### A.3.2.15.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS0_LSB | 34h |

### A.3.2.15.2 Function

MAC LONG ADDRESS for PAN0 LSB

### A.3.2.15.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.15.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS0_LSB | MAC LONG ADDRESS for PAN0 LSB<br>MAC Long Address for PAN0, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

# A.3.2.16 MAC LONG ADDRESS 0 MSB (MACLONGADDRS0_MSB)

## A.3.2.16.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS0_MSB | 38h |

## A.3.2.16.2 Function

MAC LONG ADDRESS for PAN0 MSB

## A.3.2.16.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS0_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.3.2.16.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS0_MSB | MAC LONG ADDRESS for PAN0 MSB<br>MAC Long Address for PAN0, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

# A.3.2.17 RECEIVE FRAME FILTER (RX_FRAME_FILTER)

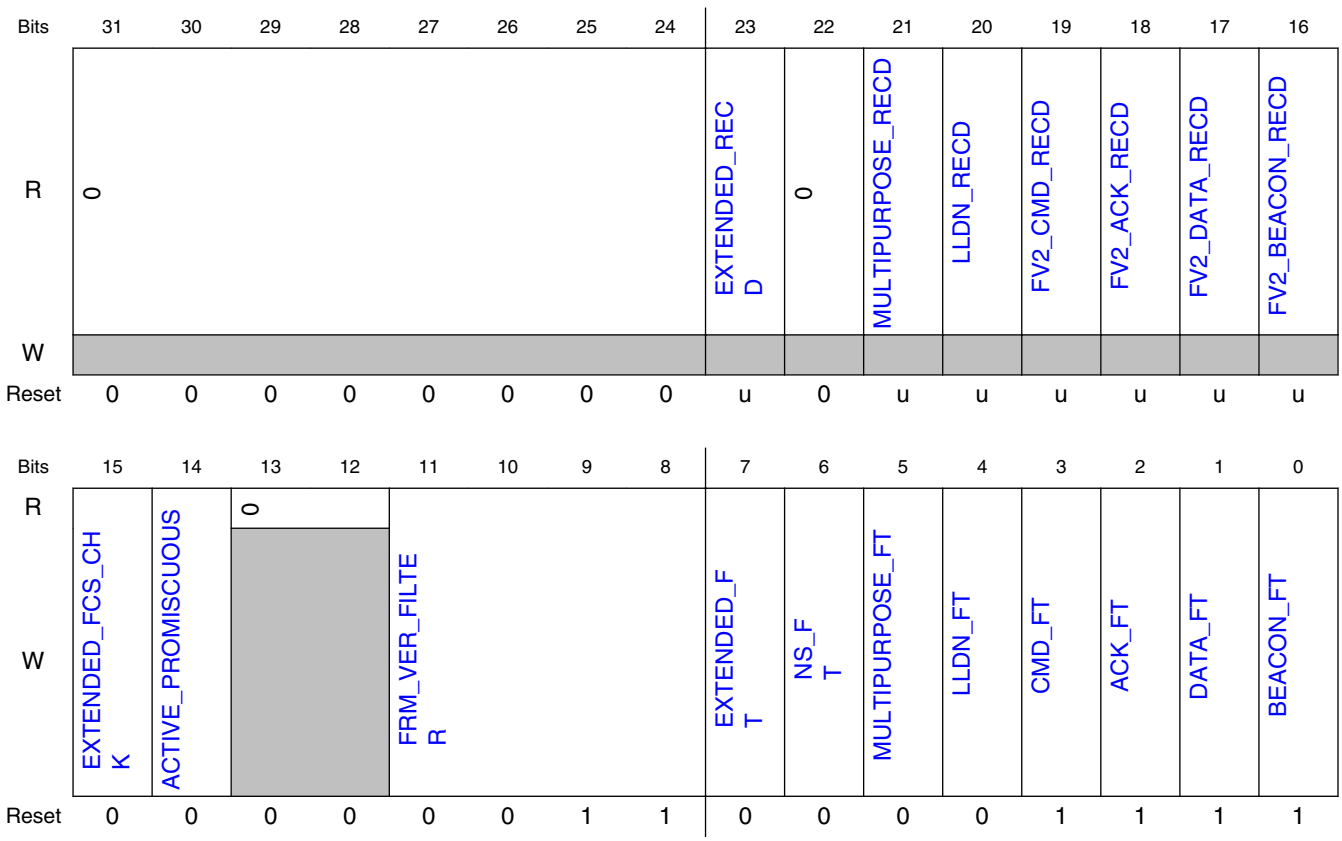
## A.3.2.17.1 Offset

| Register | Offset |
|---|---|
| RX_FRAME_FILTER | 3Ch |

### A.3.2.17.2 Function

RECEIVE FRAME FILTER

### A.3.2.17.3 Diagram




### A.3.2.17.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23<br>EXTENDED_RECD | Extended Packet Received<br>0b - The last packet received was not Frame Type EXTENDED<br>1b - The last packet received was Frame Type EXTENDED, and EXTENDED_FT=1 to allow such packets. |
| 22<br>— | Reserved. |
| 21 | Multipurpose Packet Received<br>0b - last packet received was not Frame Type MULTIPURPOSE |

*Table continues on the next page...*

| Field | Function |
|---|---|
| MULTIPURPOSE_RECD | 1b - The last packet received was Frame Type MULTIPURPOSE, and MULTIPURPOSE_FT=1 to allow such packets. |
| 20<br>LLDN_RECD | LLDN Packet Received<br>0b - The last packet received was not Frame Type LLDN<br>1b - The last packet received was Frame Type LLDN, and LLDN_FT=1 to allow such packets. |
| 19<br>FV2_CMD_RECD | Frame Version 2 MAC Command Packet Received<br>0b - The last packet received was not Frame Type MAC Command with Frame Version 2<br>1b - The last packet received was Frame Type MAC Command with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 18<br>FV2_ACK_RECD | Frame Version 2 Acknowledge Packet Received<br>0b - The last packet received was not Frame Type Ack with Frame Version 2<br>1b - The last packet received was Frame Type Ack with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 17<br>FV2_DATA_RECD | Frame Version 2 Data Packet Received<br>0b - The last packet received was not Frame Type Data with Frame Version 2<br>1b - The last packet received was Frame Type Data with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 16<br>FV2_BEACON_RECD | Frame Version 2 Beacon Packet Received<br>0b - The last packet received was not Frame Type Beacon with Frame Version 2<br>1b - The last packet received was Frame Type Beacon with Frame Version 2, and FRM_VER_FILTER[2]=1 to allow such packets |
| 15<br>EXTENDED_FCS_CHK | Verify FCS on Frame Type Extended<br>0b - Packet Processor will not check FCS for Frame Type EXTENDED (default)<br>1b - Packet Processor will check FCS at end-of-packet based on packet length derived from PHR, for Frame Type EXTENDED |
| 14<br>ACTIVE_PROMISCUOUS | Active Promiscuous<br>0b - normal operation<br>1b - Provide Data Indication on all received packets under the same rules which apply in PROMISCUOUS mode, however acknowledge those packets under rules which apply in non-PROMISCUOUS mode |
| 13-12<br>— | Reserved. |
| 11-8<br>FRM_VER_FILTER | Frame Version selector.<br>The incoming packet's Frame Control Field is parsed to obtain the FrameVersion subfield, and that value is compared against this register, in accordance with the following:<br>xxx1: Accept received packets with FrameVersion=00<br>xx1x: Accept received packets with FrameVersion=01<br>x1xx: Accept received packets with FrameVersion=10<br>1xxx: Accept received packets with FrameVersion=11<br>These filtering rules apply to Beacon, Acknowledge, Data, and MAC Command Frame Types, since these frame types require a 2-octet Frame Control Field which embeds a 2-bit FrameVersion subfield. Later frame types introduced in the 802.15.4e addendum (LLDN, Multipurpose) don't guarantee a FrameVersion subfield with the original meaning, so these filtering rules do not apply to these frame types. See registers LLDN_FT and MULTIPURPOSE_FT.<br>For Acknowledge frames, FrameVersion bits are ignored by the Packet Processor, irrespective of FRM_VER_FILTER. |
| 7<br>EXTENDED_FT | Extended Frame Type Enable<br>0b - reject all Extended frames<br>1b - Extended frame type enabled (Frame Type 7). |
| 6 | "Not Specified" Frame Type Enable |

*Table continues on the next page...*

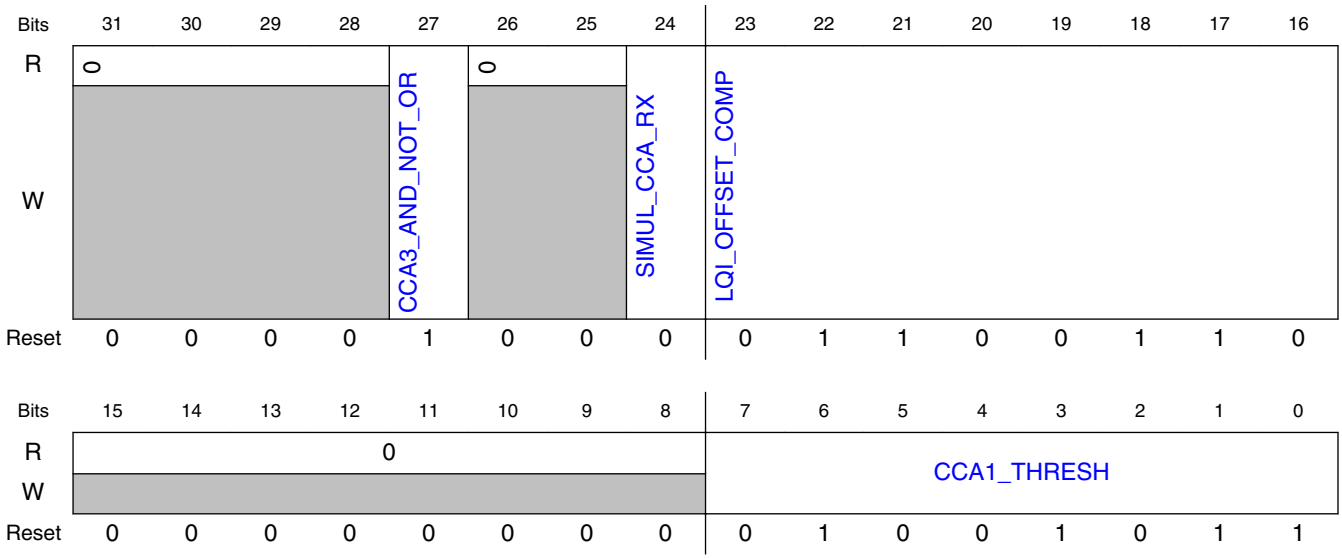
| Field | Function |
|---|---|
| NS_FT | This bit enables reception of all Frame Types not covered by the other _FT bits in this register.<br>0b - reject all "Not Specified" frames<br>1b - Not-specified (reserved) frame type enabled. Applies to Frame Type 6. No packet filtering is performed, except for frame length checking (FrameLength>=5 and FrameLength<=127). No AUTOACK is transmitted for this Frame Type |
| 5<br>MULTIPURPOSE_FT | Multipurpose Frame Type Enable<br>0b - reject all Multipurpose frames<br>1b - Multipurpose frame type enabled (Frame Type 5). |
| 4<br>LLDN_FT | LLDN Frame Type Enable<br>0b - reject all LLDN frames<br>1b - LLDN frame type enabled (Frame Type 4). |
| 3<br>CMD_FT | MAC Command Frame Type Enable<br>0b - reject all MAC Command frames<br>1b - MAC Command frame type enabled. |
| 2<br>ACK_FT | Ack Frame Type Enable<br>0b - reject all Acknowledge frames<br>1b - Acknowledge frame type enabled. |
| 1<br>DATA_FT | Data Frame Type Enable<br>0b - reject all Beacon frames<br>1b - Data frame type enabled. |
| 0<br>BEACON_FT | Beacon Frame Type Enable<br>0b - reject all Beacon frames<br>1b - Beacon frame type enabled. |

## A.3.2.18 CCA AND LQI CONTROL (CCA_LQI_CTRL)

### A.3.2.18.1 Offset

| Register | Offset |
|---|---|
| CCA_LQI_CTRL | 40h |

## A.3.2.18.2 Diagram




## A.3.2.18.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>CCA3_AND_NOT_OR | CCA Mode 3 AND not OR<br>Determines the way CCA3 is required to be detected<br>0b - CCA1 or CCA2<br>1b - CCA1 and CCA2 |
| 26-25<br>— | Reserved. |
| 24<br>SIMUL_CCA_RX | Simultaneous CCA and Receive Enable<br>During autosequences of type Sequence T or Sequence TR with CCA before TX, this bit allows a packet to be received if preamble and SFD are detected during the CCA measurement window.<br>0b - Packets can't be received during CCA measurement<br>1b - Packet reception is enabled during CCA measurement if preamble and SFD are detected |
| 23-16<br>LQI_OFFSET_COMP | LQI Offset Compensation<br>Programmable amount to offset RSSI based LQI value |
| 15-8<br>— | Reserved. |
| 7-0<br>CCA1_THRESH | CCA Mode 1 Threshold<br>Programmable energy threshold register for CCA mode 1. |

# A.3.2.19 CCA2 CONTROL (CCA2_CTRL)

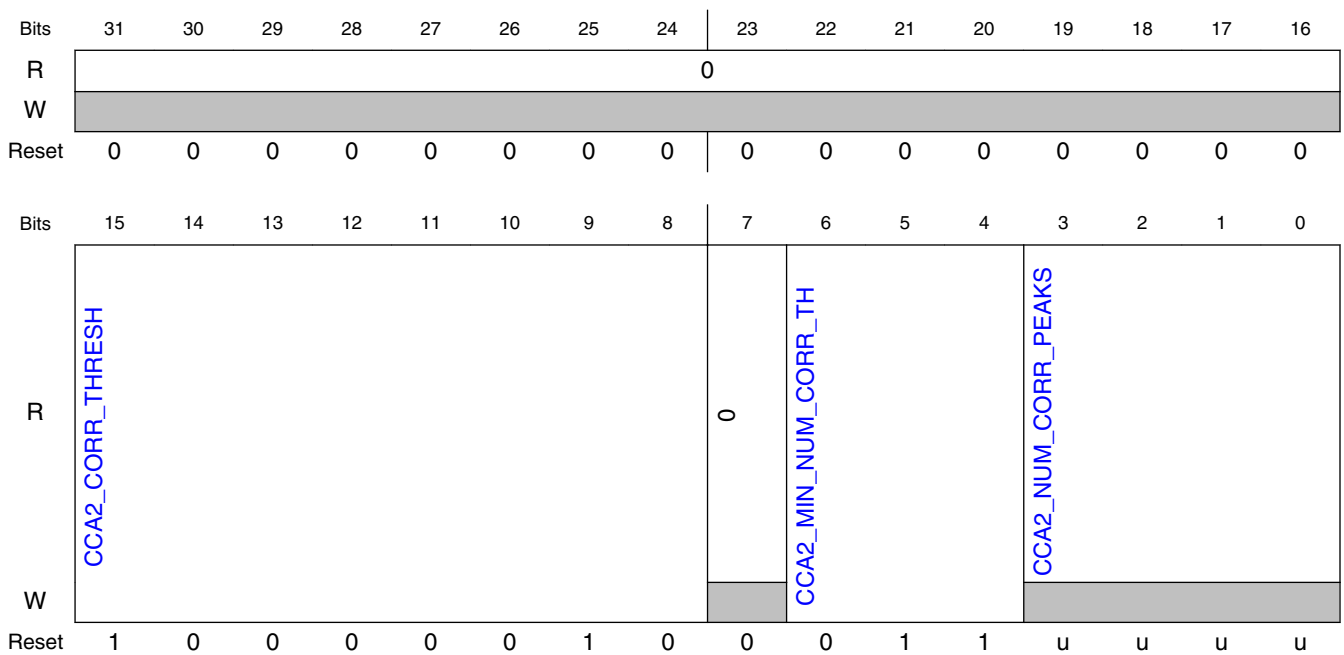
## A.3.2.19.1 Offset

| Register | Offset |
|---|---|
| CCA2_CTRL | 44h |

## A.3.2.19.2 Function

CCA Mode 2 Control

## A.3.2.19.3 Diagram




## A.3.2.19.4 Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15-8<br>CCA2_CORR_THRESH | CCA Mode 2 Correlation Threshold<br>Correlator threshold used to qualify correlator peaks counted by CCA2_NUM_CORR_PEAKS. Correlator peaks exceeding this value increment CCA2_NUM_CORR_PEAKS. |
| 7 | Reserved. |

*Table continues on the next page...*

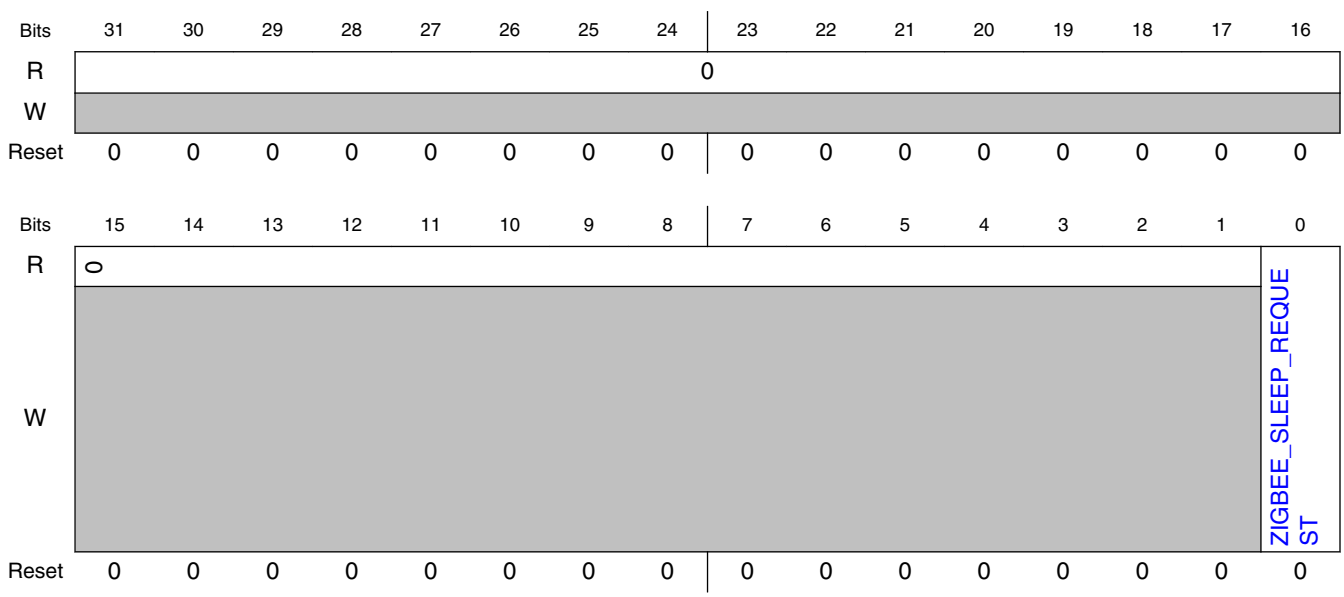
| Field | Function |
|---|---|
| — | |
| 6-4<br>CCA2_MIN_NUM_CORR_TH | CCA Mode 2 Threshold Number of Correlation Peaks<br>Programmable threshold to be compared against number of correlation peaks that exceeded CCA2_CORR_THRESH for detecting CCA mode 2. Number of peaks detected = CCA2_MIN_NUM_CORR_TH + 1; Example: If it is programmed to 3, CCA2 logic looks for at least 4 correlation peaks that crossed the threshold, to indicate channel is idle or busy. |
| 3-0<br>CCA2_NUM_CORR_PEAKS | CCA Mode 2 Number of Correlation Peaks Detected<br>Counts of number of peaks that crossed CCA2_CORR_THRESH in CCA Mode 2 operation |

## A.3.2.20 DSM CONTROL (DSM_CTRL)

### A.3.2.20.1 Offset

| Register | Offset |
|---|---|
| DSM_CTRL | 4Ch |

### A.3.2.20.2 Diagram



### A.3.2.20.3 Fields

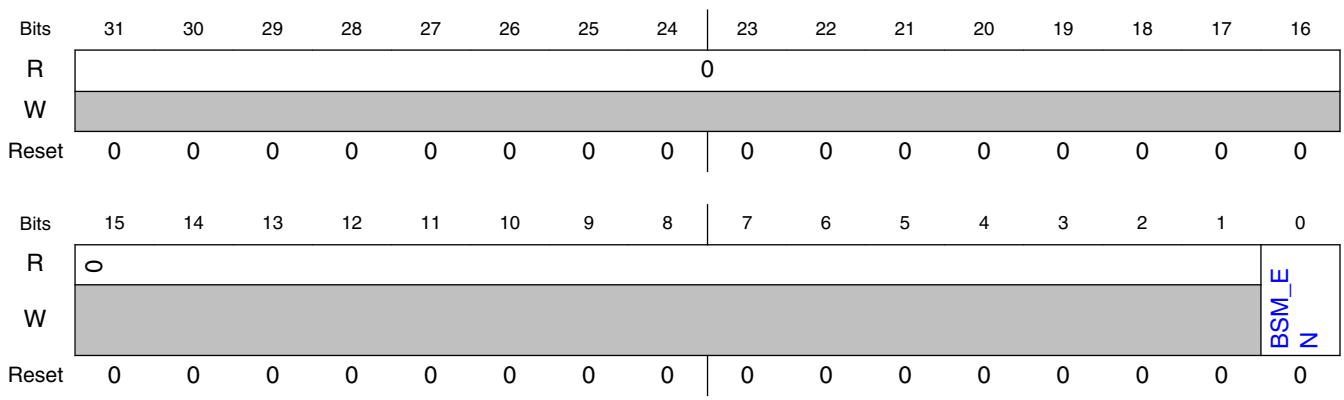
| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>ZIGBEE_SLEEP_REQUEST | 802.15.4 Deep Sleep Mode Request for Manual DSM<br>If 802.15.4 is assigned to Manual DSM (XCVR_CTRL[MAN_DSM_SEL] = 1), setting ZIGBEE_SLEEP_REQUEST to 1 enables Deep Sleep Mode (DSM) to be entered when the RSIM DSM_TIMER matches the RSIM MAN_SLEEP register. |

## A.3.2.21 BSM CONTROL (BSM_CTRL)

### A.3.2.21.1 Offset

| Register | Offset |
|---|---|
| BSM_CTRL | 50h |

### A.3.2.21.2 Diagram




### A.3.2.21.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>BSM_EN | BSM Enable<br>0b - 802.15.4 Bit Streaming Mode Disabled<br>1b - 802.15.4 Bit Streaming Mode Enabled |

# A.3.2.22 MAC SHORT ADDRESS FOR PAN1 (MACSHORTADDRS1)

## A.3.2.22.1 Offset

| Register | Offset |
|---|---|
| MACSHORTADDRS1 | 54h |

## A.3.2.22.2 Function

The MACSHORTADDRS1 register .

## A.3.2.22.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACSHORTADDRS1 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACPANID1 | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## A.3.2.22.4 Fields

| Field | Function |
|---|---|
| 31-16<br>MACSHORTADDRS1 | MAC SHORT ADDRESS for PAN1<br>MAC Short Address for PAN1, for 16-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |
| 15-0<br>MACPANID1 | MAC PAN ID for PAN1<br>MAC PAN ID for PAN1. The packet processor compares the incoming packet's Destination PAN ID against the contents of this register to determine if the packet is addressed to this device; or if the incoming packet is a Beacon frame, the packet processor compares the incoming packet Source PAN ID against this register. Also, if PANCORDNTR1=1, and the incoming packet has no Destination Address field, and if the incoming packet is a Data or MAC Command frame, the packet processor compares the incoming packet Source PAN ID against this register. |

## A.3.2.23 MAC LONG ADDRESS 1 LSB (MACLONGADDRS1_LSB)

### A.3.2.23.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS1_LSB | 58h |

### A.3.2.23.2 Function

MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device.

### A.3.2.23.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_LSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.23.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS1_LSB | MAC LONG ADDRESS for PAN1 LSB<br>MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

## A.3.2.24 MAC LONG ADDRESS 1 MSB (MACLONGADDRS1_MSB)

### A.3.2.24.1 Offset

| Register | Offset |
|---|---|
| MACLONGADDRS1_MSB | 5Ch |

### A.3.2.24.2 Function

MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device.

### A.3.2.24.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MACLONGADDRS1_MSB | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.2.24.4 Fields

| Field | Function |
|---|---|
| 31-0<br>MACLONGADDRS1_MSB | MAC LONG ADDRESS for PAN1 MSB<br>MAC Long Address for PAN1, for 64-bit destination addressing mode. The packet processor compares the incoming packet's Destination Address against the contents of this register to determine if the packet is addressed to this device. |

## A.3.2.25 DUAL PAN CONTROL (DUAL_PAN_CTRL)

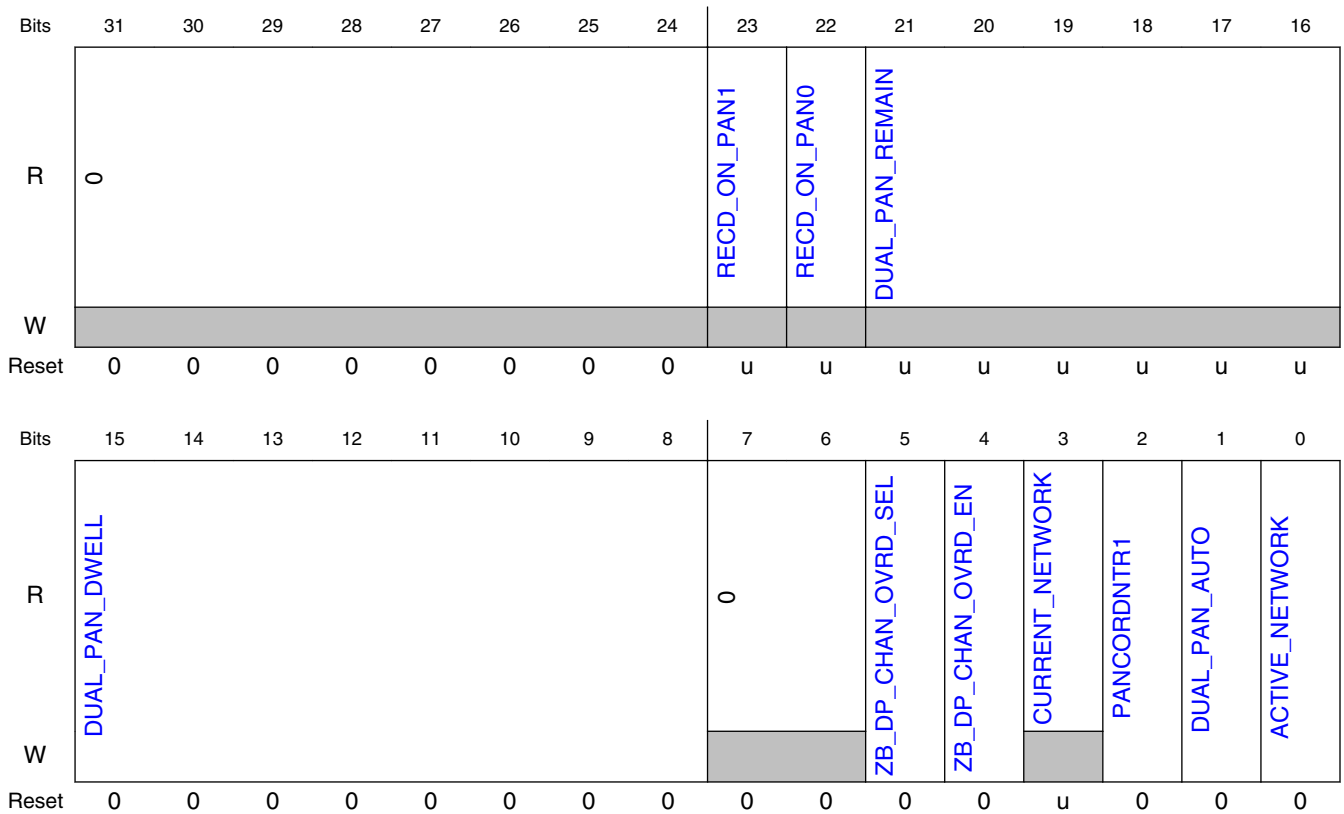
### A.3.2.25.1 Offset

| Register | Offset |
|---|---|
| DUAL_PAN_CTRL | 60h |

## A.3.2.25.2 Function

Dual PAN Control Register

## A.3.2.25.3 Diagram




## A.3.2.25.4 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23<br>RECD_ON_PAN1 | Last Packet was Received on PAN1<br>Indicates the packet which was just received, was received on PAN1. In Dual PAN mode operating on 2 different channels, RECD_ON_PAN1 will be set if CURRENT_NETWORK=1 when the packet was received, regardless of FILTERFAIL status. In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN1 will be set only if a valid packet was received on PAN1 (PAN1's FILTERFAIL_FLAG is deasserted). RECD_ON_PAN1 remains valid until the start of the next autoseqeuence. |
| 22<br>RECD_ON_PAN0 | Last Packet was Received on PAN0<br>Indicates the packet which was just received, was received on PAN0. In Dual PAN mode operating on 2 different channels, RECD_ON_PAN0 will be set if CURRENT_NETWORK=0 when the packet was |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | received, regardless of FILTERFAIL status. In DUAL PAN mode operating with same channel on both networks, CURRENT_NETWORK will be ignored and RECD_ON_PAN0 will be set only if a valid packet was received on PAN0 (PAN0's FILTERFAIL_FLAG is deasserted). RECD_ON_PAN0 remains valid until the start of the next autoseqeuence. |
| 21-16<br>DUAL_PAN_REMAIN | Time Remaining before next PAN switch in auto Dual PAN mode<br>This read-only register indicates time remaining before next PAN switch in auto Dual PAN mode. The units for this register, depend on the PRESCALER setting (bits [1:0]) in the DUAL_PAN_DWELL register. Tthe units are the same as the TIMEBASE determined by the prescaler, as per the following table:<br>DUAL_PAN_DWELL[1:0]=0b00 (TIMEBASE=0.5ms) DUAL_PAN_REMAIN UNITS=0.5ms<br>DUAL_PAN_DWELL[1:0]=0b01 (TIMEBASE=2.5ms) DUAL_PAN_REMAIN UNITS=2.5ms<br>DUAL_PAN_DWELL[1:0]=0b10 (TIMEBASE=10ms) DUAL_PAN_REMAIN UNITS=10ms<br>DUAL_PAN_DWELL[1:0]=0b11 (TIMEBASE=50ms) DUAL_PAN_REMAIN UNITS=50ms<br>The readback value indicates that between N-1 and N timebase units remain until the next PAN switch. For example, a DUAL_PAN_REMAIN readback value of 3, with a DUAL_PAN_DWELL PRESCALER setting of 2 (10ms), indicates that between 20ms (2*10ms) and 30ms (3*10ms), remain until the next automatic PAN switch. |
| 15-8<br>DUAL_PAN_DWELL | Dual PAN Channel Frequency Dwell Time<br>Channel Frequency Dwell Time. In Auto Dual PAN mode, hardware will toggle the PAN, after dwelling on the current PAN for the interval described below (assuming Preamble/SFD not detected). A write to DUAL_PAN_DWELL, always re-initilizes the DWELL TIMER to the programmed value. If a write to DUAL_PAN_DWELL occurs during an autosequence, the DWELL TIMER will begin counting down immediately. If a write to DUAL_PAN_DWELL occurs when there is no autosequence underway, the DWELL TIMER will not begin counting until the next autosequence begins; it will begin counting at the start of the sequence warmup.<br>DUAL_PAN_DWELL[1:0] select the timebase for the dwell time. There are 4 options:<br>DUAL_PAN_DWELL[1:0]=0b00 --> TIMEBASE=0.5ms<br>DUAL_PAN_DWELL[1:0]=0b01 --> TIMEBASE=2.5ms<br>DUAL_PAN_DWELL[1:0]=0b10 --> TIMEBASE=10ms<br>DUAL_PAN_DWELL[1:0]=0b11 --> TIMEBASE=50ms<br>DUAL_PAN_DWELL[7:2] select how many multiples of the time base to use, from 1 (DUAL_PAN_DWELL[7:2]=0) to 64 (DUAL_PAN_DWELL[7:2]=63). Given those definitions, the table below specifies the minimum and maximum dwell times for each TIMEBASE option:<br>DUAL_PAN_DWELL[1:0]=0b00 --> MIN_DWELL=0.5ms MAX_DWELL=32ms<br>DUAL_PAN_DWELL[1:0]=0b01 --> MIN_DWELL=2.5ms MAX_DWELL=160ms<br>DUAL_PAN_DWELL[1:0]=0b10 --> MIN_DWELL=10ms MAX_DWELL=640ms<br>DUAL_PAN_DWELL[1:0]=0b11 --> MIN_DWELL=50ms MAX_DWELL=3.2seconds<br>A write to DUAL_PAN_DWELL also causes the value of ACTIVE_NETWORK to get latched into the hardware. This latched value will be the starting point for the automatic dual-pan mode (i.e., start on PAN0 or on PAN1). The starting value takes effect immediately (if sequence is underway and DUAL_PAN_AUTO=1), or is otherwise delayed until sequence starts and DUAL_PAN_AUTO=1. |
| 7-6<br>— | Reserved. |
| 5<br>ZB_DP_CHAN_OVRD_SEL | Dual PAN Channel Override Selector<br>This bit works with **ZB_DP_CHAN_OVRD_EN** to allow one of the two Dual PAN channels to use Direct Frequency programming. See description for **ZB_DP_CHAN_OVRD_EN**. |
| 4 | Dual PAN Channel Override Enable |

*Table continues on the next page...*

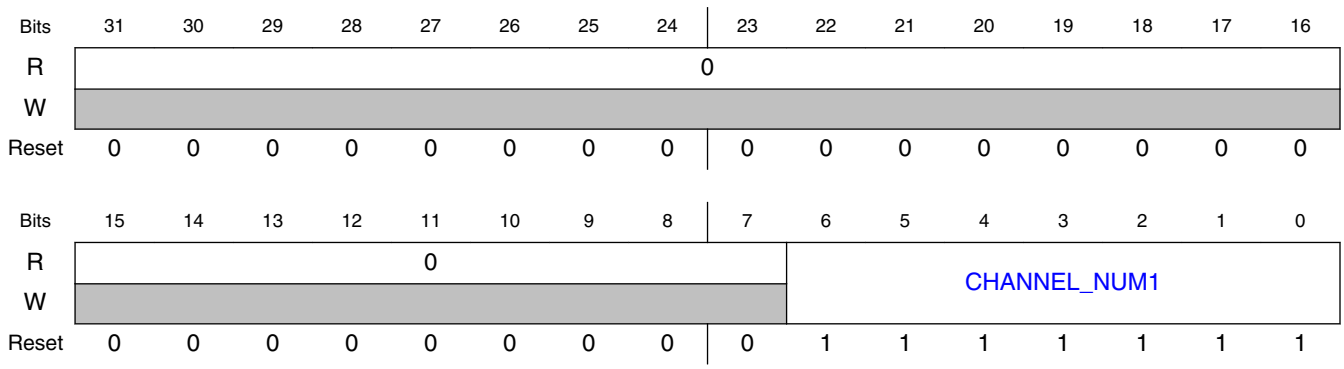
| Field | Function |
|---|---|
| ZB_DP_CHAN_OVRD_EN | In Dual PAN mode, in case there is a need to generate a frequency which may be offset from the 16 prescribed 5MHz-spaced channels, to, for example, avoid interference on one of the Dual PAN channels, a method has been provided to accomplish that, by designating one of the two PAN channels to use the transceiver's set of direct frequency-programming registers, instead of CHANNEL_NUMx. Programming the direct frequency-programming registers -- integer, numerator, and denominator, allows an RF frequency to be selected with much more precision than the 5MHz granularity of the Zigbee mapped-channel registers, CHANNEL_NUM0 and CHANNEL_NUM1.<br><br>Two bits have been provided in 802.15.4 space to realize this feature: ZB_DP_CHAN_OVRD_SEL and ZB_DP_CHAN_OVRD_EN. When ZB_DP_CHAN_OVRD_EN=1, this enables one of the Dual PAN channels to use the direct frequency programming. The ZB_DP_CHAN_OVRD_SEL bit determines which channel uses the direct programming, according to the following:<br><br>If ZB_DP_CHAN_OVRD_EN=0, then PAN0 frequency is determined by register CHANNEL_NUM0[6:0] and PAN1 frequency is determined by register CHANNEL_NUM1[6:0].<br><br>If ZB_DP_CHAN_OVRD_EN=1 and ZB_DP_CHAN_OVRD_SEL=0, then PAN0 frequency is determined by DIRECT FREQUENCY PROGRAMMING and PAN1 frequency is determined by register CHANNEL_NUM1[6:0].<br><br>If ZB_DP_CHAN_OVRD_EN=1 and ZB_DP_CHAN_OVRD_SEL=1, then PAN0 frequency is determined by register CHANNEL_NUM0[6:0] and PAN1 frequency is determined by DIRECT FREQUENCY PROGRAMMING.<br><br>**NOTE:** Direct Frequency Programming is accomplished by setting the PLL's Integer, Numerator, and Denominator registers to the appropriate values for the desired RF frequency. |
| 3<br><br>CURRENT_NETWORK | Indicates which PAN is currently selected by hardware<br><br>This read-only bit indicates which PAN is currently selected by hardware in automatic Dual PAN mode<br>0b - PAN0 is selected<br>1b - PAN1 is selected |
| 2<br><br>PANCORDNTR1 | Device is a PAN Coordinator on PAN1<br><br>Device is a PAN Coordinator on PAN1. Allows device to receive packets with no destination address, if Source PAN ID matches. |
| 1<br><br>DUAL_PAN_AUTO | Activates automatic Dual PAN operating mode<br><br>Activates automatic Dual PAN operating mode. In this mode, PAN-switching is controlled by hardware at a pre-programmed rate, determined by DUAL_PAN_DWELL.<br>0: Manual Dual PAN mode (or Single PAN mode).<br>1: Auto Dual PAN Mode<br><br>Whenever DUAL_PAN_AUTO=0, CURRENT_NETWORK=ACTIVE_NETWORK at all times. In other words, software directly controls which PAN is selected. Whenever DUAL_PAN_AUTO=1, CURRENT_NETWORK is controlled by hardware. |
| 0<br><br>ACTIVE_NETWORK | Active Network Selector<br><br>Selects the PAN on which to transceive, by activating a PAN parameter set (PAN0 or PAN1). In Manual Dual PAN mode (or Single PAN mode), this bit selects the active PAN parameter set (channel and addressing parameters) which governs all autosequences. In Auto Dual PAN mode, this bit selects the PAN on which to begin transceiving, latched at the point at which DUAL_PAN_DWELL register is written.<br>0b - Select PAN0<br>1b - Select PAN1 |

## A.3.2.26 CHANNEL NUMBER 1 (CHANNEL_NUM1)

### A.3.2.26.1 Offset

| Register | Offset |
|---|---|
| CHANNEL_NUM1 | 64h |

### A.3.2.26.2 Diagram




### A.3.2.26.3 Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>CHANNEL_NUM1 | Channel Number for PAN1<br>This is the mapped channel number used to transmit and receive Zigbee packets. This register applies to PAN1 only. CHANNEL_NUM1 should be in the range:<br>11 <= CHANNEL_NUM1 <= 26<br><br>**Note**: This register should not be programmed, and left in its default state, if Dual PAN mode is not in use. |

## A.3.2.27 SAM CONTROL (SAM_CTRL)

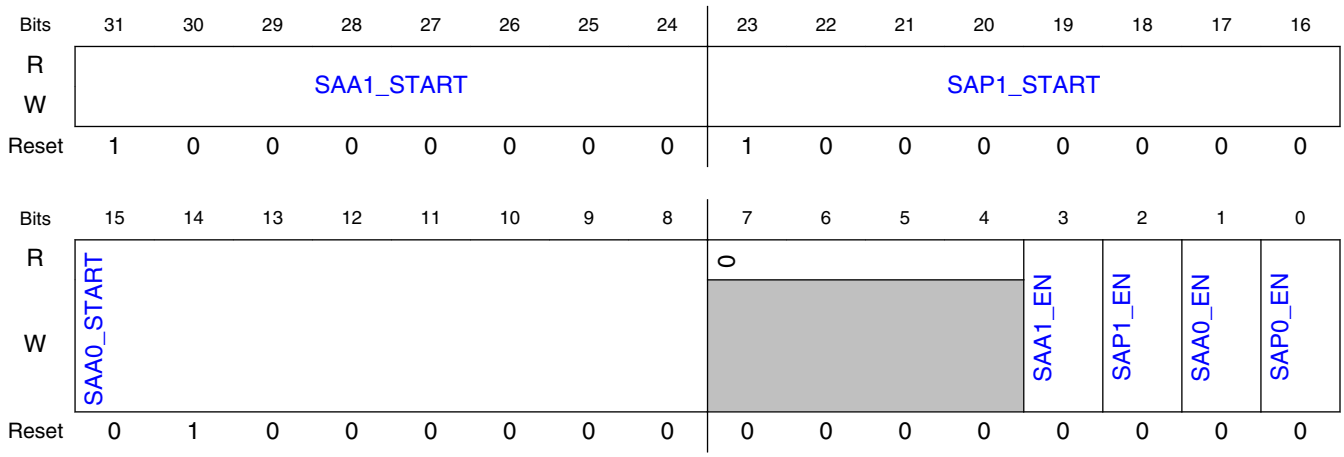
### A.3.2.27.1 Offset

| Register | Offset |
|---|---|
| SAM_CTRL | 68h |

## A.3.2.27.2 Function

Source Address Management Control Register

## A.3.2.27.3 Diagram




## A.3.2.27.4 Fields

| Field | Function |
|---|---|
| 31-24<br>SAA1_START | First Index of SAA1 partition |
| 23-16<br>SAP1_START | First Index of SAP1 partition |
| 15-8<br>SAA0_START | First Index of SAA0 partition |
| 7-4<br>— | Reserved. |
| 3<br>SAA1_EN | Enables SAA1 Partition of the SAM Table<br>0b - Disables SAA1 Partition<br>1b - Enables SAA1 Partition |
| 2<br>SAP1_EN | Enables SAP1 Partition of the SAM Table<br>0b - Disables SAP1 Partition<br>1b - Enables SAP1 Partition |
| 1<br>SAA0_EN | Enables SAA0 Partition of the SAM Table<br>0b - Disables SAA0 Partition<br>1b - Enables SAA0 Partition |
| 0<br>SAP0_EN | Enables SAP0 Partition of the SAM Table<br>0b - Disables SAP0 Partition<br>1b - Enables SAP0 Partition |

# A.3.2.28 SOURCE ADDRESS MANAGEMENT TABLE (SAM_TABLE)

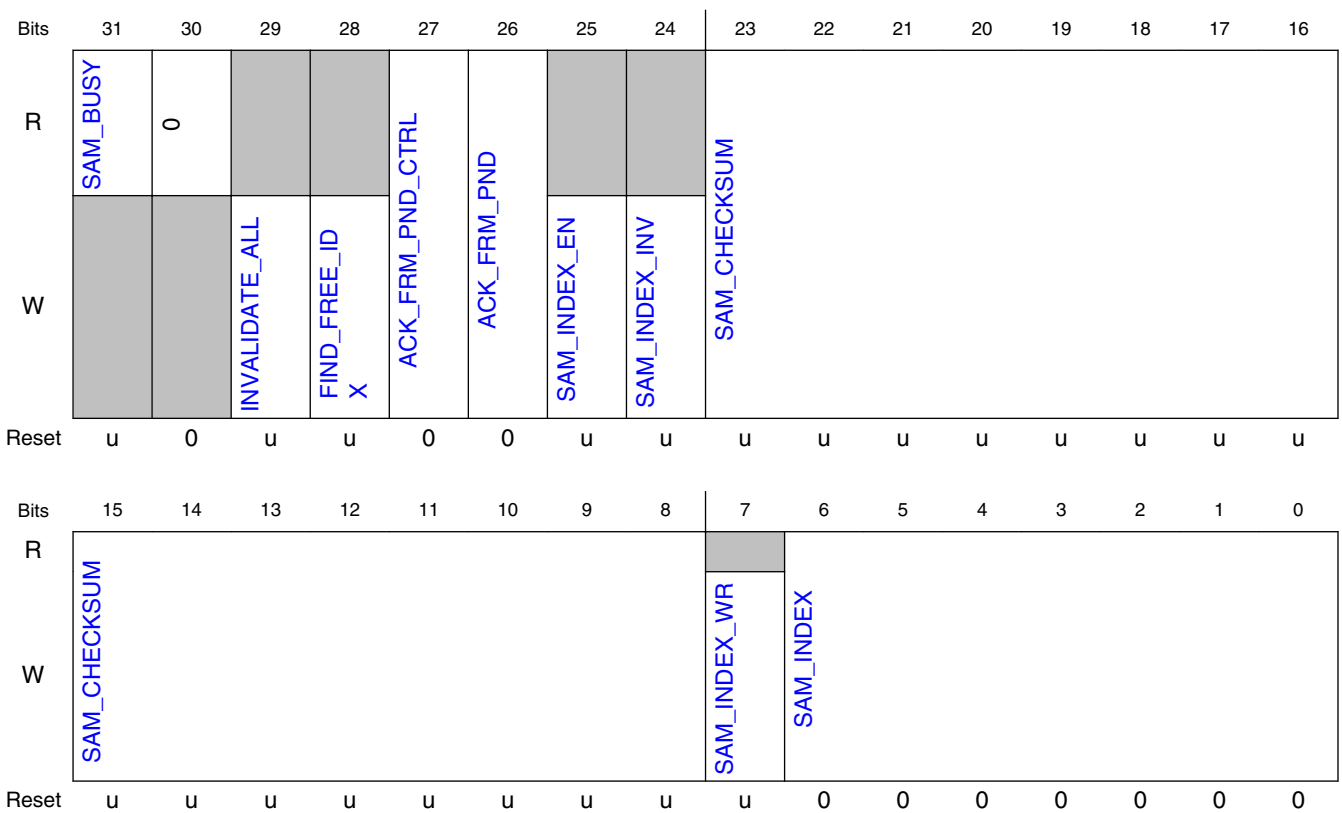
## A.3.2.28.1 Offset

| Register | Offset |
|---|---|
| SAM_TABLE | 6Ch |

## A.3.2.28.2 Function

Source Address Management Table

## A.3.2.28.3 Diagram




## A.3.2.28.4 Fields

| Field | Function |
|---|---|
| 31<br>SAM_BUSY | SAM Table Update Status Bit |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Hardware is in the process of updating the Source Address table, either in response to a poll indication from the packet processor, or due to software setting FIND_FREE_IDX=1. In the latter case, software should poll SAM_BUSY until low before accessing the "First Free Index" registers. Read-only bit. |
| 30<br>— | Reserved. |
| 29<br>INVALIDATE_ALL | Invalidate Entire SAM Table<br>Writing a 1 to this bit clears all 128 Valid bits. Invalidates the entire table. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| 28<br>FIND_FREE_IDX | Find First Free Index<br>After modifying Valid bits (enabling or invalidating), write this bit to 1 to force hardware to update the "First Free Index" registers to account for the changed Valid bits. This hardware update process takes 4us. Software can poll SAM_BUSY to determine when the table update is complete. Write-only bit. Writing 0 to this bit has no effect. Readback value is indeterminate. |
| 27<br>ACK_FRM_PND_CTRL | Manual Control for AutoTxAck FramePending field<br>0b - the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet is determined by hardware<br>1b - the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet tracks ACK_FRM_PEND |
| 26<br>ACK_FRM_PND | State of AutoTxAck FramePending field when SAM Accelleration is Disabled<br>Software can take manual control of the FramePending field of the Frame Control Field of the next automatic TX acknowledge packet, by setting ACK_FRM_PND_CTRL=1; in that case FramePending will track the state of this bit. The FramePending field also tracks this bit if Source Address Management is completely disabled, i.e.,<br>SAP0_EN=SAA0_EN=SAP1_EN=SAA1_EN=0<br>Otherwise, the FramePending field is determined by Source Address Management (SAM) hardware. |
| 25<br>SAM_INDEX_EN | Enable the SAM table index selected by SAM_INDEX |
| 24<br>SAM_INDEX_INV | Invalidate the SAM table index selected by SAM_INDEX |
| 23-8<br>SAM_CHECKSUM | Software-computed source address checksum, to be installed into a table index<br>Software-computed source address checksum, to be installed into a table index. The value on SAM_CHECKSUM[15:0] can be installed into the table with a single, atomic 32-bit write; in that case, the write data would contain the desired SAM_INDEX[6:0] and SAM_CHECKSUM[15:0], and SAM_INDEX_WR=1.<br>If SAM_INDEX_WR=0, then the SAM_INDEX[6:0] register is written, but the checksum is *not* written to the table.<br>The readback value of SAM_CHECKSUM[15:0] is the contents of the SAM Table at the location pointed to by SAM_INDEX[6:0]. To readback from a specific table index, software should first write the desired index to SAM_INDEX[6:0], and then read back the checksum from the table on SAM_CHECKSUM[15:0]. |
| 7<br>SAM_INDEX_WR | Enables SAM Table Contents to be updated<br>For 32-bit writes, SAM_INDEX_WR must be set to indicate that the table entry specified by SAM_INDEX[6:0] is to be written; if SAM_INDEX_WR=0, the table entry is not written, but the SAM_INDEX[6:0] register is updated. For 8-bit writes, this bit is ignored. |
| 6-0<br>SAM_INDEX | Contains the SAM table index to be enabled or invalidated |

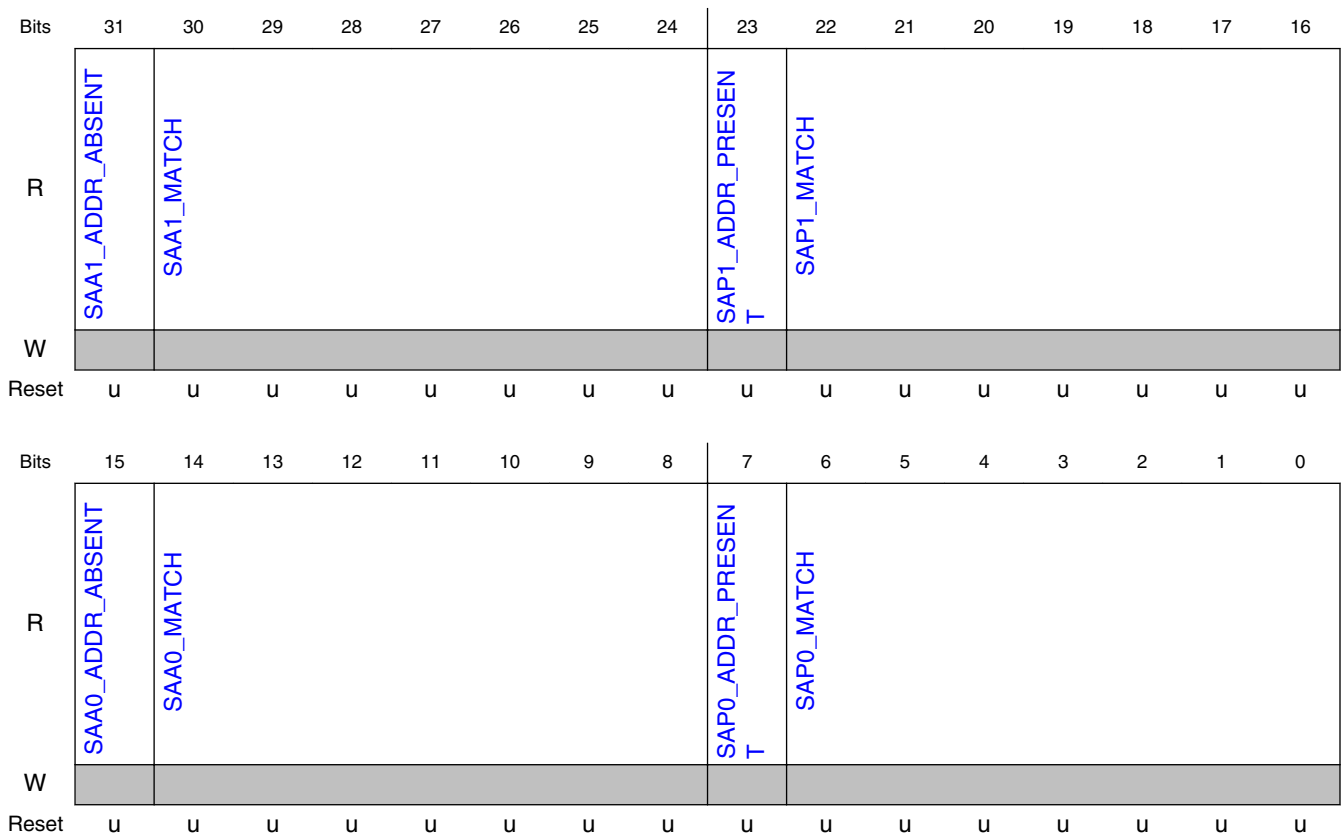
| Field | Function |
|---|---|
| | Contains the table index to be enabled or invalidated. Software must ensure that the index is within the range of the desired partition. |

## A.3.2.29 SOURCE ADDRESS MANAGEMENT MATCH (SAM_MATCH)

### A.3.2.29.1 Offset

| Register | Offset |
|---|---|
| SAM_MATCH | 70h |

### A.3.2.29.2 Diagram



### A.3.2.29.3 Fields

| Field | Function |
|---|---|
| 31<br>SAA1_ADDR_ABSENT | A Checksum Match is Absent in the SAP1 Partition of the SAM Table |
| 30-24<br>SAA1_MATCH | Index in the SAA1 Partition of the SAM Table corresponding to the first checksum match |
| 23<br>SAP1_ADDR_PRESENT | A Checksum Match is Present in the SAP1 Partition of the SAM Table |
| 22-16<br>SAP1_MATCH | Index in the SAP1 Partition of the SAM Table corresponding to the first checksum match |
| 15<br>SAA0_ADDR_ABSENT | A Checksum Match is Absent in the SAA0 Partition of the SAM Table |
| 14-8<br>SAA0_MATCH | Index in the SAA0 Partition of the SAM Table corresponding to the first checksum match |
| 7<br>SAP0_ADDR_PRESENT | A Checksum Match is Present in the SAP0 Partition of the SAM Table |
| 6-0<br>SAP0_MATCH | Index in the SAP0 Partition of the SAM Table corresponding to the first checksum match |

## A.3.2.30 SAM FREE INDEX (SAM_FREE_IDX)

### A.3.2.30.1 Offset

| Register | Offset |
|---|---|
| SAM_FREE_IDX | 74h |

## A.3.2.30.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SAA1_1ST_FREE_IDX | | | | | | | | SAP1_1ST_FREE_IDX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SAA0_1ST_FREE_IDX | | | | | | | | SAP0_1ST_FREE_IDX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

## A.3.2.30.3 Fields

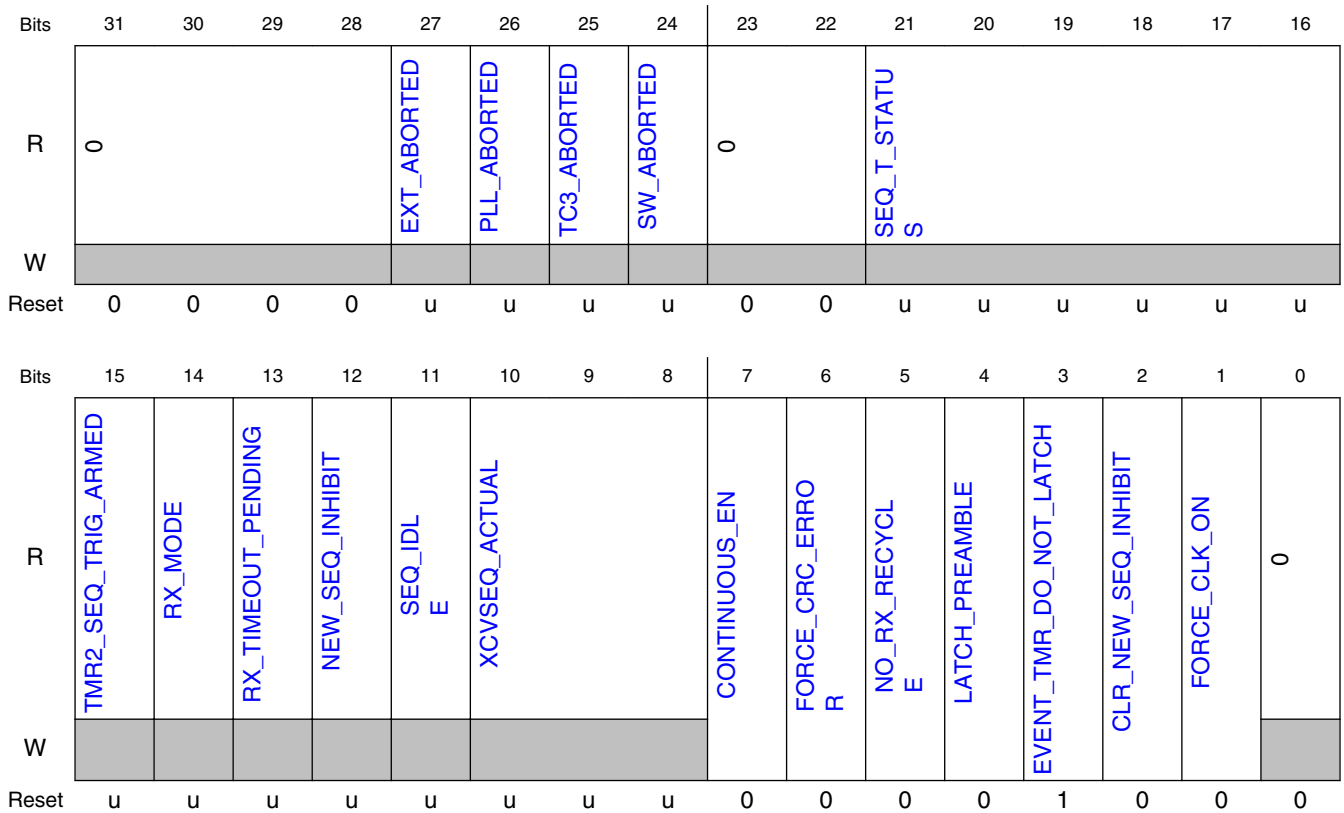
| Field | Function |
|---|---|
| 31-24<br>SAA1_1ST_FREE_IDX | First non-enabled (invalid) index in the SAA1 partition |
| 23-16<br>SAP1_1ST_FREE_IDX | First non-enabled (invalid) index in the SAP1 partition |
| 15-8<br>SAA0_1ST_FREE_IDX | First non-enabled (invalid) index in the SAA0 partition |
| 7-0<br>SAP0_1ST_FREE_IDX | First non-enabled (invalid) index in the SAP0 partition |

# A.3.2.31 SEQUENCE CONTROL AND STATUS (SEQ_CTRL_STS)

## A.3.2.31.1 Offset

| Register | Offset |
|---|---|
| SEQ_CTRL_STS | 78h |

## A.3.2.31.2 Diagram




## A.3.2.31.3 Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>EXT_ABORTED | Autosequence has terminated due to a Wake-On-Radio command<br>when asserted, indicates that the autosequence has terminated due to an external event, such as a Wake-On-Radio stop command. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 26<br>PLL_ABORTED | Autosequence has terminated due to an PLL unlock event<br>when asserted, indicates that the autosequence has terminated due to an PLL unlock event. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 25<br>TC3_ABORTED | autosequence has terminated due to an TMR3 timeout<br>when asserted, indicates that the autosequence has terminated due to an TC3 (TMR3) timeout during a receive operation. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 24<br>SW_ABORTED | Autosequence has terminated due to a Software abort. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | when asserted, indicates that the autosequence has terminated due to an Software abort. Software can abort any programmed autosequence by writing Sequence I to XCVSEQ. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. Read-only bit. |
| 23-22<br>—— | Reserved. |
| 21-16<br>SEQ_T_STATUS | Status of the just-completed or ongoing Sequence T or Sequence TR<br>Status of the just-completed (or ongoing) Sequence T or Sequence TR autosequence. This register is valid at all times during, and after, the Sequence T or Sequence TR. Not valid for other types of autosequences. This is a read-only register. The bits of this register map to status, according to the following table:<br>[0] 1st CCA complete (CCABFRTX=1)<br>[1] 2nd CCA complete (SLOTTED=1)<br>[2] Tx operation complete<br>[3] Rx Recycle occurred (Sequence TR only)<br>[4] Rx operation complete (Sequence TR only)<br>[5] TxAck operation complete(Sequence TR only) |
| 15<br>TMR2_SEQ_TRIG_ARMED | indicates that TMR2 has been programmed and is armed to trigger a new autosequence<br>when asserted, indicates that TMR2 has been programmed and is "armed" to trigger a new autosequence, when Zigbee Sequence Manager timer-triggering mode is selected (i.e., TMRTRIGEN=1). When timer-triggering mode is selected, TMR2 must be re-programmed (using either T2CMP or T2PRIMECMP), in advance of each new sequence. Once TMR2 is programmed, this bit will be asserted, and will remain asserted until the new sequence commences (at TMR2 match). Hardware will deassert this bit when the new sequence starts. When TMRTRIGEN=0, this bit should be ignored. Read-only bit. |
| 14<br>RX_MODE | RX Operation in Progress<br>when asserted, this Sequence Manager Output indicates that an RX operation is in progress. An RX operation can be part of a complex transmit autosequence such as a Sequence TR. CCA and ED operations are considered RX operations, during which rx_mode is asserted. Read-only bit. |
| 13<br>RX_TIMEOUT_PENDING | Indicates a TMR3 RX Timeout is Pending<br>when asserted, indicates that a TMR3 timeout (RX timeout) flag has been set by Hardware, but the Sequence Manager has not yet aborted because an RX operation is not currently underway. This would be the case, for example, during a Sequence TR, if a TMR3 timeout were to occur during the transmit operation of this sequence; the sequence would not be aborted by Hardware until the receive operation begins. This bit will always be 0 if TC3TMOUT=0. Read-only bit. |
| 12<br>NEW_SEQ_INHIBIT | New Sequence Inhibit<br>When asserted, indicates that a new programmed autosequence has commenced (TMR2 match has occurred if TMRTRIGEN=1). Once this bit is asserted, software is blocked from commanding any "new" autosequences (other than Sequence I to abort the current sequence), until the current sequence completes. Hardware will ignore a sequence-change command from software while this bit is asserted. Hardware will automatically deassert this bit once the sequence completes. Read-only bit. |
| 11<br>SEQ_IDLE | ZSM Sequence Idle Indicator |
| 10-8<br>XCVSEQ_ACTUAL | Indicates the programmed sequence that has been recognized by the ZSM Sequence Manager<br>Reflects the programmed sequence that has been recognized by the Zigbee Sequence Manager. Takes into account the fact that sequence-change commands from software are ignored while a sequence is underway (see NEW_SEQ_INHIBIT). Read-only bits. |
| 7 | Enable Continuous TX or RX Mode |

*Table continues on the next page...*

| Field | Function |
|---|---|
| CONTINUOUS_EN | Continuous Mode Enable (Continuous TX or RX).<br>**Note:** Dual PAN mode should not be engaged in Continuous TX or RX modes.<br>0b - normal operation<br>1b - Continuous TX or RX mode is enabled (depending on XCVSEQ setting). |
| 6<br>FORCE_CRC_ERROR | Induce a CRC Error in Transmitted Packets<br>0b - normal operation<br>1b - Force the next transmitted packet to have a CRC error |
| 5<br>NO_RX_RECYCLE | Disable Automatic RX Sequence Recycling<br>when asserted, prevents the Zigbee Sequence Manager (ZSM) from automatically re-starting (recycling) the receiver when a packet is received which results in a FilterFail or CRC failure. Normally, on a RX recycle, the ZSM returns to the RX_WU (warmup) state, and then resumes from there with a new, foreshortened, Rx warmup, in search of a new preamble. When this bit is set, the Sequence Manager will instead return to idle state, and issue a SEQIRQ, after a FilterFail or CRC failure. |
| 4<br>LATCH_PREAMBLE | Stickiness Control for Preamble Detection<br>0b - Don't make PREAMBLE_DET and SFD_DET bits of PHY_STS (SEQ_STATE) Register "sticky", i.e, these status bits reflect the realtime, dynamic state of preamble_detect and sfd_detect<br>1b - Make PREAMBLE_DET and SFD_DET bits of PHY_STS (SEQ_STATE) Register "sticky", i.e.,occurrences of preamble and SFD detection are latched and held until the start of the next autosequence |
| 3<br>EVENT_TMR_DO_NOT_LATCH | Overrides the automatic hardware latching of the Event Timer<br>when asserted, overrides the automatic hardware latching of the Event Timer that prevents the timer from updating while software reads the 3 Event Timer bytes. This allows the Event Timer LS byte to continue to update without reading the upper 2 bytes. Overriding the automatic latching of the Event Timer should be used with caution, as it can allow the Event Timer lower bytes to get out-of-sync with the upper bytes. However, it can be useful when polling the Event Timer LS byte for a value that is just a few counts in the future. |
| 2<br>CLR_NEW_SEQ_INHIBIT | Overrides the automatic hardware locking of the programmed XCVSEQ while an autosequence is underway<br>when asserted, overrides the automatic hardware locking of the programmed XCVSEQ while an autosequence is underway. Asserting this feature will allow software to change the programmed autosequence "on-the-fly", without aborting and returning to idle between sequences. Overriding the hardware lockout of XCVSEQ should be used with caution, since the Sequence Manager is not designed (or verified) for manual state transitions between one type of autosequence and other (i.e., Sequence T -> Sequence R). |
| 1<br>FORCE_CLK_ON | Force On 802.15.4 phy_gck<br>Force On 802.15.4 phy_gck<br>0b - Allow TSM to control 802.15.4 phy_gck, for minimum power consumption (default)<br>1b - Force on 802.15.4 phy_gclk at all times, for debug purposes only |
| 0<br>— | Reserved. |

## A.3.2.32 ACK DELAY (ACKDELAY)

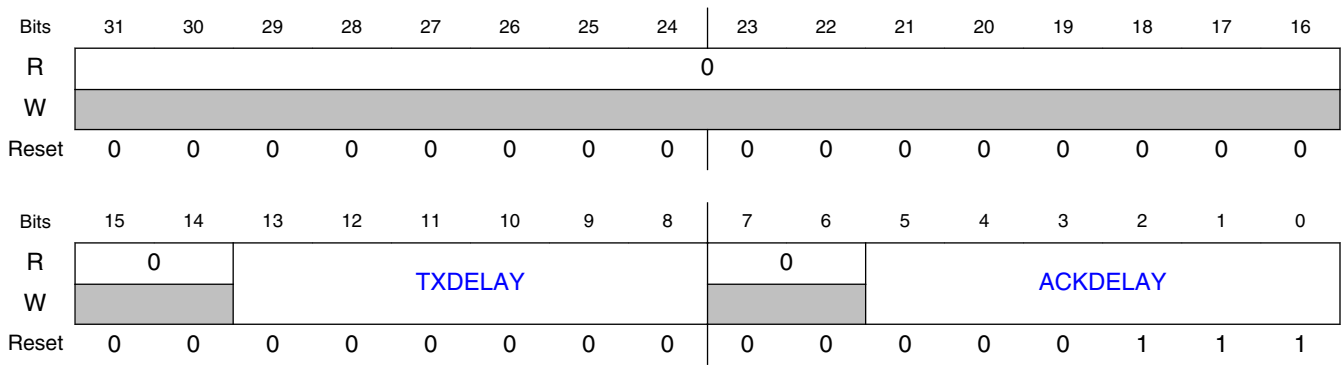
## A.3.2.32.1 Offset

| Register | Offset |
|---|---|
| ACKDELAY | 7Ch |

## A.3.2.32.2 Function

Provides a fine-tune adjustment of the time delay between Rx warmdown and the beginning of Tx warmup for an autoTxAck packet.

## A.3.2.32.3 Diagram




## A.3.2.32.4 Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved. |
| 13-8<br>TXDELAY | TX Delay<br>Provides a fine-tune adjustment of the time delay between post-CCA Rx warm-down and the beginning of Tx warm-up for an Tx (non-Ack) packet. TXDELAY register will apply in both SLOTTED and UNSLOTTED modes, but only to T sequences (e.g., T, TR, and T(R) ), not TxAck operations. This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Max TXDELAY = 0x1F.<br>Min TXDELAY = 0x2D. |
| 7-6<br>— | Reserved. |
| 5-0<br>ACKDELAY | ACK Delay |

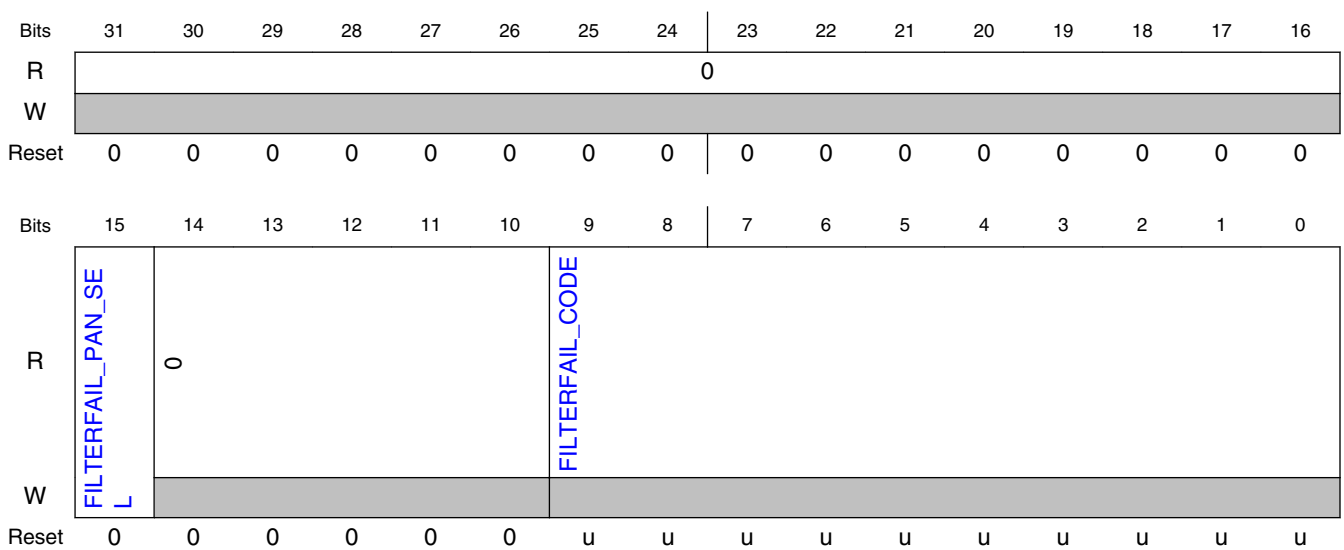
| Field | Function |
|---|---|
| | Provides a fine-tune adjustment of the time delay between Rx warmdown and the beginning of Tx warmup for an Tx Acknowledge packet. ACKDELAY register will apply to both SLOTTED and UNSLOTTED TxAck, but only to TxAck (not T sequences). This is a two's complement value. The minimum permissible value is -19 (0x2D). Values less than -19 will lead to unexpected results.<br>Resolution = 2us.<br>Range = +/- 62us.<br>Max ACKDELAY = 0x1F.<br>Min ACKDELAY = 0x2D. |

## A.3.2.33 FILTER FAIL CODE (FILTERFAIL_CODE)

### A.3.2.33.1 Offset

| Register | Offset |
|---|---|
| FILTERFAIL_CODE | 80h |

### A.3.2.33.2 Diagram




### A.3.2.33.3 Fields

| Field | Function |
|---|---|
| 31-16 | Reserved. |

*Table continues on the next page...*

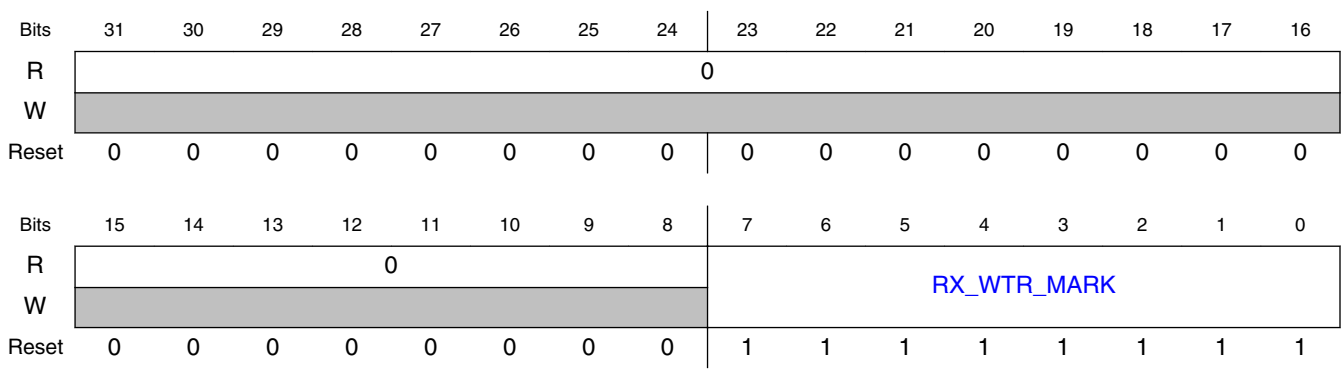
| Field | Function |
|---|---|
| — | |
| 15<br>FILTERFAIL_PAN_SEL | PAN Selector for Filter Fail Code<br>0b - FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN0<br>1b - FILTERFAIL_CODE[9:0] will report the FILTERFAIL status of PAN1 |
| 14-10<br>— | Reserved. |
| 9-0<br>FILTERFAIL_CODE | Filter Fail Code<br>Code inidicating what condition, or conditions, caused the Packet Processor to reject the just-received packet. The bits of FILTERFAIL_CODE indicate the reason for packet rejection. For a description of the individual FILTERFAIL_CODE bits, see the 802.15.4 Link Layer Controller chapter. |

## A.3.2.34 RECEIVE WATER MARK (RX_WTR_MARK)

### A.3.2.34.1 Offset

| Register | Offset |
|---|---|
| RX_WTR_MARK | 84h |

### A.3.2.34.2 Diagram




### A.3.2.34.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0 | RECEIVE WATER MARK |

| Field | Function |
|---|---|
| RX_WTR_MARK | Receive byte count (octets) needed to trigger a RXWTRMRKIRQ interrupt . A setting of 0 generates an interrupt at end of the Frame Length field (first byte after SFD). A setting of 1 generates an interrupt after the first byte of Frame Control Field, etc. |

## A.3.2.35 SLOT PRELOAD (SLOT_PRELOAD)

### A.3.2.35.1 Offset

| Register | Offset |
|---|---|
| SLOT_PRELOAD | 8Ch |

### A.3.2.35.2 Function

Slotted Mode Preload

### A.3.2.35.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SLOT_PRELOAD | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

### A.3.2.35.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>SLOT_PRELOAD | Slotted Mode Preload<br>This register represents the number that gets loaded into the slot_timer at SFD detect, which ultimately determines when the next slot boundary will occur. Due to processing delays within the analog front-end and digital modem, the point at which SFD is detected by the modem, is delayed relative to over-the-air timing. This register setting compensates for that delay. This timing parameter is critical for the Sequence R autosequence in slotted mode, when an automatic TxAck is required. |

# A.3.2.36 802.15.4 SEQUENCE STATE (SEQ_STATE)

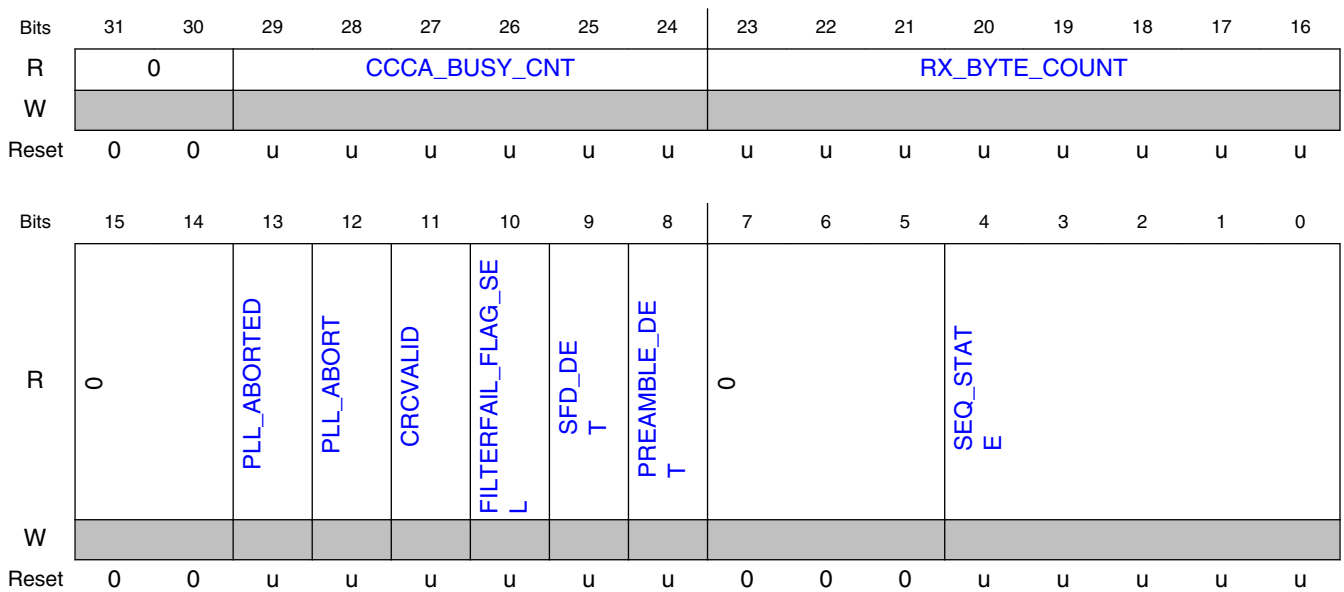
## A.3.2.36.1 Offset

| Register | Offset |
|---|---|
| SEQ_STATE | 90h |

## A.3.2.36.2 Function

802.15.4 Sequence State Register

## A.3.2.36.3 Diagram




## A.3.2.36.4 Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-24<br>CCCA_BUSY_CNT | Number of CCA Measurements resulting in Busy Channel<br>For Sequence CCCA mode only, this register indicates the number of "busy" CCA attempts which occurred during the autosequence, before the channel was detected to be idle. This register can also be |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | read in real-time (during the autosequence) to determine how many busy CCA attempts have occurred to that point. The register saturates at 63 (i.e, if there are more than 63 busy attempts, the register will continue to read 63). This register is automatically cleared to zero by hardware when the next autosequence commences. Read-only register. |
| 23-16<br>RX_BYTE_COUNT | Realtime Received Byte Count<br>During packet reception, this read-only register is a real-time indicator of the number of bytes that have been received. This register will read 0 until SFD and PHR have been received. It will read 1 after the first byte of Frame Control Field has been received, etc. |
| 15-14<br>— | Reserved. |
| 13<br>PLL_ABORTED | Autosequence has terminated due to an PLL unlock event<br>when asserted, indicates that the autosequence has terminated due to an PLL unlock event. This bit is valid at the SEQIRQ interrupt. Hardware will maintain this bit asserted until the next autosequence commences. This bit is a read-only mirror of the register bit of the same name in the ABORT_STS (SEQ_CTRL_STS) register. |
| 12<br>PLL_ABORT | Raw PLL Abort Signal<br>This bit reflects the instantaneous, consolidated status of the PLL unlock detection circuits; if asserted high, indicates that at least one of the three PLL unlock detect mechanisms is currently reporting an unlocked condition. |
| 11<br>CRCVALID | CRC Valid Indicator<br>Cyclic Redundancy Check Valid: This flag indicates the compare result between the FCS field, in the most-recently received frame, and the internally calculated CRC value. This flag is cleared at next receiver warm up.<br>0b - Rx FCS != calculated CRC (incorrect)<br>1b - Rx FCS = calculated CRC (correct) |
| 10<br>FILTERFAIL_FLAG_SEL | Consolidated Filter Fail Flag<br>0: The incoming, or just-received packet, passed packet filtering rules.<br>1: The incoming, or just-received packet, failed packet filtering rules<br>When FILTERFAIL_FLAG_SEL=1, a non-zero FILTERFAIL_CODE is present (see FILTERFAIL_CODE registers).<br>In Dual PAN mode, FILTERFAIL_FLAG_SEL applies to either or both networks, as follows:<br>**A:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=0, FILTERFAIL_FLAG_SEL applies to PAN0.<br>**B:** If PAN0 and PAN1 occupy different channels and CURRENT_NETWORK=1, FILTERFAIL_FLAG_SEL applies to PAN1.<br>**C:** If PAN0 and PAN1 occupy the same channel, FILTERFAIL_FLAG_SEL is the logical 'AND' of the individual PANs' FILTERFAIL_FLAG bits. |
| 9<br>SFD_DET | SFD Detected<br>0: an 802.15.4 preamble-and-SFD have not been detected.<br>1: An 802.15.4 preamble-and-SFD have been detected.<br>The function of this read-only bit depends on the setting of the LATCH_PREAMBLE bit of the SEQ_MGR_CTRL register. If LATCH_PREAMBLE=1, any preamble-and-SFD detection during a Sequence R (even false detections), will set this bit, and it will remain set (sticky) until the start of the next autosequence. If LATCH_PREAMBLE=0, this bit is not sticky, and reflects the instantaneous state of the SFD-detection circuit; for false SFD, the bit will clear when the false nature of the SFD is recognized (i.e., an RX recycle). When LATCH_PREAMBLE=0, SFD_DET should be considered valid only while an autosequence is underway. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 8<br>PREAMBLE_DET | Preamble Detected<br>0: an 802.15.4 preamble has not been detected.<br>1: An 802.15.4 preamble has been detected.<br>The function of this read-only bit depends on the setting of the LATCH_PREAMBLE bit of the SEQ_MGR_CTRL register. If LATCH_PREAMBLE=1, any preamble detection during a Sequence R (even false detections), will set this bit, and it will remain set (sticky) until the start of the next autosequence. If LATCH_PREAMBLE=0, this bit is not sticky, and reflects the instantaneous state of the preamble-detection circuit; for false preambles, the bit will clear when the false nature of the preamble is recognized. When LATCH_PREAMBLE=0, PREAMBLE_DET should be considered valid only while an autosequence is underway. |
| 7-5<br>— | Reserved. |
| 4-0<br>SEQ_STATE | ZSM Sequence State<br>This read-only register reflects the instantaneous state of the 802.15.4 Sequence Manager |

## A.3.2.37 TIMER PRESCALER (TMR_PRESCALE)

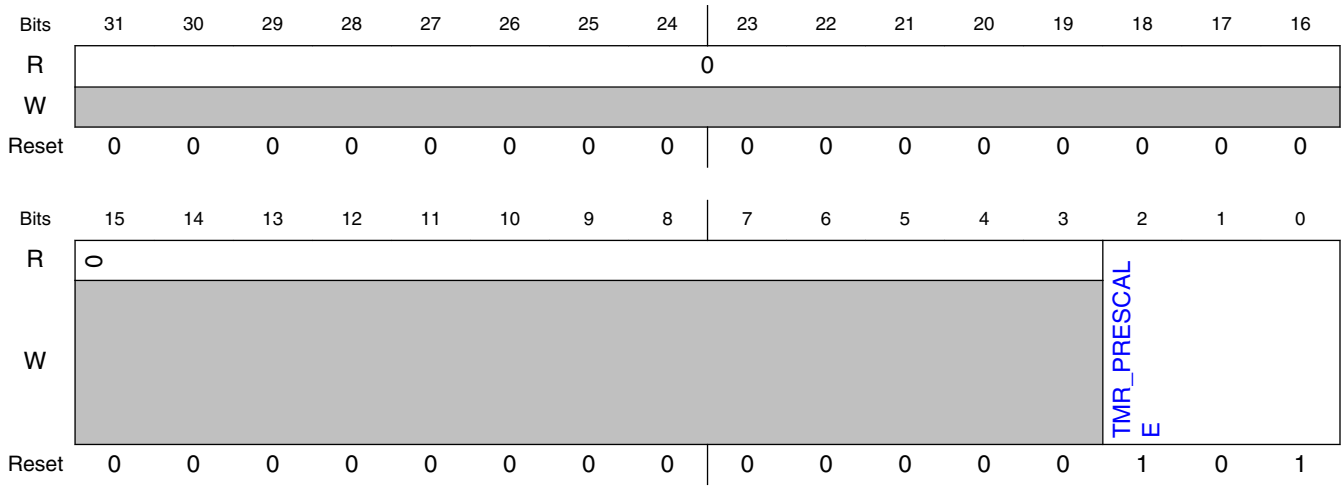
### A.3.2.37.1 Offset

| Register | Offset |
|---|---|
| TMR_PRESCALE | 94h |

### A.3.2.37.2 Function

Timer Prescaler Control Register

### A.3.2.37.3 Diagram




### A.3.2.37.4 Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved. |
| 2-0<br>TMR_PRESCALE | Timer Prescaler<br>Timer Prescaler. Establishes the Event Timer clock rate (and also maximum timer duration)<br>**Note:**To take advantage of the EVENT_TMR Fractional bits for 802.15.4 DSM mode, only the default setting ("5", or 62.5KHz) is allowed.<br>000b - Reserved<br>001b - Reserved<br>010b - 500kHz (33.55 S)<br>011b - 250kHz (67.11 S)<br>100b - 125kHz (134.22 S)<br>101b - 62.5kHz (268.44 S) -- default<br>110b - 31.25kHz (536.87 S)<br>111b - 15.625kHz (1073.74 S) |

## A.3.2.38 LENIENCY LSB (LENIENCY_LSB)

### A.3.2.38.1 Offset

| Register | Offset |
|---|---|
| LENIENCY_LSB | 98h |

### A.3.2.38.2 Function

Packet Processor Leniency Bits (LSB)

### A.3.2.38.3 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LENIENCY_LSB | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LENIENCY_LSB | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.3.2.38.4 Fields

| Field | Function |
|---|---|
| 31-0<br>LENIENCY_LSB | Leniency LSB Register<br>The Packet Processor performs filtering on all received packets, in order to determine whether the packet is intended for the device. The packet filtering is based on rules. In case any of the packet filtering rules need to be overriden, a 40-bit "leniency register" has been provided. When the leniency register is programmed to its default value (0), all hardware packet filtering rules are in effect, and if an incoming packet violates any rule, a "Filter Fail" will occur (packet will be rejected). When a given leniency register bit is asserted, the packet filtering rule assigned to that bit will not be in effect, and if any incoming packet violates that rule (but no other rules), then a "Filter Fail" will not occur, the packet will not be rejected, the packet will be treated as "intended for the device", and software will be notified of the incoming packet. For a desciption of the function of the individual leniency bits, see the 802.15.4 Link Layer Controller chapter. |

## A.3.2.39 LENIENCY MSB (LENIENCY_MSB)

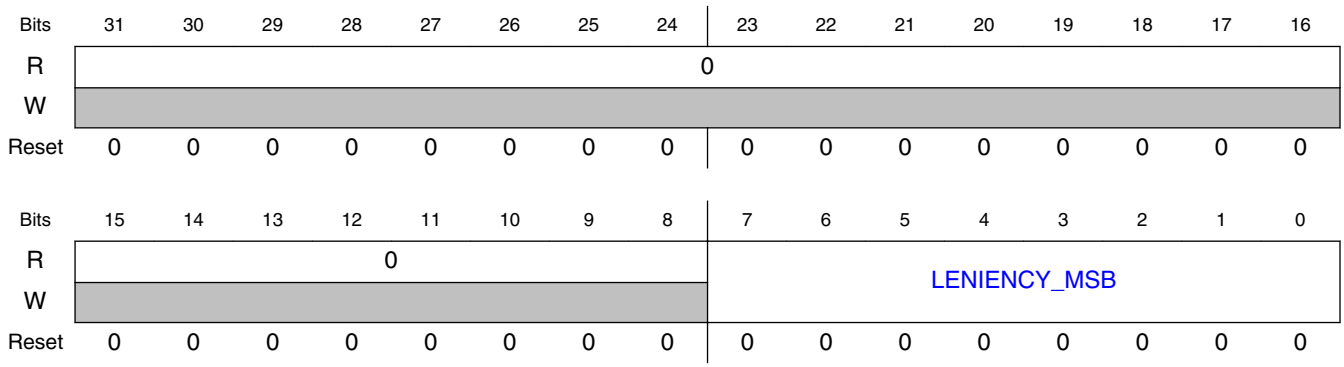
### A.3.2.39.1 Offset

| Register | Offset |
|---|---|
| LENIENCY_MSB | 9Ch |

### A.3.2.39.2 Function

Packet Processor Leniency Bits (MSB)

### A.3.2.39.3 Diagram




### A.3.2.39.4 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>LENIENCY_MSB | Leniency MSB Register<br>The Packet Processor performs filtering on all received packets, in order to determine whether the packet is intended for the device. The packet filtering is based on rules. In case any of the packet filtering rules need to be overriden, a 40-bit "leniency register" has been provided. When the leniency register is programmed to its default value (0), all hardware packet filtering rules are in effect, and if an incoming packet violates any rule, a "Filter Fail" will occur (packet will be rejected). When a given leniency register bit is asserted, the packet filtering rule assigned to that bit will not be in effect, and if any incoming packet violates that rule (but no other rules), then a "Filter Fail" will not occur, the packet will not be rejected, the packet will be treated as "intended for the device", and software will be notified of the incoming packet. For a desciption of the function of the individual leniency bits, see the 802.15.4 Link Layer Controller chapter. |

## A.3.2.40 PART ID (PART_ID)

### A.3.2.40.1 Offset

| Register | Offset |
|---|---|
| PART_ID | A0h |

### A.3.2.40.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PART_ID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### A.3.2.40.3 Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>PART_ID | 802.15.4 Part ID |

## A.3.2.41 Packet Buffer TX (PKT_BUFFER_TX0 - PKT_BUFFER_TX63)

### A.3.2.41.1 Offset

For a = 0 to 63:

| Register | Offset |
|---|---|
| PKT_BUFFER_TXa | 100h + (a × 2h) |

### A.3.2.41.2 Function

Packet Buffer TX

### A.3.2.41.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKT_BUFFER_TX | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### A.3.2.41.4 Fields

| Field | Function |
|---|---|
| 15-0<br>PKT_BUFFER_TX | Packet Buffer Entry<br>Packet Buffer Word |

## A.3.2.42 Packet Buffer RX (PKT_BUFFER_RX0 - PKT_BUFFER_RX63)

### A.3.2.42.1 Offset

For a = 0 to 63:

| Register | Offset |
|---|---|
| PKT_BUFFER_RXa | 180h + (a × 2h) |

### A.3.2.42.2 Function

Packet Buffer RX

### A.3.2.42.3 Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PKT_BUFFER_RX | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### A.3.2.42.4 Fields

| Field | Function |
|---|---|
| 15-0<br>PKT_BUFFER_RX | Packet Buffer Entry<br>Packet Buffer Word |

## A.3.3 GENERIC FSK register descriptions

### A.3.3.1 FSK Memory map

GENFSK base address: 4103_3000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | IRQ CONTROL (IRQ_CTRL) | 32 | RW | See description. |
| 4h | EVENT TIMER (EVENT_TMR) | 32 | RW | See description. |
| 8h | T1 COMPARE (T1_CMP) | 32 | RW | 00FF_FFFFh |
| Ch | T2 COMPARE (T2_CMP) | 32 | RW | 00FF_FFFFh |
| 10h | TIMESTAMP (TIMESTAMP) | 32 | RO | See description. |
| 14h | TRANSCEIVER CONTROL (XCVR_CTRL) | 32 | RW | See description. |
| 18h | TRANSCEIVER STATUS (XCVR_STS) | 32 | RO | See description. |
| 1Ch | TRANSCEIVER CONFIGURATION (XCVR_CFG) | 32 | RW | See description. |
| 20h | CHANNEL NUMBER (CHANNEL_NUM) | 32 | RW | 0000_0000h |
| 24h | TRANSMIT POWER (TX_POWER) | 32 | RW | 0000_0000h |
| 28h | NETWORK ADDRESS CONTROL (NTW_ADR_CTRL) | 32 | RW | See description. |
| 2Ch | NETWORK ADDRESS 0 (NTW_ADR_0) | 32 | RW | 5555_5555h |
| 30h | NETWORK ADDRESS 1 (NTW_ADR_1) | 32 | RW | 5555_5555h |
| 34h | NETWORK ADDRESS 2 (NTW_ADR_2) | 32 | RW | 5555_5555h |
| 38h | NETWORK ADDRESS 3 (NTW_ADR_3) | 32 | RW | 5555_5555h |
| 3Ch | RECEIVE WATERMARK (RX_WATERMARK) | 32 | RW | See description. |
| 40h | DSM CONTROL (DSM_CTRL) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

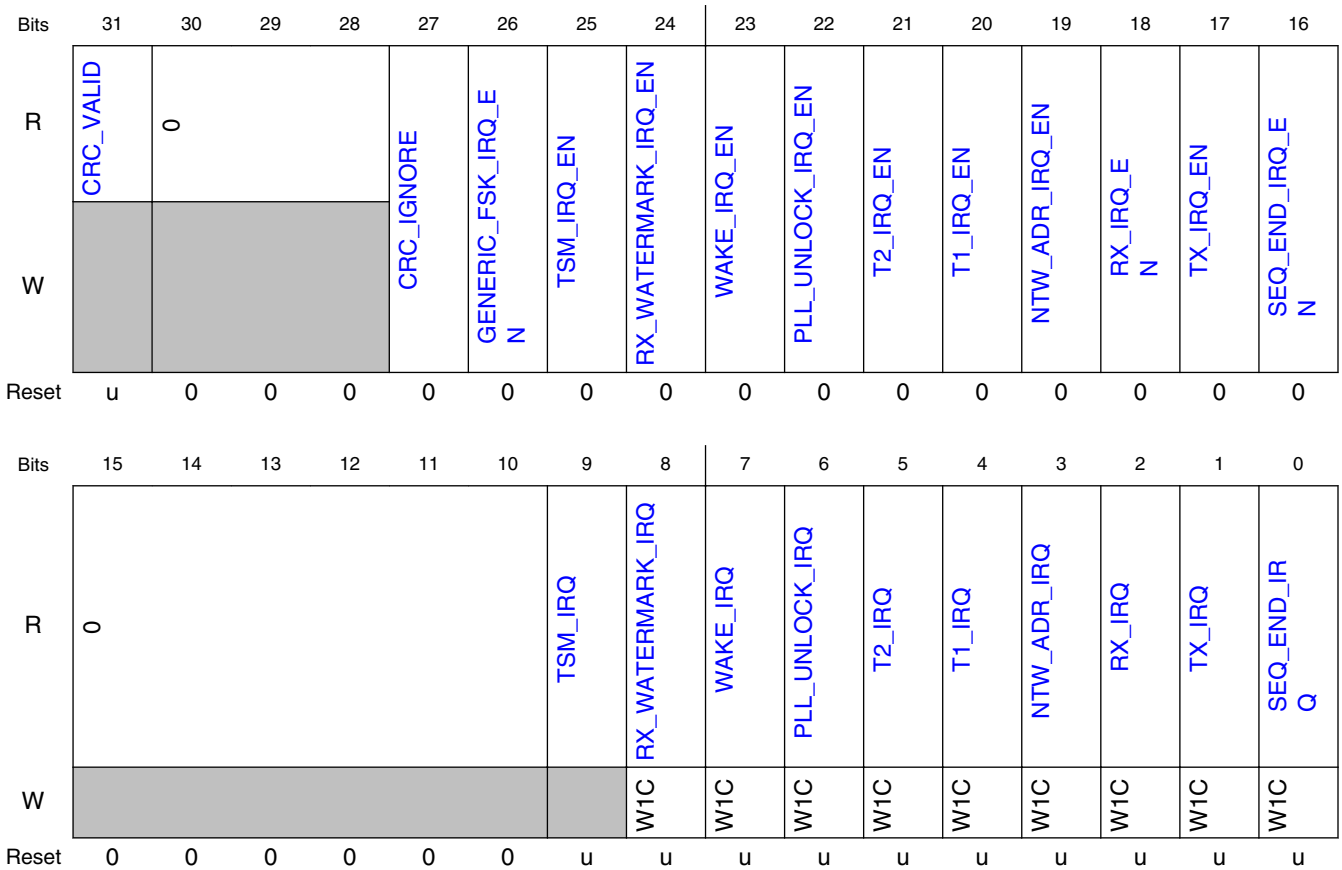
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 44h | PART ID (PART_ID) | 32 | RO | 0000_0001h |
| 60h | PACKET CONFIGURATION (PACKET_CFG) | 32 | RW | See description. |
| 64h | H0 CONFIGURATION (H0_CFG) | 32 | RW | 0000_0000h |
| 68h | H1 CONFIGURATION (H1_CFG) | 32 | RW | 0000_0000h |
| 6Ch | CRC CONFIGURATION (CRC_CFG) | 32 | RW | 0000_0002h |
| 70h | CRC INITIALIZATION (CRC_INIT) | 32 | RW | 0000_0000h |
| 74h | CRC POLYNOMIAL (CRC_POLY) | 32 | RW | 1021_0000h |
| 78h | CRC XOR OUT (CRC_XOR_OUT) | 32 | RW | 0000_0000h |
| 7Ch | WHITENER CONFIGURATION (WHITEN_CFG) | 32 | RW | 01FF_0918h |
| 80h | WHITENER POLYNOMIAL (WHITEN_POLY) | 32 | RW | 0000_0021h |
| 84h | WHITENER SIZE THRESHOLD (WHITEN_SZ_THR) | 32 | RW | 0000_0800h |
| 88h | BIT RATE (BITRATE) | 32 | RW | 0000_0000h |
| 8Ch | PACKET BUFFER PARTITION POINT (PB_PARTITION) | 32 | RW | 0000_0220h |
| 700h - F7Eh | PACKET BUFFER (PACKET_BUFFER_0 - PACKET_BUFFER_1087) | 16 | RW | See description. |

## A.3.3.2 IRQ CONTROL (IRQ_CTRL)

### A.3.3.2.1 Offset

| Register | Offset |
|---|---|
| IRQ_CTRL | 0h |

## A.3.3.2.2 Diagram




## A.3.3.2.3 Fields

| Field | Function |
|---|---|
| 31<br>CRC_VALID | CRC Valid<br>CRC Valid indicator for RX packets. This bit becomes valid at RX_IRQ, and remains valid until the start of the next RX TSM sequence.<br>0b - CRC of RX packet is not valid.<br>1b - CRC of RX packet is valid. |
| 30-28<br>— | Reserved. |
| 27<br>CRC_IGNORE | CRC Ignore<br>If set, assert RX_IRQ even for a received packet which fails CRC verification.<br>0b - RX_IRQ will not be asserted for a received packet which fails CRC verification.<br>1b - RX_IRQ will be asserted even for a received packet which fails CRC verification. |
| 26<br>GENERIC_FSK_IRQ_EN | GENERIC_FSK_IRQ Master Enable<br>Master enable for the GENERIC_FSK_IRQ interrupt line to the MCU.<br>0b - All GENERIC_FSK Interrupts are disabled.<br>1b - All GENERIC_FSK Interrupts can be enabled. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 25<br>TSM_IRQ_EN | TSM_IRQ Enable<br>0b - TSM Interrupt is not enabled.<br>1b - TSM Interrupt is enabled. |
| 24<br>RX_WATERMARK_IRQ_EN | RX_WATERMARK_IRQ Enable<br>0b - RX Watermark Interrupt is not enabled.<br>1b - RX Watermark Interrupt is enabled. |
| 23<br>WAKE_IRQ_EN | WAKE_IRQ Enable<br>0b - Wake Interrupt is not enabled.<br>1b - Wake Interrupt is enabled. |
| 22<br>PLL_UNLOCK_IRQ_EN | PLL_UNLOCK_IRQ Enable<br>0b - PLL Unlock Interrupt is not enabled.<br>1b - PLL Unlock Interrupt is enabled. |
| 21<br>T2_IRQ_EN | T2_IRQ Enable<br>0b - Timer1 (T2) Compare Interrupt is not enabled.<br>1b - Timer1 (T2) Compare Interrupt is enabled. |
| 20<br>T1_IRQ_EN | T1_IRQ Enable<br>0b - Timer1 (T1) Compare Interrupt is not enabled.<br>1b - Timer1 (T1) Compare Interrupt is enabled. |
| 19<br>NTW_ADR_IRQ_EN | NTW_ADR_IRQ Enable<br>0b - Network Address Match Interrupt is not enabled.<br>1b - Network Address Match Interrupt is enabled. |
| 18<br>RX_IRQ_EN | RX_IRQ Enable<br>0b - RX Interrupt is not enabled.<br>1b - RX Interrupt is enabled. |
| 17<br>TX_IRQ_EN | TX_IRQ Enable<br>0b - TX Interrupt is not enabled.<br>1b - TX Interrupt is enabled. |
| 16<br>SEQ_END_IRQ_EN | SEQ_END_IRQ Enable<br>0b - Sequence End Interrupt is not enabled.<br>1b - Sequence End Interrupt is enabled. |
| 15-10<br>— | Reserved. |
| 9<br>TSM_IRQ | TSM Interrupt<br>Indicates TSM0_IRQ or TSM1_IRQ is set in XCVR_STATUS. Clear the bits there. For debug purposes.<br>0b - TSM0_IRQ and TSM1_IRQ are both clear.<br>1b - Indicates TSM0_IRQ or TSM1_IRQ is set in XCVR_STATUS. |
| 8<br>RX_WATERMARK_IRQ | RX Watermark Interrupt<br>Asserts when RX Byte Counter == RX_WATERMARK[12:0]<br>0b - RX Watermark Interrupt is not asserted.<br>1b - RX Watermark Interrupt is asserted. |
| 7<br>WAKE_IRQ | Wake Interrrupt<br>**For Manual DSM:** This is WAKE_IRQ. A WAKE_IRQ will be triggered when the GENERIC_FSK Link Layer Controller has awoken from a Manual DSM (Deep Sleep Mode) cycle. WAKE_IRQ indicates that the RF Oscillator has been restarted, and the GENERIC_FSK EVENT_TMR has resumed counting.<br>**For Wake-On-Radio:** This is WOR_IRQ. See Section Wake-On-Radio<br>0b - Wake Interrupt is not asserted.<br>1b - Wake Interrupt is asserted. |

*Table continues on the next page...*

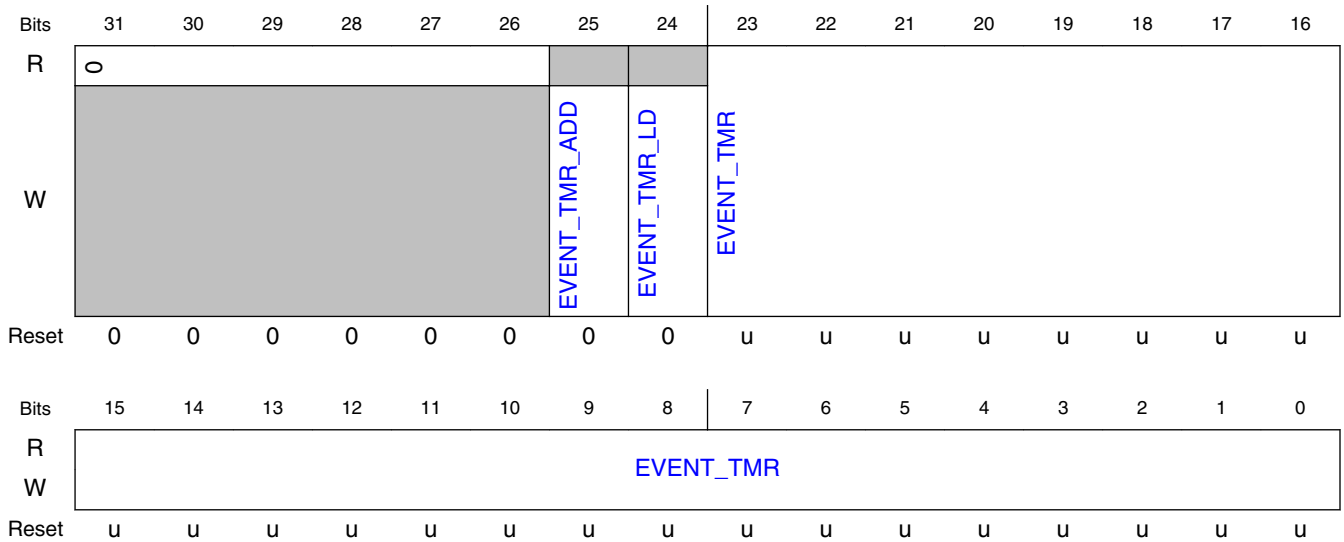
| Field | Function |
|---|---|
| 6<br>PLL_UNLOCK_IRQ | PLL Unlock Interrupt<br>An unlock event has occurred.<br>0b - PLL Unlock Interrupt is not asserted.<br>1b - PLL Unlock Interrupt is asserted. |
| 5<br>T2_IRQ | Timer2 (T2) Compare Interrupt<br>0b - Timer2 (T2) Compare Interrupt is not asserted.<br>1b - Timer2 (T2) Compare Interrupt is asserted. |
| 4<br>T1_IRQ | Timer1 (T1) Compare Interrupt<br>0b - Timer1 (T1) Compare Interrupt is not asserted.<br>1b - Timer1 (T1) Compare Interrupt is asserted. |
| 3<br>NTW_ADR_IRQ | Network Address Match Interrupt<br>A Network Address Match has occurred.<br>0b - Network Address Match Interrupt is not asserted.<br>1b - Network Address Match Interrupt is asserted. |
| 2<br>RX_IRQ | RX Interrupt<br>The RX sequence has completed with a successful packet reception.<br>0b - RX Interrupt is not asserted.<br>1b - RX Interrupt is asserted. |
| 1<br>TX_IRQ | TX Interrupt<br>The TX sequence has completed with a successful packet transmission.<br>0b - TX Interrupt is not asserted.<br>1b - TX Interrupt is asserted. |
| 0<br>SEQ_END_IRQ | Sequence End Interrupt<br>Will assert when any TX or RX sequence ends for any reason.<br>0b - Sequence End Interrupt is not asserted.<br>1b - Sequence End Interrupt is asserted. |

## A.3.3.3 EVENT TIMER (EVENT_TMR)

### A.3.3.3.1 Offset

| Register | Offset |
|---|---|
| EVENT_TMR | 4h |

### A.3.3.3.2 Diagram




### A.3.3.3.3 Fields

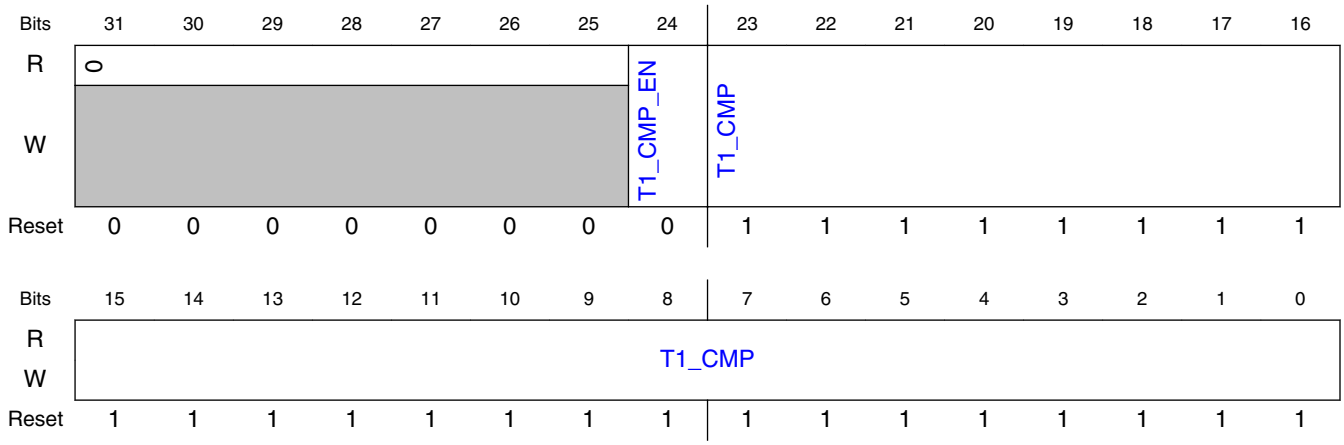
| Field | Function |
|---|---|
| 31-26<br>— | Reserved. |
| 25<br>EVENT_TMR_ADD | Event Timer Add<br>A write access with this bit increments EVENT_TMR by the contents of EVENT_TMR[23:0]. This is a signed addition. |
| 24<br>EVENT_TMR_LD | Event Timer Load<br>A write access with this bit set loads EVENT_TMR with the contents of EVENT_TMR[23:0] |
| 23-0<br>EVENT_TMR | Event Timer<br>Event Timer can be read in these byte locations. To update the Event Timer, either: 1. Write the desired EVENT_TMR to these bytes and set EVENT_TMR_LD=1, or, 2. Write the desired EVENT_TMR increment amount to these bytes and set EVENT_TMR_ADD=1.<br><br>**Note:** for EVENT_TMR_ADD, EVENT_TMR[23:0] is a signed, two's-complement value. |

## A.3.3.4 T1 COMPARE (T1_CMP)

### A.3.3.4.1 Offset

| Register | Offset |
|---|---|
| T1_CMP | 8h |

### A.3.3.4.2 Diagram




### A.3.3.4.3 Fields

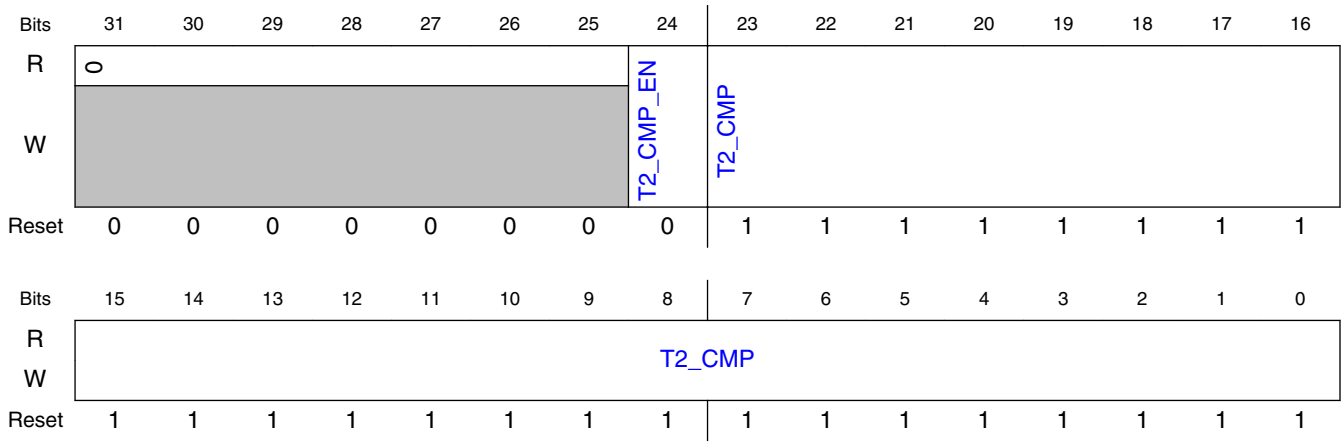
| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>T1_CMP_EN | Timer1 (T1) Compare Enable<br>Enable Timer Compare #1 (T1_CMP) to generate T1_IRQ and/or execute Sequence Commands. |
| 23-0<br>T1_CMP | Timer1 (T1) Compare Value<br>Timer1 (T1) Compare Value. Can be used to generate T1_IRQ and/or launch Sequence Commands |

## A.3.3.5 T2 COMPARE (T2_CMP)

### A.3.3.5.1 Offset

| Register | Offset |
|---|---|
| T2_CMP | Ch |

### A.3.3.5.2 Diagram




### A.3.3.5.3 Fields

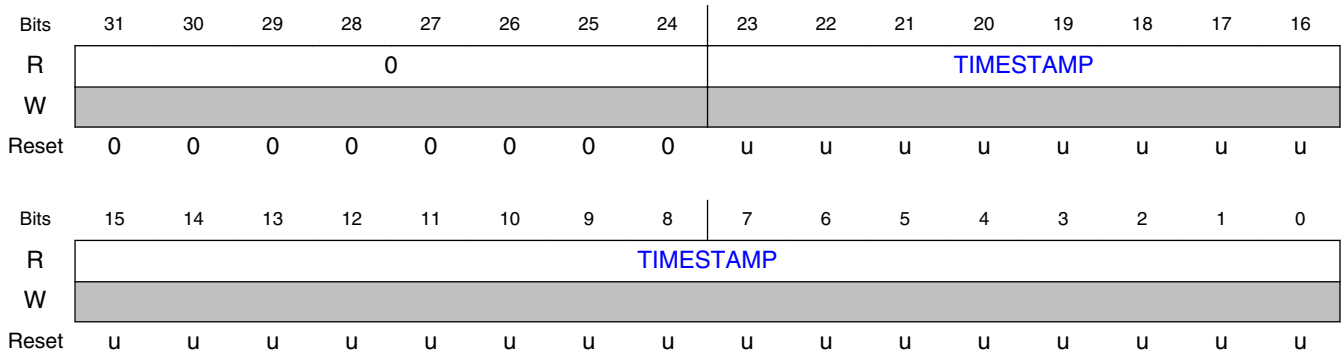
| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>T2_CMP_EN | Timer2 (T2) Compare Enable<br>Enable Timer Compare #2 (T2_CMP) to generate T2_IRQ and/or execute Sequence Commands. |
| 23-0<br>T2_CMP | Timer2 (T2) Compare Value<br>Timer2 (T2) Compare Value. Can be used to generate T2_IRQ and/or launch Sequence Commands |

## A.3.3.6 TIMESTAMP (TIMESTAMP)

### A.3.3.6.1 Offset

| Register | Offset |
|---|---|
| TIMESTAMP | 10h |

### A.3.3.6.2 Diagram




### A.3.3.6.3 Fields

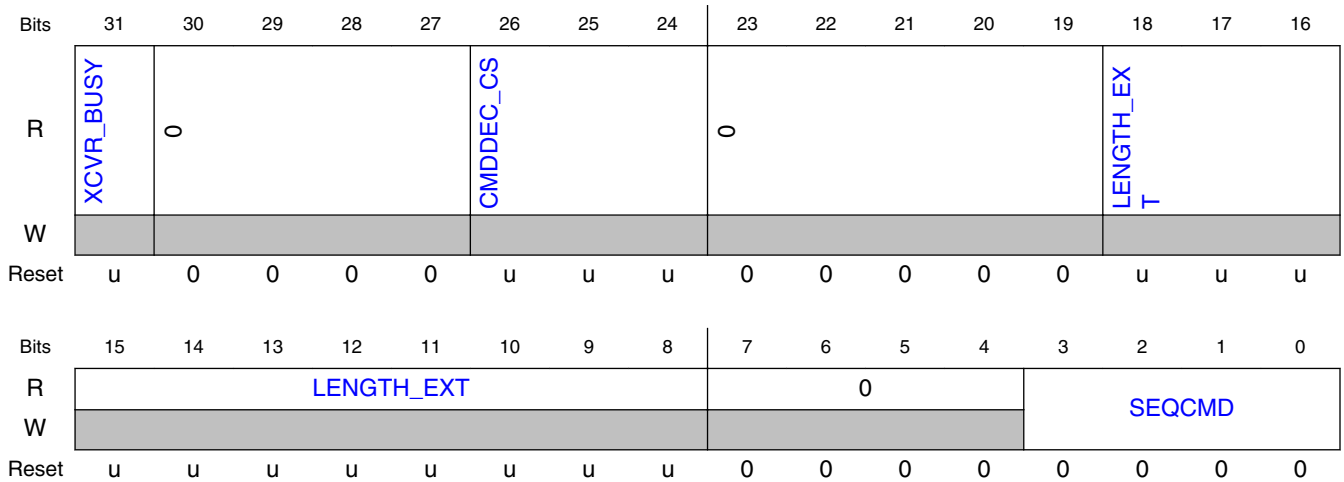
| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>TIMESTAMP | Received Packet Timestamp<br>Received Packet Timestamp. Captured at NTW_ADR_IRQ. |

## A.3.3.7 TRANSCEIVER CONTROL (XCVR_CTRL)

### A.3.3.7.1 Offset

| Register | Offset |
|---|---|
| XCVR_CTRL | 14h |

## A.3.3.7.2 Diagram




## A.3.3.7.3 Fields

| Field | Function |
|---|---|
| 31<br>XCVR_BUSY | Transceiver Busy<br>For multi-protocol arbitration, XCVR_BUSY=1 indicates an RX or TX operation is underway, by either GENERIC_FSK, or some other protocol.<br>0b - IDLE<br>1b - BUSY |
| 30-27<br>— | Reserved. |
| 26-24<br>CMDDEC_CS | Command Decode<br>Current State of the Command Decoder FSM (debug only) |
| 23-19<br>— | Reserved. |
| 18-8<br>LENGTH_EXT | Extracted Length Field<br>The read-only register reflects the contents of the LENGTH field of the Header of the most-recently received packet. This is the raw, unmodified LENGTH field, as it appears in the received bit stream, unmodified by LENGTH_ADJ or any other parameter. |
| 7-4<br>— | Reserved. |
| 3-0<br>SEQCMD | Sequence Commands<br>0000b - No Action<br>0001b - TX Start Now<br>0010b - TX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP)<br>0011b - TX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>0100b - TX Cancel -- Cancels pending TX events but do not abort a TX-in-progress<br>0101b - RX Start Now<br>0110b - RX Start @ T1 Timer Compare Match (EVENT_TMR = T1_CMP)<br>0111b - RX Start @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>1000b - RX Stop @ T1 Timer Compare Match (EVENT_TMR = T1_CMP) |

| Field | Function |
|---|---|
| | 1001b - RX Stop @ T2 Timer Compare Match (EVENT_TMR = T2_CMP)<br>1010b - RX Cancel -- Cancels pending RX events but do not abort a RX-in-progress<br>1011b - Abort All - Cancels all pending events and abort any sequence-in-progress<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

# A.3.3.8 TRANSCEIVER STATUS (XCVR_STS)

## A.3.3.8.1 Offset

| Register | Offset |
|---|---|
| XCVR_STS | 18h |

## A.3.3.8.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LQI | | | | | | | | RSSI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VALID | LQI_VALID | 0 | RX_IN_WARMDN | RX_IN_PROGRESS | RX_IN_SEARCH | RX_IN_WARMUP | RX_STOP_T2_PEND | RX_STOP_T1_PEND | RX_START_T2_PEND | RX_START_T1_PEND | TX_IN_WARMDN | TX_IN_PROGRESS | TX_IN_WARMUP | TX_START_T2_PEND | TX_START_T1_PEND |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u |

## A.3.3.8.3 Fields

| Field | Function |
|---|---|
| 31-24<br>LQI | Link Quality Indicator<br>This field is valid when LQI_VALID=1. LQI is a unsigned, unitless value. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 23-16<br>RSSI | Received Signal Stength Indicator, in dBm |
| 15<br>CRC_VALID | CRC Valid Indicator<br>CRC Valid indicator for RX packets.<br>0b - CRC is not valid for RX packet.<br>1b - CRC is valid for RX packet. |
| 14<br>LQI_VALID | LQI Valid Indicator<br>LQI Valid indicator for RX packets. This bit becomes set when the LQI computation completes for the packet currently being received, and remains set for the remainder of the packet.<br>0b - LQI is not yet valid for RX packet.<br>1b - LQI is valid for RX packet. |
| 13<br>— | Reserved. |
| 12<br>RX_IN_WARMDN | RX Warmdown Status<br>RX Sequence in TSM Warmdown |
| 11<br>RX_IN_PROGRESS | RX in Progress Status<br>RX Packet Reception Currently Underway |
| 10<br>RX_IN_SEARCH | RX Search Status<br>RX Sequence in Network Address Search |
| 9<br>RX_IN_WARMUP | RX Warmup Status<br>RX Sequence in TSM Warmup |
| 8<br>RX_STOP_T2_PEND | RX T2 Start Pending Status<br>RX Sequence will stop @ next T2 Match |
| 7<br>RX_STOP_T1_PEND | RX T1 Stop Pending Status<br>RX Sequence will stop @ next T1 Match |
| 6<br>RX_START_T2_PEND | RX T2 Start Pending Status<br>RX Sequence will start @ next T2 Match |
| 5<br>RX_START_T1_PEND | RX T1 Start Pending Status<br>RX Sequence will start @ next T1 Match |
| 4<br>TX_IN_WARMDN | TX Warmdown Status<br>TX Sequence in TSM Warmdown |
| 3<br>TX_IN_PROGRESS | TX in Progress Status<br>TX Packet Transmission Currently Underway |
| 2 | TX Warmup Status |

*Table continues on the next page...*

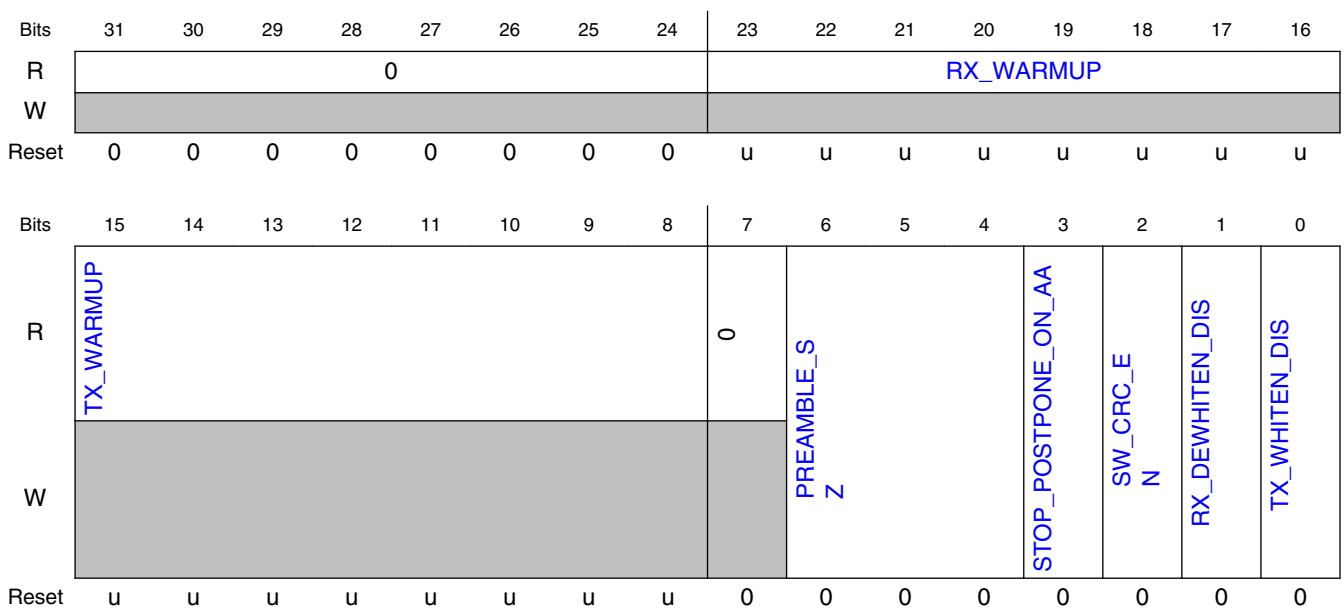
| Field | Function |
|---|---|
| TX_IN_WARMUP | TX Sequence in TSM Warmup |
| 1<br>TX_START_T2_PEND | TX T2 Start Pending Status<br>TX Sequence will start @ next T2 Match |
| 0<br>TX_START_T1_PEND | TX T1 Start Pending Status<br>TX Sequence will start @ next T1 Match |

## A.3.3.9 TRANSCEIVER CONFIGURATION (XCVR_CFG)

### A.3.3.9.1 Offset

| Register | Offset |
|---|---|
| XCVR_CFG | 1Ch |

### A.3.3.9.2 Diagram



### A.3.3.9.3 Fields

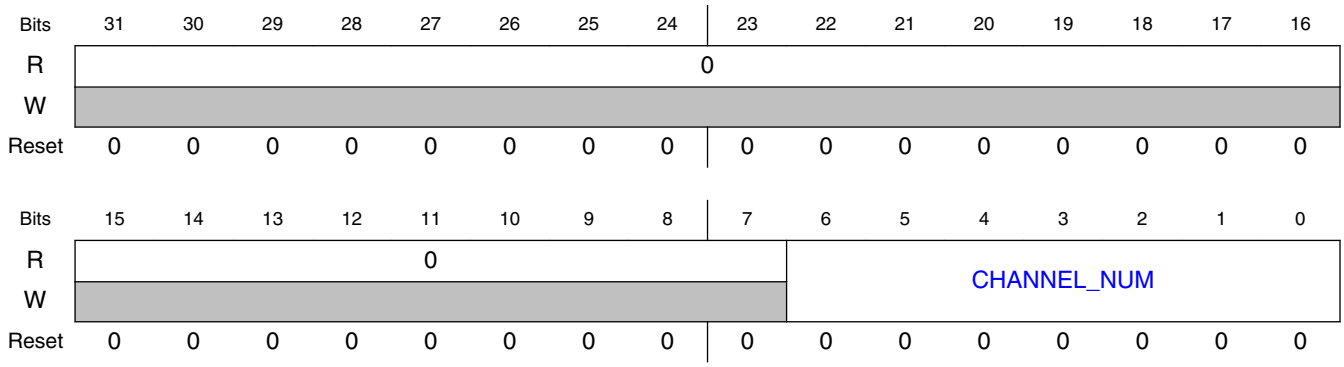
| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-16<br>RX_WARMUP | Receive Warmup Time<br>RX warmup time, in microseconds, provided to Link Layer software, so that warmup time can be accounted for when scheduling over-the-air transactions. |
| 15-8<br>TX_WARMUP | Transmit Warmup Time<br>TX warmup time, in microseconds, provided to Link Layer software, so that warmup time can be accounted for when scheduling over-the-air transactions. |
| 7<br>— | Reserved. |
| 6-4<br>PREAMBLE_SZ | Preamble Size<br>Number of Octets = PREAMBLE_SZ + 1, where 0 <= PREAMBLE_SZ <= 7 |
| 3<br>STOP_POSTPONE_ON_AA | Postpone Stop Command Timeout On Access Address Match Enable<br>If this bit is set, if a RX STOP condition occurs, due to an Event Timer match to a previously issued RX_STOP_T1 or RX_STOP_T2 sequence command, while a packet is being received (*ntw_adr_matched* = 1), the timeout-related abort will be deferred untl the packet reception is completed. If the packet is good, HW will signal Data Indication; otherwise the Sequence Manager will return to IDLE and await further commanding.<br>0b - STOP Abort will occur on RX_STOP_T1 or RX_STOP_T1 Event Timer match, regardless of ntw_adr_matched<br>1b - STOP Abort will be deferred on RX_STOP_T1 or RX_STOP_T1 Event Timer match, if ntw_adr_matched is asserted; otherwise the RX_STOP Abort will occur immediately |
| 2<br>SW_CRC_EN | Software CRC Enable<br>Software override of the HW-computed CRC for TX. Software must write CRC to Packet Buffer (RAM) |
| 1<br>RX_DEWHITEN_DIS | RX De-Whitening Disable<br>Disable all de-whitening on RX packets |
| 0<br>TX_WHITEN_DIS | TX Whitening Disable<br>Disable all whitening on TX packets |

## A.3.3.10 CHANNEL NUMBER (CHANNEL_NUM)

### A.3.3.10.1 Offset

| Register | Offset |
|---|---|
| CHANNEL_NUM | 20h |

### A.3.3.10.2 Diagram




### A.3.3.10.3 Fields

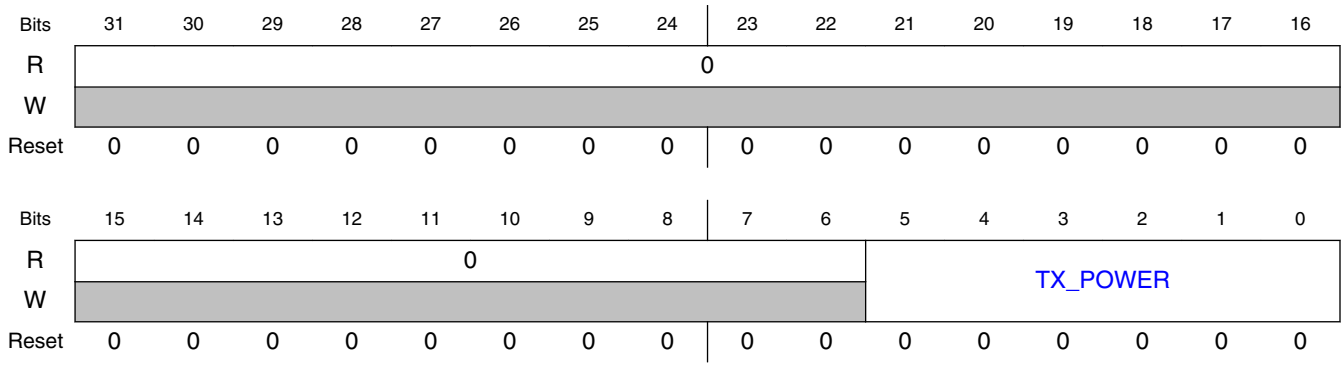
| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-0<br>CHANNEL_NUM | Channel Number<br>RF Channel Select: 0 <= CHANNEL_NUM <= 127; Formula: F = (2360 + CHANNEL_NUM) [in MHz] |

## A.3.3.11 TRANSMIT POWER (TX_POWER)

### A.3.3.11.1 Offset

| Register | Offset |
|---|---|
| TX_POWER | 24h |

### A.3.3.11.2 Diagram




### A.3.3.11.3 Fields

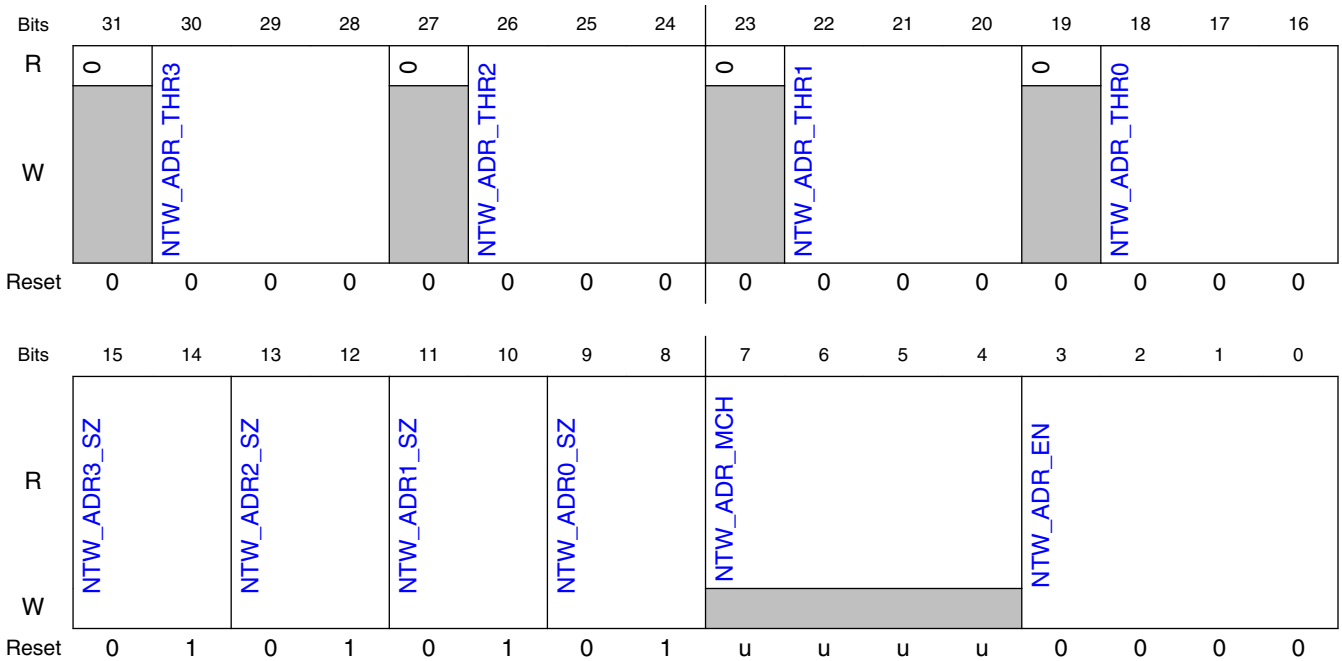
| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5-0<br>TX_POWER | Transmit Power<br>PA Power Level. |

## A.3.3.12 NETWORK ADDRESS CONTROL (NTW_ADR_CTRL)

### A.3.3.12.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_CTRL | 28h |

## A.3.3.12.2 Diagram




## A.3.3.12.3 Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30-28<br>NTW_ADR_THR3 | Network Address 3 Threshold<br>Number of Tolerated bit errors for Network Address 3 |
| 27<br>— | Reserved. |
| 26-24<br>NTW_ADR_THR2 | Network Address 2 Threshold<br>Number of Tolerated bit errors for Network Address 2 |
| 23<br>— | Reserved. |
| 22-20<br>NTW_ADR_THR1 | Network Address 1 Threshold<br>Number of Tolerated bit errors for Network Address 1 |
| 19<br>— | Reserved. |
| 18-16 | Network Address 0 Threshold<br>Number of Tolerated bit errors for Network Address 0 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| NTW_ADR_THR0 | |
| 15-14<br>NTW_ADR3_SZ | Network Address 3 Size<br>00b - Network Address 3 requires a 8-bit correlation<br>01b - Network Address 3 requires a 16-bit correlation<br>10b - Network Address 3 requires a 24-bit correlation<br>11b - Network Address 3 requires a 32-bit correlation |
| 13-12<br>NTW_ADR2_SZ | Network Address 2 Size<br>00b - Network Address 2 requires a 8-bit correlation<br>01b - Network Address 2 requires a 16-bit correlation<br>10b - Network Address 2 requires a 24-bit correlation<br>11b - Network Address 2 requires a 32-bit correlation |
| 11-10<br>NTW_ADR1_SZ | Network Address 1 Size<br>00b - Network Address 1 requires a 8-bit correlation<br>01b - Network Address 1 requires a 16-bit correlation<br>10b - Network Address 1 requires a 24-bit correlation<br>11b - Network Address 1 requires a 32-bit correlation |
| 9-8<br>NTW_ADR0_SZ | Network Address 0 Size<br>00b - Network Address 0 requires a 8-bit correlation<br>01b - Network Address 0 requires a 16-bit correlation<br>10b - Network Address 0 requires a 24-bit correlation<br>11b - Network Address 0 requires a 32-bit correlation |
| 7-4<br>NTW_ADR_MCH | Network Address Match<br>Indicates which of the 4 Network Addresses has matched in the PHY. Valid during an RX sequence at the point of match, and remains asserted until either:<br>1. The next RX sequence begins (if the current packet passed CRC and header filtering), or,<br>2. An RX recycle to Network Address search (if the current packet failed CRC or header filtering)<br>0001b - Network Address 0 has matched<br>0010b - Network Address 1 has matched<br>0100b - Network Address 2 has matched<br>1000b - Network Address 3 has matched |
| 3-0<br>NTW_ADR_EN | Network Address Enable<br>Enable Network Address N for PHY correlation, where 0 ≤ N ≤ 3. Any bit combination can be set.<br>0001b - Enable Network Address 0 for correlation<br>0010b - Enable Network Address 1 for correlation<br>0100b - Enable Network Address 2 for correlation<br>1000b - Enable Network Address 3 for correlation |

## A.3.3.13 NETWORK ADDRESS 0 (NTW_ADR_0)

### A.3.3.13.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_0 | 2Ch |

### A.3.3.13.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_0 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_0 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

### A.3.3.13.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_0 | Network Address 0<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[0]] = 1 |

## A.3.3.14 NETWORK ADDRESS 1 (NTW_ADR_1)

### A.3.3.14.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_1 | 30h |

### A.3.3.14.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_1 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_1 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

#### A.3.3.14.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_1 | Network Address 1<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[1]] = 1 |

## A.3.3.15 NETWORK ADDRESS 2 (NTW_ADR_2)

#### A.3.3.15.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_2 | 34h |

#### A.3.3.15.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_2 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_2 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

#### A.3.3.15.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_2 | Network Address 2<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[2]] = 1 |

## A.3.3.16 NETWORK ADDRESS 3 (NTW_ADR_3)

### A.3.3.16.1 Offset

| Register | Offset |
|---|---|
| NTW_ADR_3 | 38h |

### A.3.3.16.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_3 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NTW_ADR_3 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

### A.3.3.16.3 Fields

| Field | Function |
|---|---|
| 31-0<br>NTW_ADR_3 | Network Address 2<br>The PHY will search for this Network Address if NTW_ADR_CTRL[NTW_ADR_EN[3]] = 1 |

## A.3.3.17 RECEIVE WATERMARK (RX_WATERMARK)

### A.3.3.17.1 Offset

| Register | Offset |
|---|---|
| RX_WATERMARK | 3Ch |

### A.3.3.17.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | BYTE_COUNTER | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | RX_WATERMARK | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### A.3.3.17.3 Fields

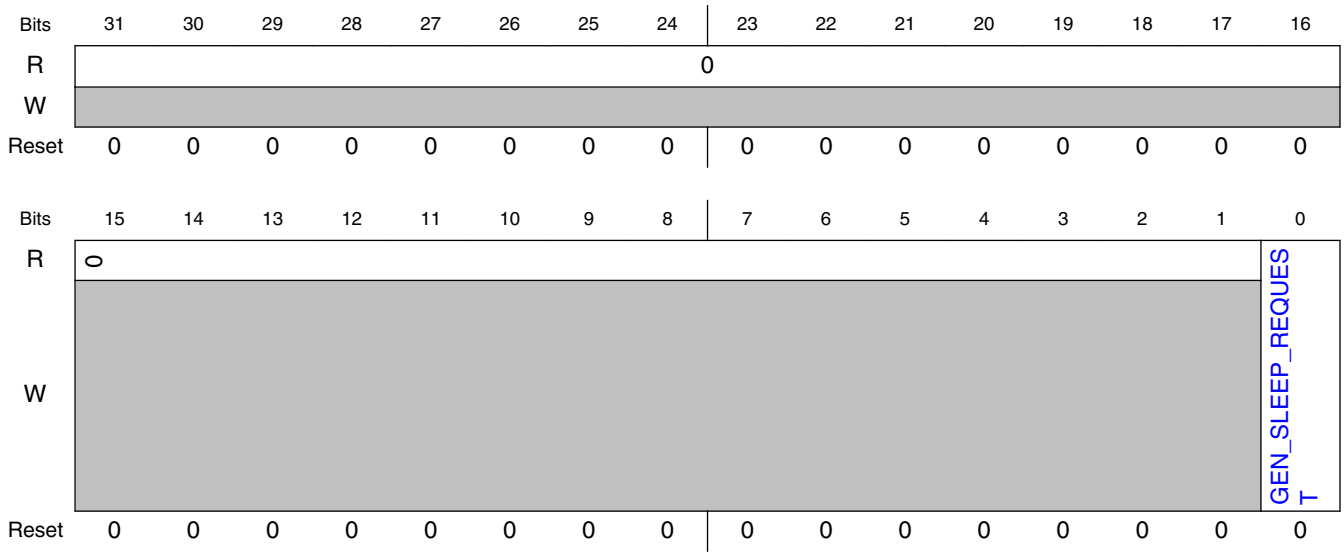
| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-16<br>BYTE_COUNTER | Byte Counter<br>Reflects the current Byte Count, for TX and RX. This is a signed, twos-complement value. For values less than zero, indicates the preamble transmission is underway (TX only). A value of 0 indicates the first octet of Network Address is being transmitted or received. A value of 1 indicates the second octet of Network Address is being transmitted or received. Etc. |
| 15-13<br>— | Reserved. |
| 12-0<br>RX_WATERMARK | Receive Watermark<br>Sets the trigger for RX_WATERMARK_IRQ. Trigger the RX_WATERMARK_IRQ when: RX Byte Counter == RX_WATERMARK[12:0] |

## A.3.3.18 DSM CONTROL (DSM_CTRL)

### A.3.3.18.1 Offset

| Register | Offset |
|---|---|
| DSM_CTRL | 40h |

## A.3.3.18.2 Diagram




## A.3.3.18.3 Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved. |
| 0<br>GEN_SLEEP_REQUEST | GENERIC_FSK Deep Sleep Mode Request<br>If GENERIC_FSK is assigned to Manual DSM (XCVR_CTRL[MAN_DSM_SEL] = 2), setting GEN_SLEEP_REQUEST to 1 enables Deep Sleep Mode (DSM) to be entered when the RSIM DSM_TIMER matches the RSIM MAN_SLEEP register. |

# A.3.3.19 PART ID (PART_ID)

## A.3.3.19.1 Offset

| Register | Offset |
|---|---|
| PART_ID | 44h |

### A.3.3.19.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PART_ID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### A.3.3.19.3 Fields

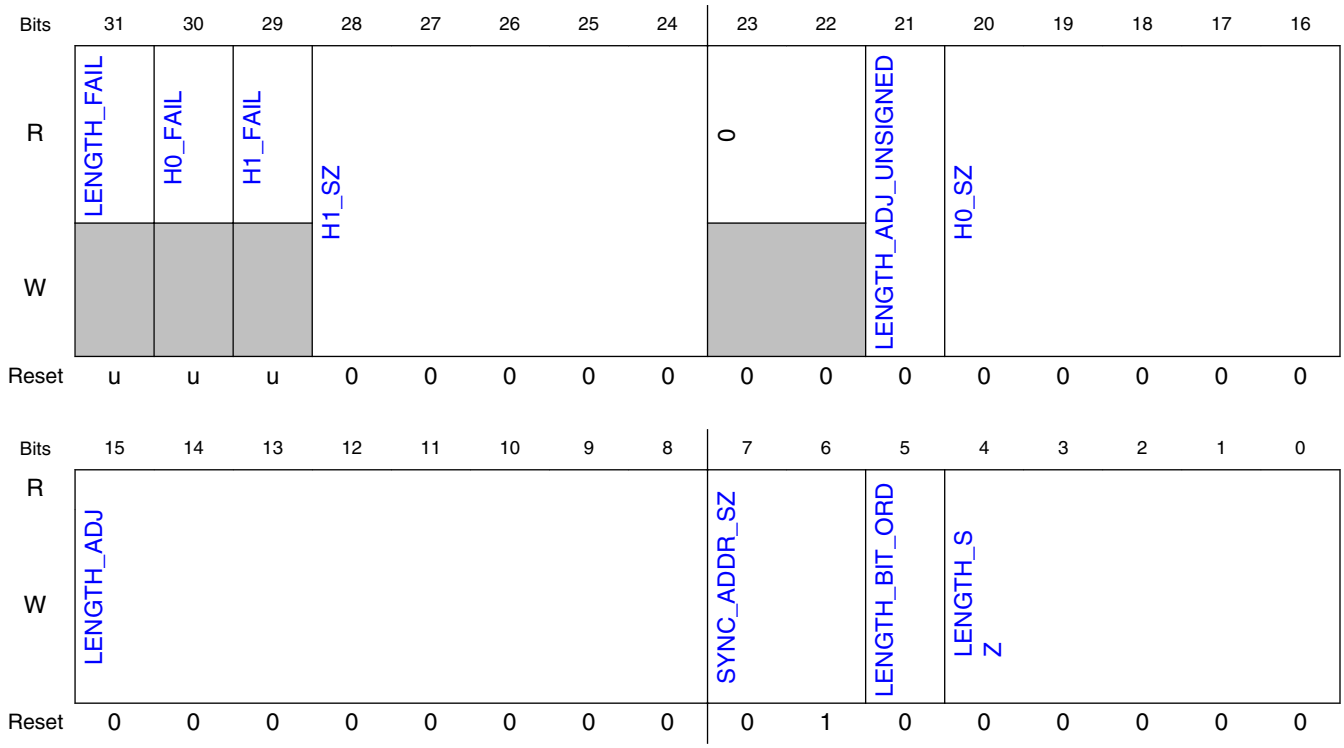
| Field | Function |
|---|---|
| 31-8<br>— | Reserved. |
| 7-0<br>PART_ID | Part ID<br>Part ID to identify HW revision of the Generic FSK Link Layer |

## A.3.3.20 PACKET CONFIGURATION (PACKET_CFG)

### A.3.3.20.1 Offset

| Register | Offset |
|---|---|
| PACKET_CFG | 60h |

## A.3.3.20.2 Diagram




## A.3.3.20.3 Fields

| Field | Function |
|---|---|
| 31<br>LENGTH_FAIL | Maximum Length Violated Status Bit<br>For packets received with REC_BAD_PKT=1, LENGTH_FAIL indicates the extracted LENGTH header field exceeded LENGTH_MAX |
| 30<br>H0_FAIL | H0 Violated Status Bit<br>For packets received with REC_BAD_PKT=1, H0_FAIL indicates the received H0 header field violates the H0_MASK/H0_MATCH pattern |
| 29<br>H1_FAIL | H1 Violated Status Bit<br>For packets received with REC_BAD_PKT=1, H1_FAIL indicates the received H1 header field violates the H1_MASK/H1_MATCH pattern |
| 28-24<br>H1_SZ | H1 Size<br>Length of H1 in bits; 0 ≤ H1_SZ ≤ 16 |
| 23-22<br>— | Reserved. |
| 21<br>LENGTH_ADJ_UNSIGNED | Length Adjustment Unsigned Enabled<br>0b - Hardware interprets LENGTH_ADJ as a signed integer (default)<br>1b - Hardware interprets LENGTH_ADJ as a unsigned integer |
| 20-16<br>H0_SZ | H0 Size<br>Size of H0 in bits; 0 ≤ H0_SZ ≤ 16 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-8<br>LENGTH_ADJ | Length Adjustment<br>Signed Adjustment to the LENGTH field for TX and RX. A value of 0 (default) means LENGTH is interpreted as PAYLOAD + CRC |
| 7-6<br>SYNC_ADDR_SZ | Sync Address Size<br>Number of Octets = SYNC_ADDR_SZ + 1, 0 ≤ SYNC_ADDR_SZ ≤ 3. |
| 5<br>LENGTH_BIT_ORD | LENGTH Bit Order<br>Bit order for the LENGTH field of the header<br>0b - LS Bit First<br>1b - MS Bit First |
| 4-0<br>LENGTH_SZ | LENGTH Size<br>Size of LENGTH field of the header, in bits; 0 ≤ LENGTH_SZ ≤ 16 |

## A.3.3.21 H0 CONFIGURATION (H0_CFG)

### A.3.3.21.1 Offset

| Register | Offset |
|---|---|
| H0_CFG | 64h |

### A.3.3.21.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H0_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H0_MATCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.3.3.21.3 Fields

| Field | Function |
|---|---|
| 31-16<br>H0_MASK | H0 Mask Register |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | For each bit that is set to 1, the received H0 field must match the corresponding bit of H0_MATCH[15:0], else the received packet is rejected. |
| 15-0<br>H0_MATCH | H0 Match Register<br>For each bit of H0_MASK[15:0] that is set to 1, the received H0 field must match this register, else the received packet is rejected. |

## A.3.3.22 H1 CONFIGURATION (H1_CFG)

### A.3.3.22.1 Offset

| Register | Offset |
|---|---|
| H1_CFG | 68h |

### A.3.3.22.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H1_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | H1_MATCH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.3.3.22.3 Fields

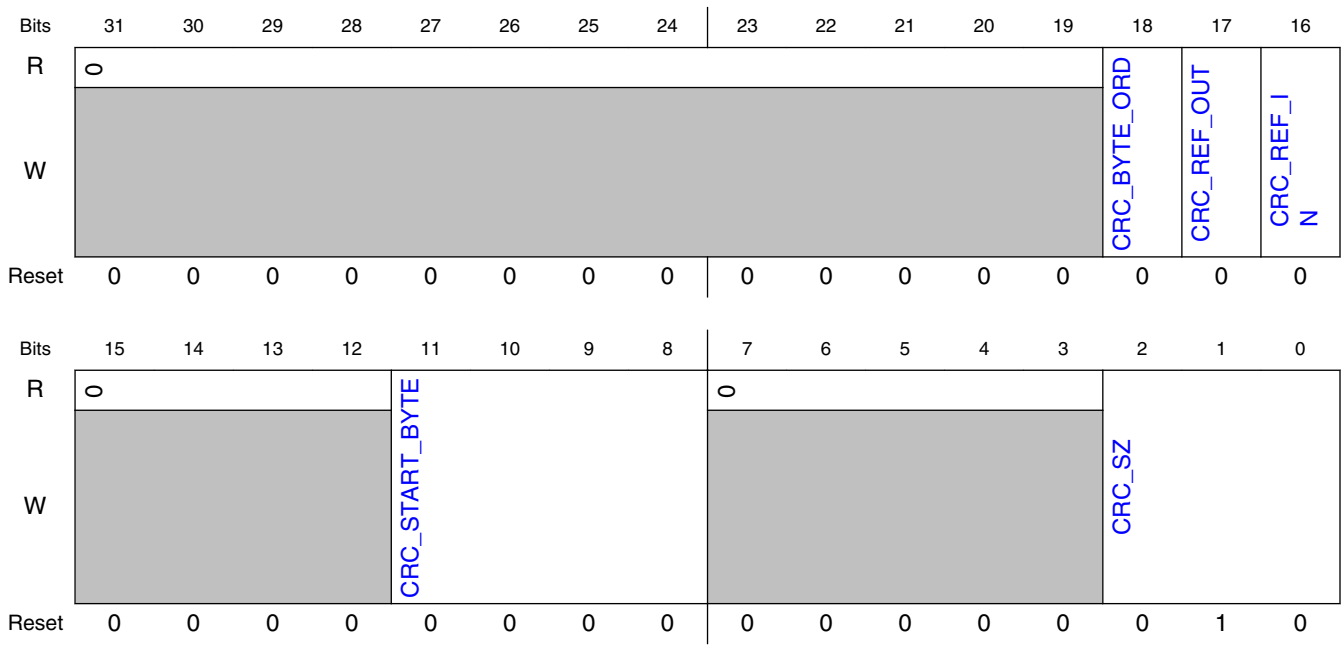
| Field | Function |
|---|---|
| 31-16<br>H1_MASK | H1 Mask Register<br>For each bit that is set to 1, the received H1 field must match the corresponding bit of H1_MATCH[15:0], else the received packet is rejected. |
| 15-0<br>H1_MATCH | H1 Match Register<br>For each bit of H1_MASK[15:0] that is set to 1, the received H1 field must match this register, else the received packet is rejected. |

# A.3.3.23 CRC CONFIGURATION (CRC_CFG)

## A.3.3.23.1 Offset

| Register | Offset |
|---|---|
| CRC_CFG | 6Ch |

## A.3.3.23.2 Diagram




## A.3.3.23.3 Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved. |
| 18<br>CRC_BYTE_ORD | CRC Byte Order<br>0b - LS Byte First<br>1b - MS Byte First |
| 17<br>CRC_REF_OUT | CRC Reflect Out<br>0b - do not manipulate CRC result<br>1b - CRC result is to be reflected bitwise (operated on entire word) |
| 16<br>CRC_REF_IN | CRC Reflect In<br>0b - do not manipulate input data stream<br>1b - reflect each byte in the input stream bitwise |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 15-12<br>— | Reserved. |
| 11-8<br>CRC_START_BYTE | Configure CRC Start Point<br>Start CRC with this byte position. Byte #0 is the first byte of Sync Address |
| 7-3<br>— | Reserved. |
| 2-0<br>CRC_SZ | CRC Size (in octets)<br>Number of CRC Octets = CRC_SZ, 0 ≤ CRC_SZ ≤ 4. |

# A.3.3.24 CRC INITIALIZATION (CRC_INIT)

## A.3.3.24.1 Offset

| Register | Offset |
|---|---|
| CRC_INIT | 70h |

## A.3.3.24.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_SEED | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_SEED | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## A.3.3.24.3 Fields

| Field | Function |
|---|---|
| 31-0<br>CRC_SEED | CRC Seed Value<br>Initial Value for CRC LFSR |

## A.3.3.25 CRC POLYNOMIAL (CRC_POLY)

### A.3.3.25.1 Offset

| Register | Offset |
|---|---|
| CRC_POLY | 74h |

### A.3.3.25.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POLY | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POLY | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.3.3.25.3 Fields

| Field | Function |
|---|---|
| 31-0<br>CRC_POLY | CRC Polynomial. |

## A.3.3.26 CRC XOR OUT (CRC_XOR_OUT)

### A.3.3.26.1 Offset

| Register | Offset |
|---|---|
| CRC_XOR_OUT | 78h |

### A.3.3.26.2 Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_XOR_OUT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_XOR_OUT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### A.3.3.26.3 Fields

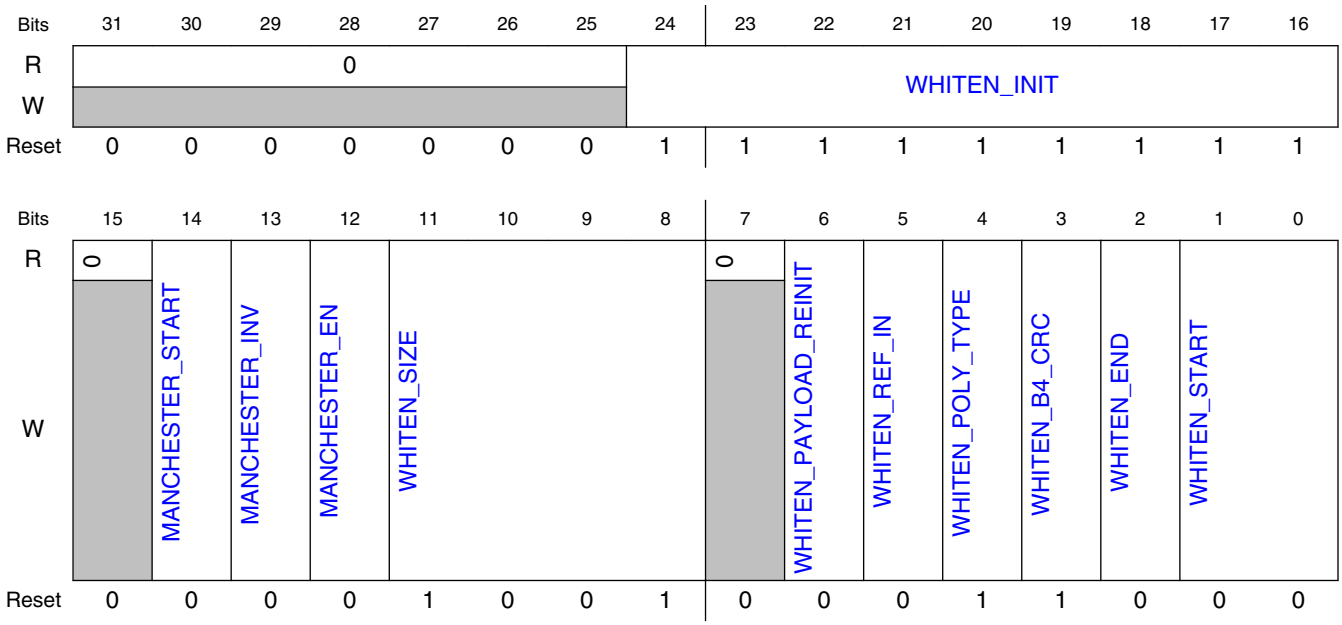
| Field | Function |
|---|---|
| 31-0<br>CRC_XOR_OUT | CRC XOR OUT Register<br>XOR mask for CRC result (for no mask, should be 0) |

## A.3.3.27 WHITENER CONFIGURATION (WHITEN_CFG)

### A.3.3.27.1 Offset

| Register | Offset |
|---|---|
| WHITEN_CFG | 7Ch |

## A.3.3.27.2 Diagram




## A.3.3.27.3 Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-16<br>WHITEN_INIT | Initialization Value for Whitening/De-whitening |
| 15<br>— | Reserved. |
| 14<br>MANCHESTER_START | Configure Manchester Encoding Start Point<br>0b - Start Manchester coding at start-of-payload<br>1b - Start Manchester coding at start-of-header |
| 13<br>MANCHESTER_INV | Configure for Inverted Manchester Encoding<br>0b - Manchester coding as per 802.3<br>1b - Manchester coding as per 802.3 but with the encoding signal inverted |
| 12<br>MANCHESTER_EN | Configure for Manchester Encoding/Decoding<br>0b - Disable Manchester encoding (TX) and decoding (RX)<br>1b - Enable Manchester encoding (TX) and decoding (RX) |
| 11-8<br>WHITEN_SIZE | Length of Whitener LFSR |
| 7<br>— | Reserved. |
| 6 | Configure for Whitener re-initialization |

*Table continues on the next page...*

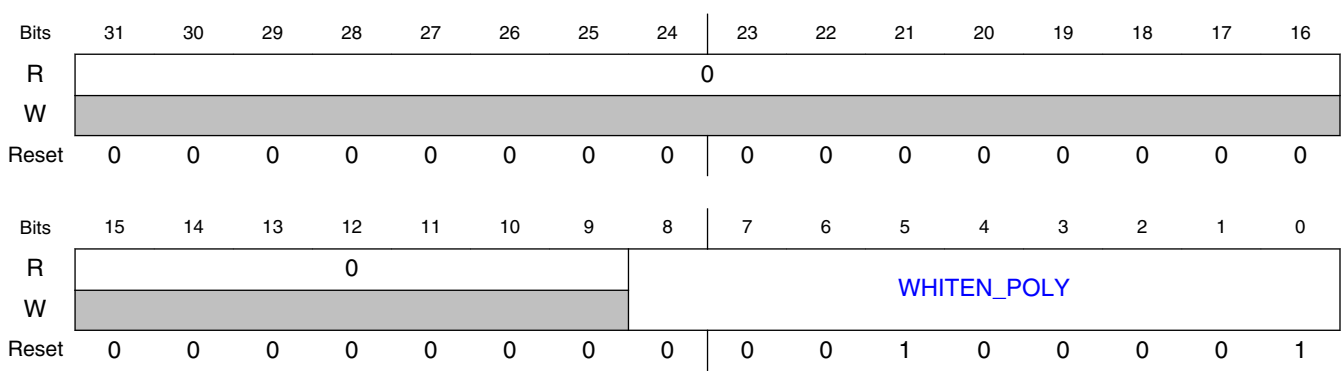
| Field | Function |
|---|---|
| WHITEN_PAYLOAD_REINIT | 0b - Don't re-initialize Whitener LFSR at start-of-payload<br>1b - Re-initialize Whitener LFSR at start-of-payload |
| 5<br>WHITEN_REF_IN | Whiten Reflect Input<br>The input data stream is reflected, bit-wise, per byte, if this register bit is asserted. Bit 7 becomes bit 0, bit 6 becomes bit 1, etc. This register bit will only cause the reflection of the payload data bits as they are used in the whiten calculation and will not cause any change in the output bit order. |
| 4<br>WHITEN_POLY_TYPE | Whiten Polynomial Type<br>A Fibonacci type LFSR is used with the whiten polynomial if this register bit is asserted. Otherwise, a Galois type LFSR is used. |
| 3<br>WHITEN_B4_CRC | Congifure for Whitening-before-CRC<br>Sets the order of Bit Stream Processing for TX and RX.<br>0b - CRC before whiten/de-whiten<br>1b - Whiten/de-whiten before CRC |
| 2<br>WHITEN_END | Configure end-of-whitening<br>0b - end whiten at end-of-payload<br>1b - end whiten at end-of-crc |
| 1-0<br>WHITEN_START | Configure Whitener Start Point<br>00b - no whitening<br>01b - start whitening at start-of-H0<br>10b - start whitening at start-of-H1 but only if LENGTH > WHITEN_SZ_THR<br>11b - start whitening at start-of-payload but only if LENGTH > WHITEN_SZ_THR |

# A.3.3.28 WHITENER POLYNOMIAL (WHITEN_POLY)

## A.3.3.28.1 Offset

| Register | Offset |
|---|---|
| WHITEN_POLY | 80h |

## A.3.3.28.2 Diagram



### A.3.3.28.3 Fields

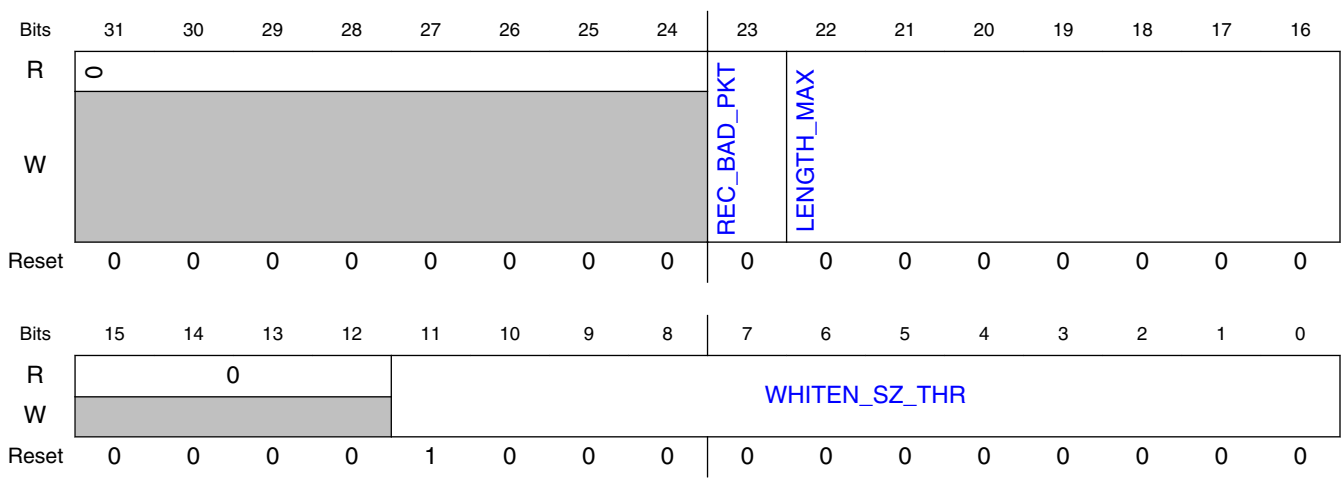
| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8-0<br>WHITEN_POLY | Whitener Polynomial<br>The 9-bit polynomial used in the whiten calculation. The polynomial value must be right-justified if smaller than 9-bits. |

## A.3.3.29 WHITENER SIZE THRESHOLD (WHITEN_SZ_THR)

### A.3.3.29.1 Offset

| Register | Offset |
|---|---|
| WHITEN_SZ_THR | 84h |

### A.3.3.29.2 Diagram




### A.3.3.29.3 Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |

*Table continues on the next page...*

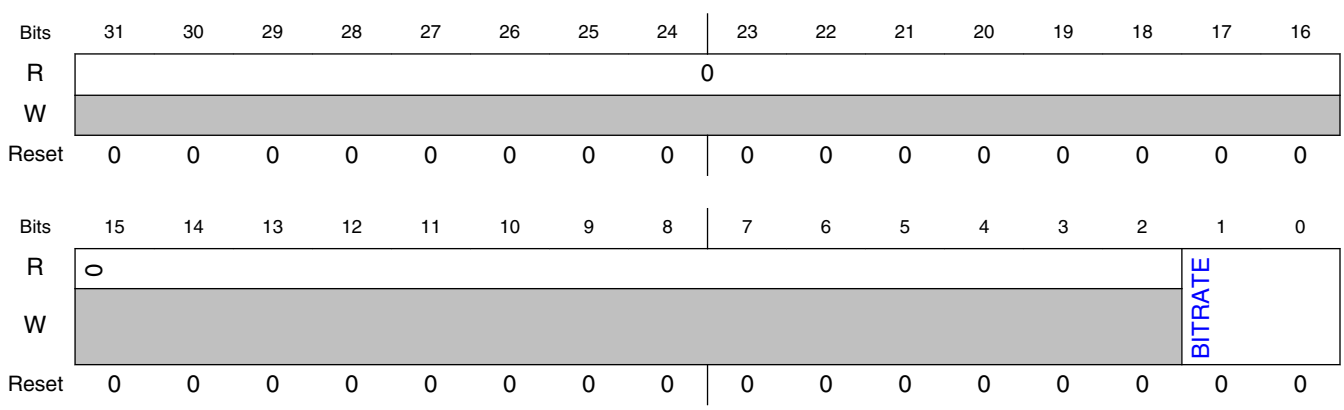
| Field | Function |
|---|---|
| 23<br>REC_BAD_PKT | Receive Bad Packets<br>Enable Packets which fail H0-filtering, H1-filtering, or Maximum Length-filtering, to be fully received without an RX recycle (intended for debug purposes)<br>0b - packets which fail H0, H1, or LENGTH_MAX result in an automatic recycle after the header is received and parsed<br>1b - packets which fail H0, H1, or LENGTH_MAX are received in their entirety |
| 22-16<br>LENGTH_MAX | Maximum Length for Received Packets<br>Sets the Maximum Length Packet that can be received, in multiples of 16 bytes. LENGTH_MAX is compared directly against the extracted LENGTH field of the header (not the adjusted length). LENGTH_MAX=0 (default) is a special case that implies no limit. |
| 15-12<br>— | Reserved. |
| 11-0<br>WHITEN_SZ_THR | Whitener Size Threshold<br>Minimum Packet Length (extracted LENGTH field) required to enable whiten. Requires WHITEN_START=2 or 3 |

# A.3.3.30 BIT RATE (BITRATE)

## A.3.3.30.1 Offset

| Register | Offset |
|---|---|
| BITRATE | 88h |

## A.3.3.30.2 Diagram



### A.3.3.30.3 Fields

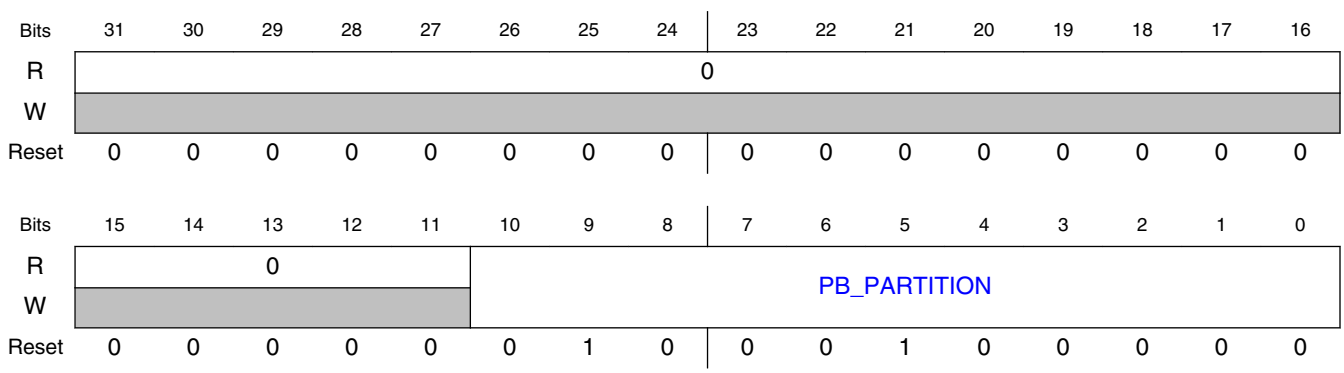
| Field | Function |
|---|---|
| 31-2<br>— | Reserved. |
| 1-0<br>BITRATE | Bit Rate<br>Selects the Bit Rate for the Generic FSK Link Layer. The blocks which interface to the Link Layer (e.g., PHY, CRC/Whitener), also adapt to the selected bit rate.<br>**NOTE:** For FSK FAD (Fast Antenna Diversity), only the 1Mbit/sec bit rate is supported.<br>00b - 1Mbit/sec<br>01b - 500Kbit/sec<br>10b - 250Kbit/sec (not supported if WHITEN_CFG[MANCHESTER_EN]=1)<br>11b - 2Mbit/sec (not supported if WHITEN_CFG[MANCHESTER_EN]=1) |

## A.3.3.31 PACKET BUFFER PARTITION POINT (PB_PARTITION)

### A.3.3.31.1 Offset

| Register | Offset |
|---|---|
| PB_PARTITION | 8Ch |

### A.3.3.31.2 Diagram




### A.3.3.31.3 Fields

| Field | Function |
|---|---|
| 31-11<br>— | Reserved. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 10-0<br>PB_PARTITION | Packet Buffer Partition Point<br>The Packet Buffer Partition Point defines the starting point for the RX segment of the Packet Buffer. The partitioning of the consolidated RAM between TX and RX is controlled by this configurable TX/RX Partition Point. PB_PARTITION can be programmed with any value between 0 (base of RAM0) and 1088 (after the last address of RAM1). The consolidated RAM is partitioned such that the TX buffer resides before the Partition Point (RAM entries 0 to (PB_PARTITION-1)), and the RX buffer resides after it (RAM entries PB_PARTITION to 1087). Programming the Partition Point register to 0 dedicates the entire consolidated RAM to RX; programming the Partition Point register to 1088 dedicates the entire consolidated RAM to TX; programming the Partition Point register to any value between 1 and 1087 yields a split TX/RX buffer. The TX segment (if any) is always first in the Packet Buffer, starting at Packet RAM address 0 of RAM0. Since the Packet Buffer RAM word-width is 2 bytes, the units for PB_PARTITION is "words". (Multiply by 2 to convert to byte addressing) |

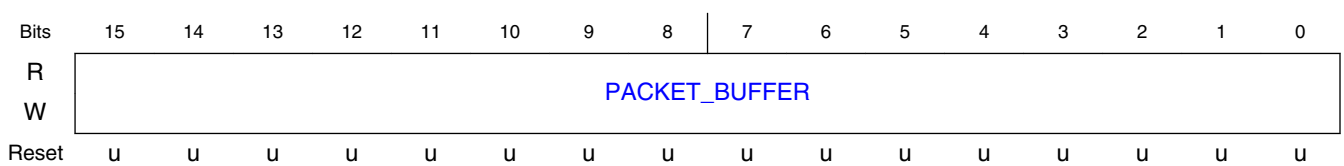
## A.3.3.32 PACKET BUFFER (PACKET_BUFFER_0 - PACKET_BUFFER_1087)

### A.3.3.32.1 Offset

For a = 0 to 1087:

| Register | Offset |
|---|---|
| PACKET_BUFFER_a | 700h + (a × 2h) |

### A.3.3.32.2 Diagram




### A.3.3.32.3 Fields

| Field | Function |
|---|---|
| 15-0<br>PACKET_BUFFER | PACKET BUFFER RAM<br>Storage for packet data. |

***How to Reach Us:***

**Home Page:**
nxp.com

**Web Support:**
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

©2017-2018 NXP B.V.

Document Number RV32M1RM
Revision 1.1, 11/27/2018